

## Curso Desarrollo Web con Python y Django

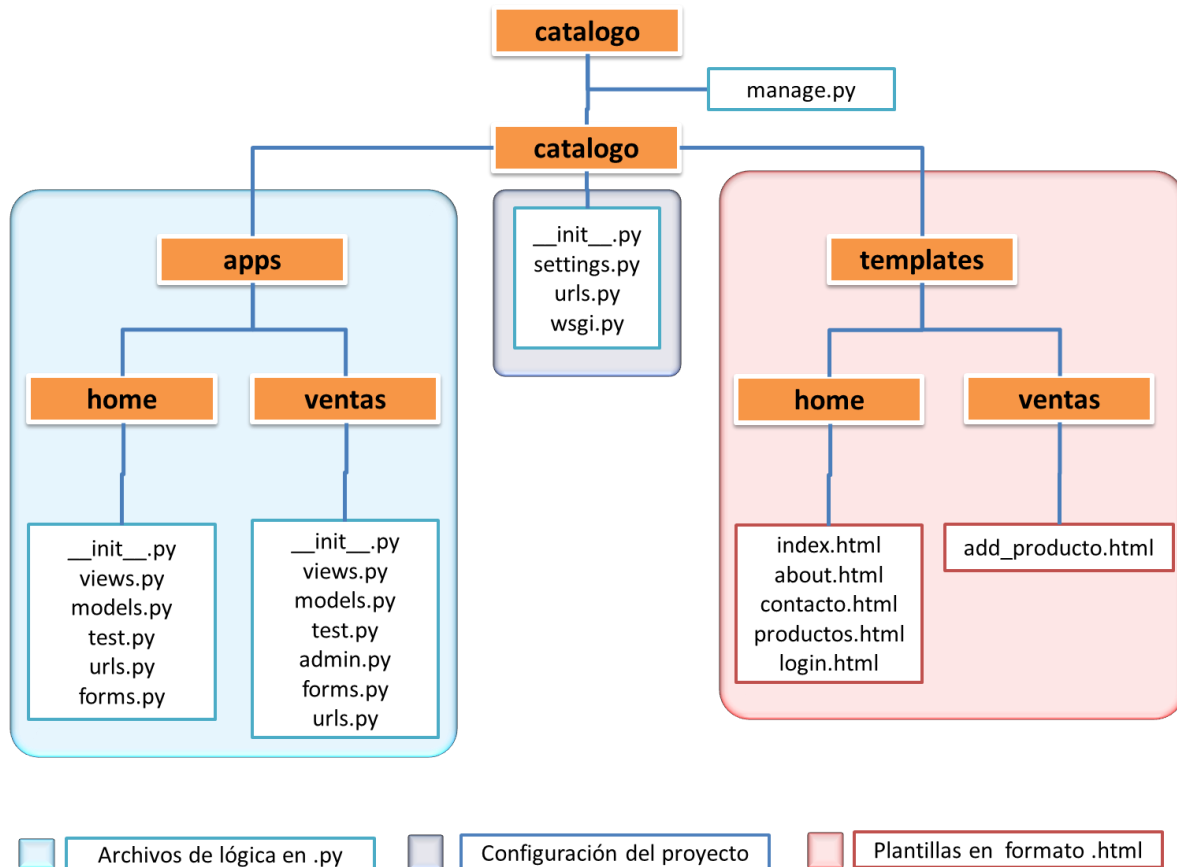
### Guía 4

## Paginas estáticas con Django

Después de crear el proyecto de catálogo vamos a organizar nuestro proyecto

Lo primordial es comprender que para cualquier proceso en Django para la elaboración de una página debe tener URL, VISTA y TEMPLATE

