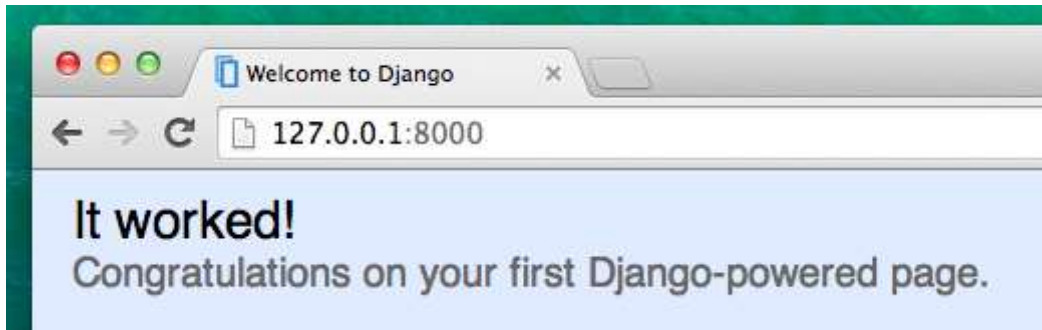


Práctica 9.1: Django urls

Vamos a construir nuestra primera página web – una página de inicio para nuestro blog. Pero primero, vamos a aprender un poco sobre URLs en Django.

¿Qué es una URL?

Una URL en este caso particular es simplemente una dirección web, puedes ver una URL cada vez que visitas cualquier sitio web - es visible en la barra de direcciones de tu navegador (Sí, `http://127.0.0.1:8000` es una URL. Y `http://uabc.mx` es también una URL).



Cada página en Internet necesita su propia URL. De esta manera tu aplicación sabe lo que debe mostrar a un usuario que abre una URL. En Django se usa algo llamado `URLconf` (configuración de URL), un conjunto de patrones que Django intentará hacer coincidir con la dirección URL recibida para encontrar la vista correcta.

¿Cómo funcionan las URLs en Django?

Vamos a abrir el archivo `mysite/urls.py` en tu editor de texto de preferencia (vim, nano, sublime) y ver cómo es:

```
"""mysite URL Configuration
[...]
"""

from django.urls import path, include
from django.contrib import admin
urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Como puedes ver, Django ya puso algo aquí para nosotros.

Las líneas que comienzan con triples comillas (`'''` o `"""`) se les llama *docstrings* – puedes escribirlas al inicio de un script, clase o método para describir que hace. Estas líneas no se ejecutaran en Python.

La URL de admin, que se visitó en una práctica anterior, ya está aquí:

```
path('admin/', admin.site.urls),
```

Esto significa que para cada URL que empiece con `admin/` Django encontrará su correspondiente *view*. En este caso estamos incluyendo en una sola línea muchas URLs de admin, así no está todo amontonado en este pequeño archivo - es más limpio y legible.

Tu primer URL de Django

Es hora de crear nuestro primer URL. Queremos que `'http://127.0.0.1:8000/` sea la página de inicio de nuestro blog y que muestre una lista de posts.

También queremos mantener el archivo `mysite/urls.py` limpio, así que importaremos URLs de nuestro `blog` al archivo principal `mysite/urls.py`.

Agrega una línea que importará `blog.urls`. Nota que estamos usando la función aquí `include` **así que deberás** agregar eso a la primera línea de `import` en el script.

Tu archivo `mysite/urls.py` debería verse algo así:

```
from django.urls import path, include
from django.contrib import admin
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
]
```

Django ahora redirigirá todo lo que vaya hacia `'http://127.0.0.1:8000/` a `blog.urls` y buscará por más instrucciones allí.

blog.urls

Crea un nuevo archivo vacío `blog/urls.py`. Agrega estas primeras dos líneas:

```
from django.urls import path
from . import views
```

Aquí solo estamos importando la función `path` de Django y todas nuestras `views` de la aplicación `blog` (todavía no tenemos ninguna, pero lo haremos en un momento).

Luego de esto, podemos agregar nuestro primer patrón URL:

```
urlpatterns = [
    path('', views.post_list, name='post_list'),
]
```

Como puedes ver, ahora estamos asignando una `view` llamada `post_list` al URL `^$`. Esta expresión regular hará juego con `^` (un inicio) seguido de `$` (un final) - por lo tanto, sólo una cadena vacía coincidirá. Y esto es correcto, ya que en los URL resolvers de Django `'http://127.0.0.1:8000/'` no es parte del URL. Este patrón mostrará a Django que `views.post_list` es el lugar correcto al que ir si alguien ingresa a tu sitio web con la dirección `'http://127.0.0.1:8000/'`.

La ultima parte, `name='post_list'`, es el nombre de la URL que se usara para identificar la vista. Este puede ser igual al nombre de la vista o puede ser algo completamente diferente. Nosotros estaremos usando los nombres de las URLs más adelante en el proyecto, así que es importante nombrar cada URL en la aplicación. También debemos de mantener los nombres de las URLs únicos y fáciles de recordar.

Si intentas visitar `http://127.0.0.1:8000/` en este momento, encontrarás un mensaje del tipo `'web page not available'`. Esto es porque el servidor (recordaste ejecutar `runserver` ?) no esta corriendo. Hecha un vistazo a la ventana de tu servidor para ver el porque.

