Laboratorio IPC2: Clase 9

Introducción a Django

¿Qué es un framework?

Un framework es una especie de plantilla, un esquema conceptual, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar. Este concepto no existe únicamente en el área de informática.

Algunos tipos de framework que existen son:

- Para aplicaciones web (Angular, Vue JS, Flask)
- Para aplicaciones en general (.NET, Spring, Grails)
- Para gestión de contenido (CMS, Wordpress)

Framework Django

Documentación oficial de Django

¿Qué es Django?

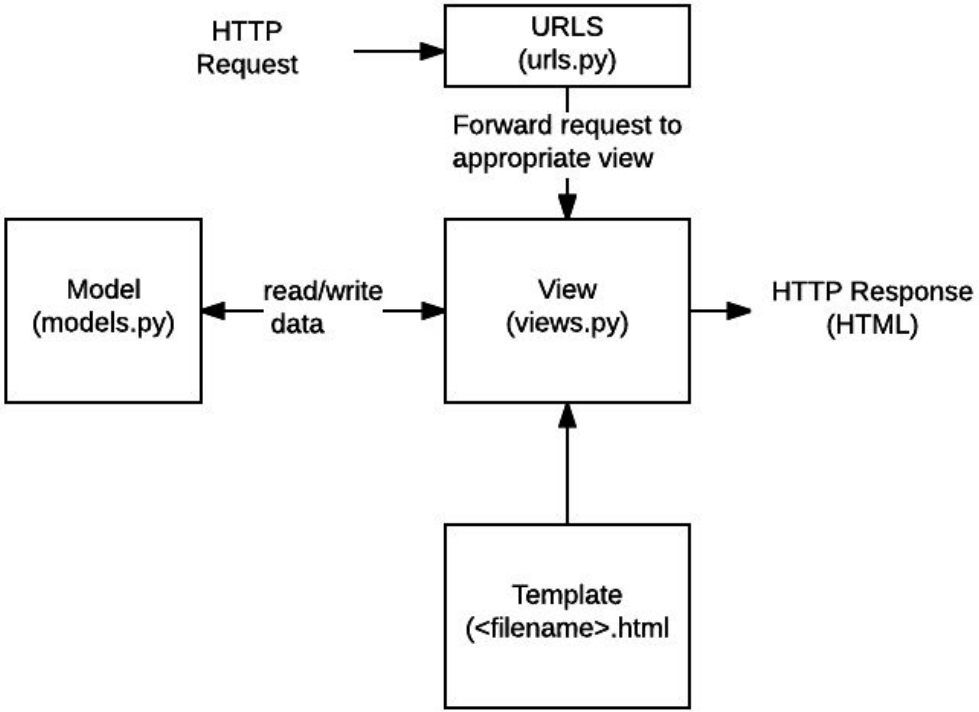
Un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles, con base al lenguaje de programación Python. Algunas de sus características:

- Gratuito y de código abierto.
- Completo
- Seguro
- Versátil
- Portable
- Escalable

Implementación del Código

Una aplicación web tradicional espera peticiones HTTP del explorador u otro cliente, y al recibir una elabora una tarea basado en el url y el tipo de petición y devuelve una respuesta, estos pasos son agrupados de cierta manera en el código.

Las aplicaciones web de Django normalmente agrupan el código en ficheros separados. Como se puede ver en la imagen a la derecha.



URLs

Aunque es posible procesar peticiones de cada URL individual vía una función individual, es mucho más sostenible escribir una función de visualización separada para cada recurso.

Se usa un mapeador URL para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición.

El mapeador URL se usa para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL puede también emparejar patrones de cadenas o dígitos específicos que aparecen en una URL y los pasan a la función de visualización como datos.

Views (vista)

Una vista es una función de gestión de peticiones que recibe peticiones HTTP y devuelve respuestas HTTP. Las vistas acceden a los datos que necesitan para satisfacer las peticiones por medio de modelos, y delegan el formateo de la respuesta a las plantillas ("templates").

Models (modelos)

Los modelos son objetos de Python que definen la estructura de los datos de una aplicación y proporcionan mecanismos para gestionar (añadir, modificar y borrar) y consultar registros en la base de datos.

Templates (plantillas)

Una plantilla (template) es un fichero de texto que define la estructura o diagrama de otro fichero (tal como una página HTML), con marcadores de posición que se utilizan para representar el contenido real.

Una vista puede crear dinámicamente una página usando una plantilla, rellenandola con datos de un modelo. Una plantilla se puede usar para definir la estructura de cualquier tipo de fichero; no tiene porqué ser HTML.