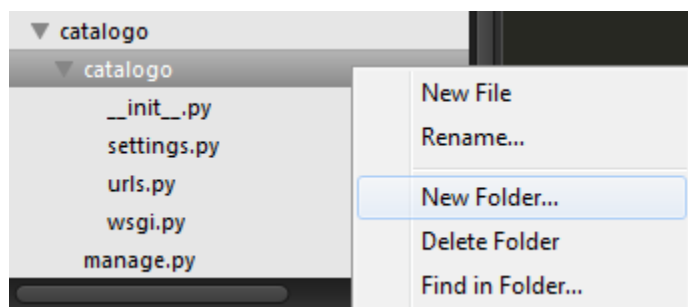
Vamos a organizar nuestro proyecto con la siguiente estructura



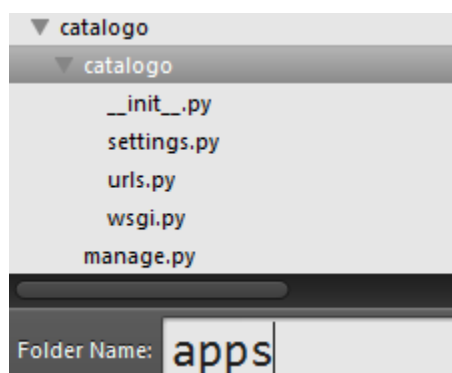
## Creación de Apps

Para crear apps para nuestro proyecto `catalogo` en Django debemos ingresar a nuestra carpeta del proyecto “`catalogo/catalogo`”

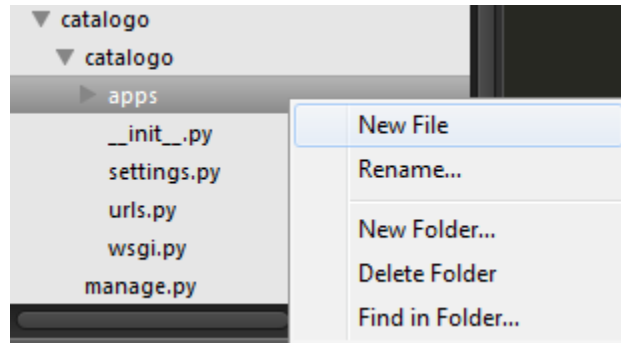
Primero que todo creamos una carpeta llamada **apps** la cual nos va a permitir organizada las apps que iremos creando



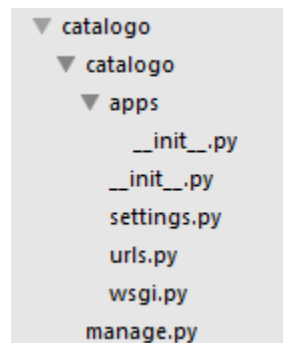
Ingresamos el nombre de nuestra carpeta **apps** y presionamos **enter**



Luego creamos un archivo vacío llamado `__init__.py`



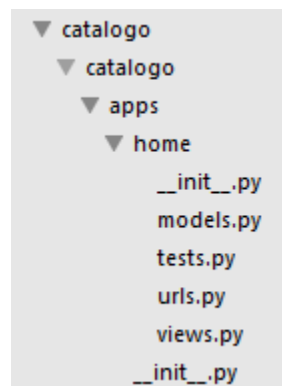
Luego presionamos `CRTL+ S` y le ponemos como nombre del archivo `__init__.py`



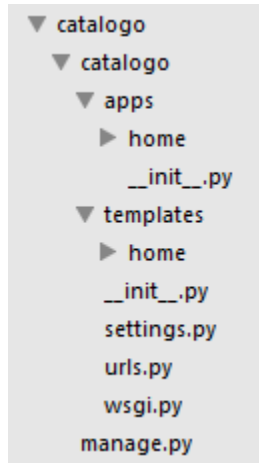
Ahora crearemos una aplicación para nuestro proyecto desde nuestro **CMD** ingresamos

**`django-admin.py startapp home`**

Una vez creada nuestra app home vemos que nos crea una carpeta llamada home y dentro de esa carpeta hay 4 archivos como se muestra en la siguiente imagen



