

# Instalacion de Django

Enrique García González

14 de abril de 2017

Este documento se trata una guia de como realizar la instalación de Django y los elementos necesarios para el correcto funcionamiento de la aplicación web, así como un pequeño manual para iniciarse con Django

# Índice

1. Instalacion y puesta en marcha	3
2. Organización de un proyecto en Django	4
3. Templates	4
4. Templates	5
5. Añadir css y javascript	5
6. MySQL en Django	6

## 1. Instalacion y puesta en marcha

En primer lugar, para instalar Django en una distribución ubuntu o debian necesitamos contar con una instalación previa de Python. En este caso suponiendo que partimos de una instalación correcta de python 3.5 debemos instalar las siguientes librerías con pip.

```
pip3 install django
pip3 install gunicorn
pip3 install psycopg2
```

Para crear un proyecto de django debemos ejecutar el siguiente comando dentro de la carpeta donde queremos que se almacene

```
django-admin startproject <nombre proyecto>
```

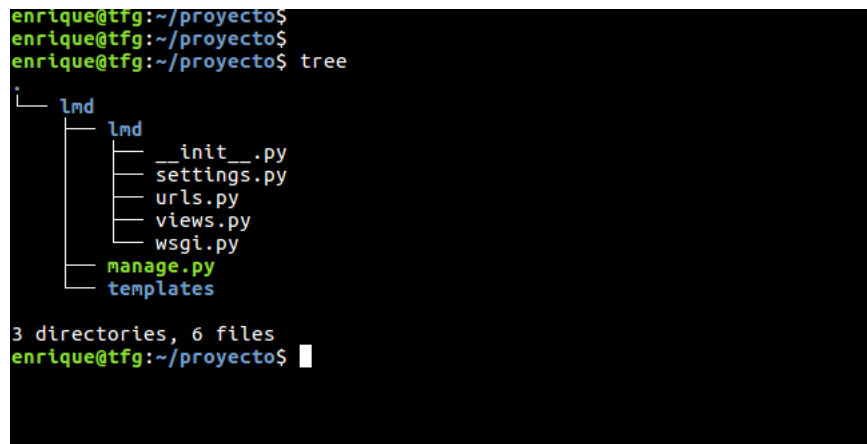
El cual nos genera la estructura básica del proyecto. Ahora entramos en esta carpeta y comprobamos que tenemos un directorio con el mismo nombre del proyecto y un archivo llamado manage.py. En este directorio creamos una carpeta llamada templates, que será la encargada de almacenar el código html del proyecto.

```
mkdir templates
```

Después entramos en el directorio que tiene el mismo nombre que el proyecto el archivo views.py

```
cd <nombre proyecto>
touch views.py
```

Al hacer esto nos tiene que quedar una estructura parecida a la siguiente



```
enrique@tfg:~/proyecto$
enrique@tfg:~/proyecto$
enrique@tfg:~/proyecto$ tree
.
├── lmd
│   ├── lmd
│   │   ├── __init__.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   ├── views.py
│   │   └── wsgi.py
│   ├── manage.py
│   └── templates
3 directories, 6 files
enrique@tfg:~/proyecto$
```

Figura 1: Estructura proyecto

## 2. Organización de un proyecto en Django

Este punto esta creado para orientar un poco sobre la organización de un proyecto en Django a aquellos usuarios que no lo han utilizado nunca.

Django sigue una estructura de MOdelo-ista-Controlador, aunque no usa su propia lógica en la implementacion, ya que la capa Controlador es manejada por el framework. Por esto las partes más interesantes son:

**Model:** Es la capa de acceso a datos. COntiene todo sobre los datos: acceso, validación, relaciones entre ellos,etc.

**Template:** Capa de representación. Contiene la información sobre como se tienen que mostrar las cosas en la página Web

**View:** Esta capa que contiene la lógica de acceso a la capa Model y se la ofrece al Template apropiado. Es como un puente entre las dos capas anteriores.