Django se refiere a este tipo de organización como arquitectura Modelo Vista Plantilla **"Model View Template (MVT)"**. Tiene muchas similaridades con la arquitectura más familiar Model View Controller.

¿Cómo utilizar Django para crear una aplicación web?

¿Cómo empezar?

Lo primero será asegurarse que tenemos python instalado en nuestro sistema, con el siguiente comando podremos ver la versión:

```
python -V
```

También es bueno asegurarse que se tiene instalado el sistema de gestión de paquetes de python pip o pip3.

```
pip -V o bien pip3 -v
```

¿Qué sigue?

Lo siguiente sería instalar el framework, para eso podemos hacerlo con el sencillo comando:

```
pip install django
```

Para corroborar la instalación del framework podemos ejecutar el comando anterior nuevamente o cualquiera de los siguientes:

```
pip list #despliega todos los paquetes instalados en el
sistema

pip show django #muestra la información de Django

python -m django --version #muestra la versión instalada
```

Lo siguiente, para crear nuestro primer proyecto en un directorio específico debemos ejecutar:

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

Esto creará una carpeta en el directorio actual con el nombre especificado. Y con el siguiente desglose:

```
misitio/
    manage.py
    misitio/
        __init__.py
        setting.py
        urls.py
        asgi.py
        wsgi.py
```

¿Qué significan los archivos?

- **misitio/** es el directorio ‘root’ del proyecto, el contenedor de todo el proyecto. Puede ser renombrado.
- **manage.py** es una utilidad a base de comandos que permite interactuar con el proyecto, para tareas administrativas .
- El directorio **interno misitio/** es el paquete Python del proyecto. Es el que se usa para importar cualquier archivo dentro del mismo, por ejemplo> misitio.settings.
- **misitio/__init__.py** es un archivo vacío que le dice a python que el directorio debe considerarse como un paquete.
- **misitio/settings.py** contiene toda la configuración del proyecto.
- **misitio/urls.py** declaración de URLs para el proyecto, tabla de contenidos del sitio.
- **misitio/asgi.py** un entry point para servidores web compatibles con ASGI.
- **misitio/wsgi.py** un entry point para servidores web compatibles con WSGI

Deployment

Lo siguiente será correr el proyecto para ver si está funcionando. Para esto debemos colocarnos en el directorio raíz o 'root'. Luego ejecutar el siguiente comando.

```
py manage.py runserver
```

Nos mostrará una pequeña advertencia sobre migraciones no aplicadas, en un principio podemos ignorarla. Pero aún así correrá la aplicación web, ahora debemos copiar la dirección que aparece en: *Starting development server at* **http://127.0.0.1:8000/**

Luego debemos ir a nuestro navegador web (Google Chrome, Microsoft Edge, Safari, Opera, etc) y escribir o pegar la dirección, y al ingresarla deberíamos poder ver la siguiente página.

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

Proyecto vs. App

Una app es una aplicación web que hace determinadas tareas. Un proyecto, por otro lado, es una colección de configuraciones y apps para un sitio web en particular. Un proyecto puede incluir múltiples apps. Una app puede estar en múltiples proyectos. Por lo tanto ahora crearemos nuestra primer app.

Ejecutamos el comando en nuestro directorio raíz('root'):

```
Py manage.py startapp <nombre>
```

```
myapp/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Con este nuevo directorio, podremos empezar a crear vistas. Y para poder utilizar las views debemos también crear un archivo de urls para nuestra app en el directorio de myapp/ y con la sintaxis ‘urls.py’.

```
from django.contrib import admin
from django.urls import include, path

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

Luego de crear nuestras url en el archivo de myapp/views.py, y de agregarlas al archivo myapp/urls.py, debemos ir a nuestro directorio raíz, y el directorio con el mismo nombre y al archivo urls (misitio/misitio/urls.py). Y agregamos los url de nuestra app como se muestra en el cuadro de arriba. Debemos asegurarnos de importar la función include, que permite hacer referencia a configuraciones de otros url.

Acceder a nuestras Views

Para acceder a las views de nuestra aplicación basta con ingresar la dirección de nuestro proyecto agregando la ruta de nuestra app:

http://127.0.0.1:8000/myapp/<nombre view>

Cabe mencionar que al momento de agregar un url en nuestro archivo de urls de nuestra app, no es necesario incluir en la ruta el slash al principio, ya que en nuestro urls.py de nuestro proyecto, ya le agregamos el slash.

Para más información consultar la documentación oficial.

FIN