Ahora dentro de la segunda carpeta de `catalogo` vamos a crear las carpetas `templates` y `home` (esta última debe estar dentro de la carpeta `templates`)



Una vez tengamos la arquitectura del proyecto organizado vamos a configurar en el **settings.py** y luego crear nuestra primera pagina

### Settings.py

```
import os
```

```
TEMPLATE_DIRS = (
    os.path.join(os.path.dirname(__file__), 'templates'),

    # Put strings here, like "/home/html/django_templates" or "C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)
```

Para el siguiente paso se debe crear un archivo de `urls.py` en la aplicación `home` (`catalogo/apps/home/urls.py`)

Vamos a crear una **vista** una **url** y un **template** para una página llamada **about** (sobre nosotros)

1. Ahora vamos a crear una url en (**catalogo/apps/home/urls.py**)

```
1 from django.conf.urls.defaults import patterns, url
2
3 urlpatterns = patterns('catalogo.apps.home.views',
4     url(r'^about/$', 'about_view', name = 'vista_about'),
5 )
```

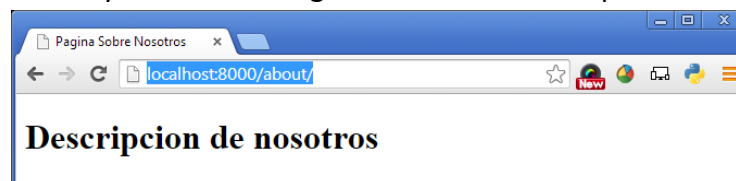
2. Editamos el archivo views.py de la aplicación home (**catalogo/apps/home/views.py**)

```
1 from django.shortcuts import render_to_response
2 from django.template import RequestContext
3
4 def about_view(request):
5     return render_to_response('home/about.html', context_instance = RequestContext(request) )
```

3. Creamos un archivo **about.html** en la carpeta **templates/home**
4. Editamos el archivo **about.html**

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <title>Pagina Sobre Nosotros</title>
5 </head>
6 <body>
7     <h1>Descripcion de nosotros</h1>
8 </body>
9 </html>
```

5. Corremos el servidor y vemos el navegador en la ruta o **url** que creamos

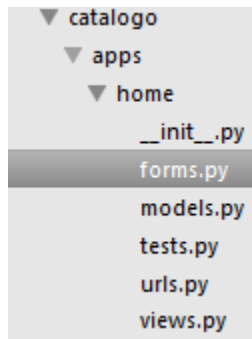


## Formularios y CSRF

Para nuestro proyecto **catalogo** crearemos un formulario

### Crear un Formulario

En nuestra aplicación **home** crearemos un archivo llamado **forms.py** (**catalogo/apps/home/forms.py**)



El archivo **forms.py** debería quedar así

```
from django import forms

class contact_form(forms.Form):
    correo = forms.EmailField(widget = forms.TextInput())
    titulo = forms.CharField(widget = forms.TextInput())
    texto = forms.CharField(widget = forms.Textarea())
```

## Crear una Vista para el Formulario

En nuestro archivo de vistas de la aplicación **home** (**catalogo/apps/home/views.py**)

```
from catalogo.apps.home.forms import contact_form

def contacto_view (request):
    formulario = contact_form()
    ctx = {'form':formulario}
    return render_to_response('home/contacto.html',ctx,context_instance = RequestContext(request))
```

## Crear un Template para el contacto

Creamos el archivo **contacto.html** en la carpeta de **home**  
(**catalogo/templates/home/contacto.html**)

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <title>Contactanos</title>
5 </head>
6 <body>
7     <form action="." method = "POST">
8         {% csrf_token %}
9         {{ form.as_p }}
10        <input type = "submit" value = "submit"/>
11    </form>
12
13 </body>
14 </html>
```