```
File "/usr/lib/python3.5/importlib/_init_.py", line 126, in import_module
    return _bootstrap.gcd_import(name[level:], package, level)
File "<frozen importlib._bootstrap>", line 986, in _gcd_import
File "<frozen importlib._bootstrap>", line 969, in _find_and_load
File "<frozen importlib._bootstrap>", line 958, in _find_and_load_unlocked
File "<frozen importlib._bootstrap>", line 673, in _load_unlocked
File "<frozen importlib._bootstrap_external>", line 673, in exec_module
File "<frozen importlib._bootstrap>", line 222, in _call_with_frames_removed
File "/home/daw/django-daw/blog/urls.py", line 5, in <module>
    url(r'^$', views.post_list),
AttributeError: module 'blog.views' has no attribute 'post_list'
```

Tu consola muestra un error, pero no es para preocuparse - es de hecho muy util. En este caso esta diciendo que **no attribute 'post_list'**. Ese es el nombre de la *vista* que Django está

tratando de encontrar y usar, pero no la hemos creado todavía. En este punto tu `/admin/` tampoco funcionará. No pasa nada, esto lo solucionaremos en la siguiente parte.

Si deseas saber mas sobre URLconfs en Django, mira la documentación oficial [URLs](#).

Práctica 9.2 - Vistas de Django

Es hora de deshacerse del error que hemos creado en la parte anterior.

Un *view* es un lugar donde ponemos la “lógica” de nuestra aplicación. Se solicitará información del `model` que creaste anteriormente y se pasará a un `template` que crearás en la próxima práctica. Las vistas son sólo funciones de Python que son un poco más complicados que lo que hicimos en la práctica Introducción a Python.

Las vistas se colocan en el archivo `views.py`. Agregaremos nuestras vistas al archivo `blog/views.py`.

blog/views.py

Bien, vamos abrir este archivo y ver lo que contiene:

```
from django.shortcuts import render
..... ## Create your views here.
```

No demasiadas cosas aquí todavía.

Recuerda que las líneas que comienzan con `#` son comentarios - esto significa que esas líneas no serán ejecutadas por Python.

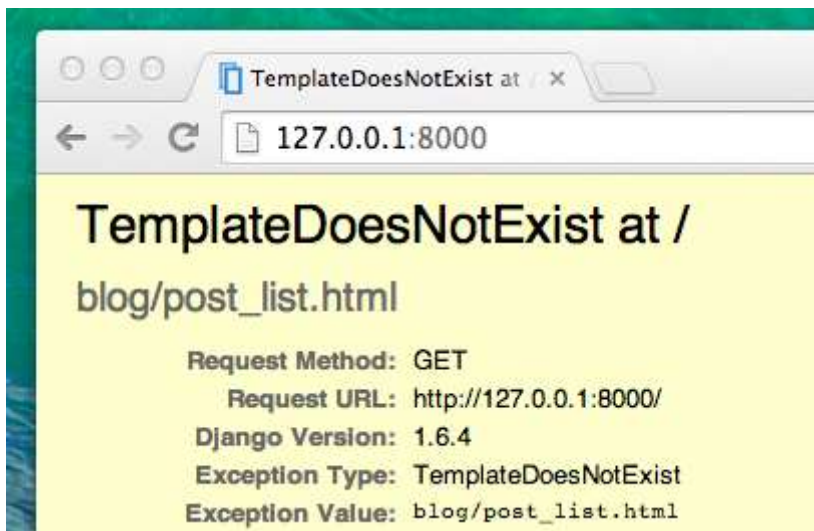
La vista más simple puede ser como esto:

```
def post_list(request):
    return render(request, 'blog/post_list.html', {})
```

Como puedes ver, hemos creado un método (`def`) llamado `post_list` que toma un `request` y hace un `return` de un método `render` que dibujará nuestra plantilla `blog/post_list.html`.

Guarda el archivo, dirígete a <http://127.0.0.1:8000/> y veamos lo que tenemos ahora.

¡Otro error! Leamos lo que está pasando ahora:



Esto indica que el servidor está corriendo de nuevo, al menos, pero todavía no se ve bien. Esta es una página de error creada por el servidor, nada de que asustarnos - justo como los errores en la consola estos de hecho nos pueden ser muy útiles. Puedes leer que el `TemplateDoesNotExist`. Corrijamos este error y creemos un *template* en la siguiente práctica.

Aprende más acerca de las views de Django leyendo la documentación oficial [Views](#)