## 3. Templates

Un template de Django es una cadena de texto que se utiliza para separar la representación de la página de los datos. Contienen también algunas reglas lógicas para regular que partes de la página se tienen que mostrar y se encarga de generar el código html. La forma más comoda y ordenada de hacerlo es crear para cada página un fichero html dentro de la carpeta templates. Para poder utilizar este método tenemos que modificar el archivo settings.py (podemos ver donde se encuentra en la figura 1). Buscamos la parte correspondiente a los templates, que es de la siguiente forma:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

Modificamos la linea DIRS para que quede de la siguiente forma

```
'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

De esta forma ya podemos utilizar la carpeta templates, creada en el apartado 1 para almacenar todos nuestros templates.

Para más información sobre los templates en django:

<http://djangobook.com/django-templates/>  
<http://djangobook.com/basic-template-tags-filters/>  
<http://djangobook.com/templates-in-views/>  
<http://djangobook.com/advanced-templates/>

## 4. Templates

Si queremos que nuestro servidor web sea accesible para otras máquinas, primero debemos modificar el archivo settings.py. Buscamos las líneas donde aparecen DEBUG y ALLOWED\_HOST y las dejamos de la siguiente manera

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = ['*']
```

Con DEBUG=False hacemos que si ocurre un error no aparezca la información de depuración en el navegador, mostrando solamente un mensaje de error

En ALLOWED\_HOSTS podemos añadir las IPs que nos interesen si queremos restringir el acceso a un conjunto de máquinas, o '\*' si queremos que pueda acceder cualquier equipo.

Para iniciar el servidor, ahora nos movemos al directorio en el que se encuentra el archivo manage.py y ejecutamos el siguiente comando

```
python3.5 manage.py runserver 0.0.0.0:<puerto>
```

Y si no ha ocurrido ningún error ya tenemos el servidor web abierto en el puerto indicado.

## 5. Añadir css y javascript

Para utilizar archivos .css y javascript en un proyecto Django, debemos crear una carpeta llamada static en el mismo lugar donde se encuentra la carpeta templates. Dentro de esta carpeta creamos los directorios que queramos para organizar los distintos tipos de archivos.

Ahora debemos modificar el archivo settings.py, la línea que contiene STATIC\_URL y añadimos esto al final

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.10/howto/static-files/
```

```

STATIC_URL = '/static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
    '/var/www/static/',
]

```

Por ultimo, un ejemplo de como añadir un script de javascript en un template:

```

{% load static %}
<script type="text/javascript" src="{% static 'js/codgo.js' %}"></script>

```

## 6. MySQL en Django

Para utilizar MySQL primero debemos instalar los siguientes paquetes:

```

sudo apt-get update
sudo apt-get install python-pip python-dev mysql-server
    libmysqlclient-dev
sudo pip install MySQL-python

```

En el transcurso de la instalación se nos preguntara por la contraseña que queremos darle al usuario root de MySQL

Ahora debemos acceder a la base de datos para hacer unos ajustes, para ello usamos el comando

```
mysql -u root -p
```

Ahora debemos crear una base de datos para nuestro proyecto (es aconsejable que cada proyecto tenga su propia base de datos) con el comando

```
CREATE DATABASE lmd CHARACTER SET UTF8;
```

Y creamos un usuario para que interactue con la base de datos

```
CREATE USER nraik@localhost IDENTIFIED BY '111111';
```

Y le damos todos los privilegios de la base de datos del proyecto

```
GRANT ALL PRIVILEGES ON lmd.* TO nraik@localhost;
```

Y guardamos los cambios realizados

```
FLUSH PRIVILEGES;
```

Ahora tenemos que indicarle al proyecto de Django que vamos a usar MySQL. Para ello buscamos en settings.py la seccion DATABASES y la modificamos de la siguiente manera:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'lmd',  
        'USER': 'nraik',  
        'PASSWORD': '111111',  
        'HOST': 'localhost',  
        'PORT': '',  
    }  
}
```

Por podemos migrar las estructuras de a base de datos antigua de la siguiente forma

```
python3.5 manage.py makemigrations  
python manage.py migrate
```

Y despues creamos el usuario administrador de la siguiente forma

```
python manage.py createsuperuser
```

Para comprobar que se ha realizado el cambio podemos acceder a la página <http://localhost:puerto;/admin> y comprobar que el usuario creado con la orden anterior y su contraseña funcionan