## Crear la Url para el formulario

Ahora agregamos al **urls.py** una url para la vista del formulario de contacto que creamos anteriormente (**catalogo/templates/home/views.py**)

```
from django.conf.urls.defaults import patterns, url

urlpatterns = patterns('catalogo.apps.home.views',
    url(r'^$', 'index_view', name = 'vista_principal'),
    url(r'^about/$', 'about_view', name = 'vista_about'),
    url(r'^productos/$', 'productos_view', name = 'vista_productos'),
    url(r'^contacto/$', 'contacto_view', name = 'vista_contacto'),
)
```

Ejecutamos nuestro servidor y vemos en navegador en la url de **/contacto**



A screenshot of a web browser window titled 'Contactanos'. The address bar shows 'localhost:8000/contacto/'. The form contains three input fields: 'Email:' with the value '123@123.com', 'Titulo:' with the value 'hola', and a large text area labeled 'Texto:' containing the text 'Estoy aprendiendo a desarrollar paginas web dinámicas con Django y Python'. Below the text area is a 'submit' button.



## Formularios, Recepción POST

Ya vimos como enviar información por el método POST por medio de un formulario ahora veremos como RECIBIRLA

### Editar la vista del formulario (contact\_view)

En nuestra aplicación **home** editaremos el archivo **views.py** en la vista de **contacto\_view()** (catalogo/apps/home/views.py)

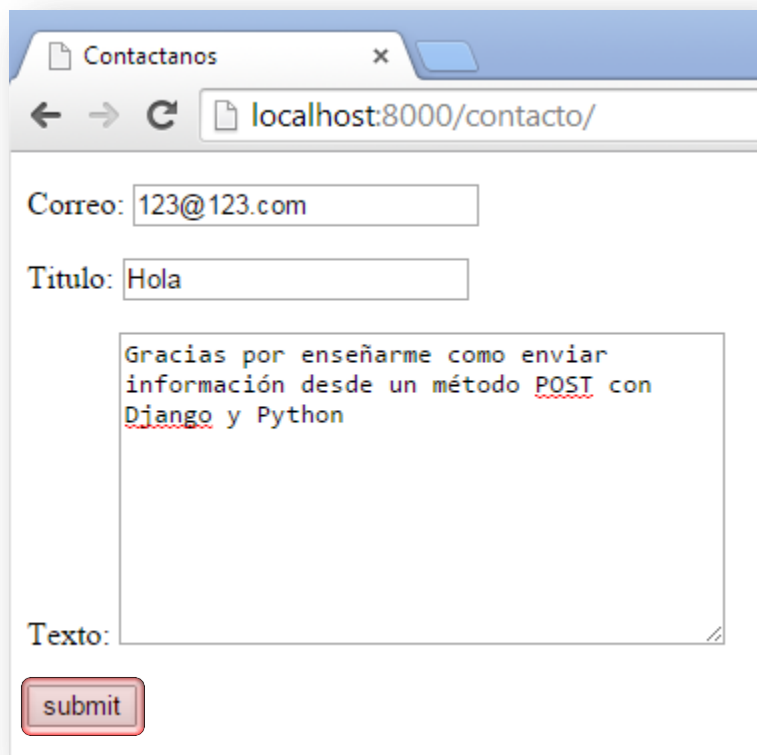
```
def contacto_view(request):
    info_enviado = False #Definir si se envio la informacion o no se envio
    email = ""
    title = ""
    text = ""
    if request.method == "POST": # evalua si el metodo fue POST
        formulario = contact_form(request.POST) #instancia del formulario con los datos ingresados
        if formulario.is_valid(): #evalua si el formulario es valido
            info_enviado = True #la informacion se envio correctamente
            email = formulario.cleaned_data['correo'] # copia el correo ingresado en email
            title = formulario.cleaned_data['titulo'] # copia el titulo ingresado en title
            text = formulario.cleaned_data['texto'] # copia el texto ingresado en text
        else: #si no fue POST entonces fue el metodo GET mostrara un formulario vacio
            formulario = contact_form() # creacion del formulario vacio
    ctx = {'form':formulario, 'email':email, "title":title, "text":text, "info_enviado":info_enviado}
    return render_to_response('home/contacto.html', ctx, context_instance = RequestContext(request))
```

## Editar el Template (contacto.html)

Editamos el archivo **contacto.html** en la carpeta de **home**  
(**catalogo/templates/home/contacto.html**)

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Contactanos</title>
</head>
<body>
  {% if info_enviado %}
    <h2>Gracias por enviar un comentario, nos pondremos en contacto</h2>
    <br>
    <h3>La informacion enviada fue la siguiente</h3>
    <p>Email usado: {{ email }} </p>
    <p>Titulo usado: {{ title }} </p>
    <p>Texto citado: </p> <br> {{ text }}
  {% else %}
    <form action="." method = "POST">
      {% csrf_token %}
      {{ form.as_p }}
      <input type = "submit" value = "submit"/>
    </form>
  {% endif %}
</body>
</html>
```

Ejecutamos nuestro servidor y vemos en navegador en la url de **/contacto**



Correo: 123@123.com

Titulo: Hola

Texto: Gracias por enseñarme como enviar información desde un método POST con Django y Python

submit

Después de seleccionar submit mostrara...



**Gracias por enviar un comentario, nos pondremos en contacto**

**La informacion enviada fue la siguiente**

Email usado: 123@123.com

Titulo usado: Hola

Texto citado:  
Gracias por enseñarme como enviar información desde un método POST con Django y Python

## PLUS

### Envío del formulario de contacto al EMAIL

#### Editar Settings.py

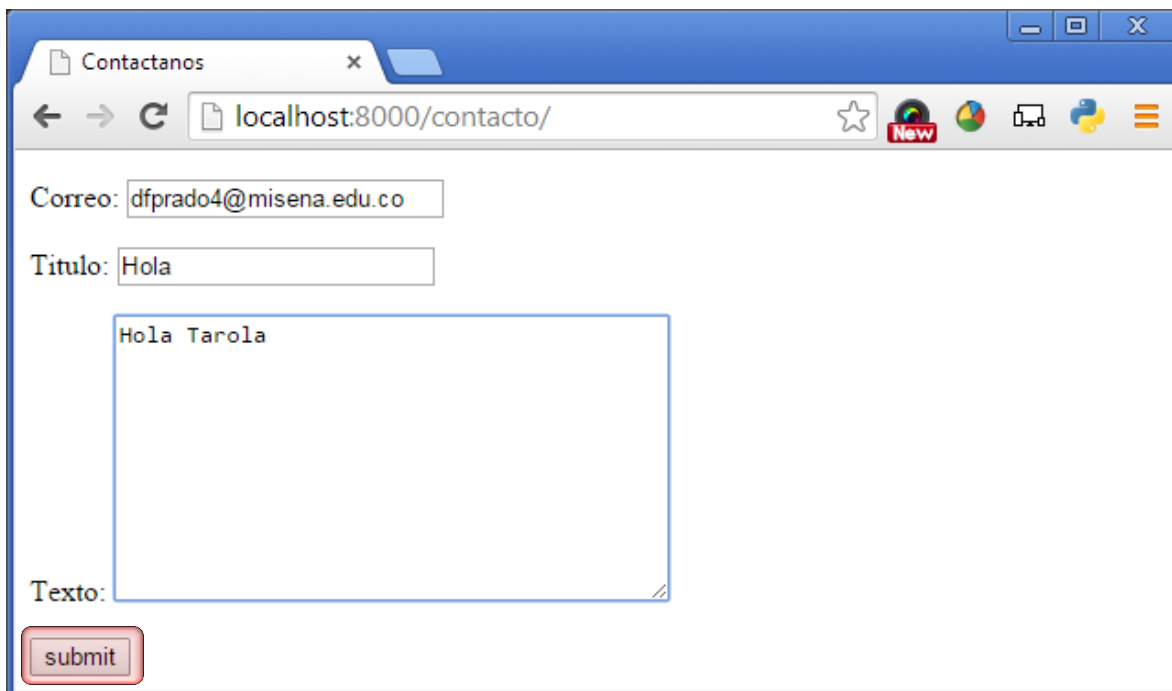
Agregamos al final del archivo la siguiente configuración

```
163 # Aquí va la configuración del servidor de correo GMAIL
164 EMAIL_HOST = 'smtp.gmail.com'
165 EMAIL_PORT = '587'
166 EMAIL_HOST_USER = 'micorreo@gmail.com' #ponen su correo
167 EMAIL_HOST_PASSWORD = 'miclave' #poner la clave de su correo
168 EMAIL_USE_TLS = True
```

#### Editar la vista (contact\_view())

```
from django.core.mail import EmailMultiAlternatives # enviamos HTML

def contacto_view(request):
    info_enviado = False #Definir si se envió la información o no se envió
    email = ""
    title = ""
    text = ""
    if request.method == "POST": # evalúa si el método fue POST
        formulario = contact_form(request.POST) #instancia del formulario con los datos ingresados
        if formulario.is_valid(): #evalúa si el formulario es válido
            info_enviado = True #la información se envió correctamente
            email = formulario.cleaned_data['correo'] # copia el correo ingresado en email
            title = formulario.cleaned_data['titulo'] # copia el título ingresado en title
            text = formulario.cleaned_data['texto'] # copia el texto ingresado en text
            ''' Bloque configuración de envío por GMAIL '''
            to_admin = 'kaoxdc@gmail.com'
            html_content = "Información recibida de %s <br> ---Mensaje--- <br> %s"%(email,text)
            msg = EmailMultiAlternatives('correo de contacto', html_content, 'from@server.com',[to_admin])
            msg.attach_alternative(html_content,'text/html') #definimos el contenido como HTML
            msg.send() #enviamos el correo
            ''' Fin del Bloque '''
        else: #si no fue POST entonces fue el método GET mostrara un formulario vacío
            formulario = contact_form() # creación del formulario vacío
            ctx = {'form':formulario, 'email':email, 'title':title, "text":text, "info_enviado":info_enviado}
            return render_to_response('home/contacto.html',ctx,context_instance = RequestContext(request))
```



A screenshot of a web browser window titled 'Contactanos'. The address bar shows 'localhost:8000/contacto/'. The form contains the following fields:

- Correo:
- Titulo:
- Text area:
- Submit button:

Se mostrara después de seleccionar la opción submit

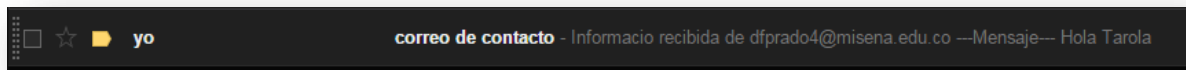


A screenshot of a web browser window showing the confirmation message after submitting the form. The address bar remains 'localhost:8000/contacto/'. The message displayed is:

**Gracias por enviar un comentario, nos pondremos en contacto**

**La informacion enviada fue la siguiente**

- Email usado: dfprado4@misena.edu.co
- Titulo usado: Hola
- Texto citado: Hola Tarola



Abro el correo que configure en **settings.py** y veo el mensaje que acaba de llegar



## Herencia de plantillas

Hasta ahora veremos cómo se hace herencia de plantillas HTML a otras plantillas para que puedan basarse en un **HTML BASE**

Hasta ahora hemos creado varias páginas web entre ellas

- Página de inicio (**localhost:8000**)
- Página de about (**/about**)
- Página de contacto (**/contacto**)

Para todos hemos creado su **vista, url y template (excepto el admin)**, ahora es muy sencillo comprender que estamos construyendo una página web administrable y dinámica de un catálogo de productos.

También debemos tener en cuenta que tenemos organizados nuestros templates dependiendo de la aplicación en la que estemos trabajando.

En base a esto crearemos una plantilla BASE que crearemos en la carpeta templates (**/catalogo/templates/base.html**)

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <title>{% block title %} Titulo Base {% endblock %}</title>
5 </head>
6 <body>
7     <header>
8         <h1>Proyecto - Catalogo de Productos</h1>
9         <nav>
10             <a href="#">Inicio</a>
11             <a href="#">Productos</a>
12             <a href="#">Contacto</a>
13             <a href="#">Acerca de ...</a>
14         </nav>
15     </header>
16     <section>
17         {% block content %}
18         {% endblock %}
19     </section>
20 </body>
21 </html>
```

### Nota:

La línea 17 y 18 subrayadas en la imagen sera el bloque en el cual hemos trabajado en las vistas que hemos creado (inicio, contacto, about, productos, etc.).

Ahora vamos a editar todas nuestras plantillas de **home** (**templates/home/**) es decir los templates ya creados

### index.html

```
1 {% extends 'base.html' %}
2
3 {% block title %}
4     Inicio - Bienvenidos
5 {% endblock %}
6
7 {% block content %}
8     <h1>Bienvenidos esta es la pagina Principal</h1>
9 {% endblock %}
```

### about.html

```
1 {% extends 'base.html' %}
2
3 {% block title %}
4     Pagina Sobre Nosotros
5 {% endblock %}
6
7 {% block content %}
8     <h1>Descripcion de nosotros</h1>
9     <h3>{{ msg }}</h3>
10 {% endblock %}
```

### contacto.html

```
1 {% extends 'base.html' %}
2
3 {% block title %}
4     Contactanos
5 {% endblock %}
6
7 {% block content %}
8     <form action="." method = "POST">
9         {% csrf_token %}
10        {{ form.as_p }}
11        <input type = "submit" value = "submit"/>
12    </form>
13 {% endblock %}
```



## Servidor de Medios

En esta ocasión vamos a ver como levantar un servidor de medios en nuestro proyecto catalogo

Lo primero que debemos hacer es crear una carpeta llamada **media** en (**catalogo/catalogo/media**)

Luego dentro de la carpeta **media** creamos 3 carpetas llamadas

- css
- images
- js

Ahora vamos a **configurar** nuestro servidor de medios en el archivo **settings.py**

```
53 MEDIA_ROOT = os.path.normpath(os.path.join(os.path.dirname(__file__), 'media/'))
54
55 # URL that handles the media served from MEDIA_ROOT. Make sure to use a
56 # trailing slash.
57 # Examples: "http://example.com/media/", "http://media.example.com/"
58 MEDIA_URL = '/media/'
```

Ahora vamos a copiar la imagen (**django.gif**) y la ponemos en la carpeta de **media/images** y en el archivo base.html agregamos la siguiente línea

```
6 <body>
7   <header>
8     
9     <h1>Proyecto - Catalogo de Productos</h1>
10   <nav>
```

Ahora vamos al archivo de **urls** de nuestro **proyecto** y agregamos la siguiente url para que nos tome todas las imágenes del proyecto

```
import settings
```

```
url(r'^media/(?P<path>.*)$', 'django.views.static.serve', {'document_root': settings.MEDIA_ROOT}),
```

Después guardamos todos los cambios y verificamos en el navegador



Luego agregamos nuestro archivo **estilo.css** a la carpeta **css** y editamos nuestro archivo **base.html** con el fin que todas nuestras plantillas quede con un estilo bonito

```
3 <head>
4   <title>{% block title %} Titulo Base {% endblock %}</title>
5   {% block css %}
6   <link rel="stylesheet" type="text/css" href="/media/css/estilo.css"/>
7   {% endblock %}
8 </head>
```

Al final guardamos los cambios y miramos el cambio en el navegador