INGENIERIA DE SOFTWARE

---------- Django ----------

Juan Scavuzzo

Desarrolladora:

Cisneros Valentina

INTRODUCION

En este informe se les va a contar la serie de fundamentos de Django.Es una guia para comenzar a aprender Django. El informe esta desarrollado en seis partes.Vamos a explorar todos los conceptos basicos con detalle, desde la instalacion, preparacion del entorno de desarrollo, modelos, vistas, plantillas, URLS y mas temas avanzados como migraciones y pruebas.

Cambios de apariencia Y tambien se veran los patrones de diseños que se identificaron a lo largo del proyecto.

➢ ¿Por qué Django?

Django es un framework web de alto nivel, escrito en Python que ayuda al desarrollo rápido y a un diseño limpio. Construido por desarrolladores experimentados, resuelve una buena parte de los problemas del desarrolloweb de tal manera que uno se pueda enfocar en escribir su app sin necesidad de reinventar la rueda. Es gratis y decódigo abierto.Sitio web: https://www.djangoproject.com. Usar un framework web, tal como Django, te permite desarrollar aplicaciones web seguras y confiables muy rapido de una forma estandarizada , sin tener que reinventar la rueda. Es definitivamente el mas completo, ofreciendo un amplio rango de caracteristicas para usar de forma rápida, tales como un servidor web autonomo para desarrollo y pruebas,motor de plantillas, procesamiento de formularios e interfaces con las herramientas de pruebas unitarias de Python. Django tambien ofrece aplicaciones construidas tales como sistemas de autenticacion, una interfaz administrativa, y muchas cosas mas.

➔ PROPOSITO

A través de este informe le contaremos cómo creamos una aplicación de encuestas básica.

Consistio de dos partes:

- ✔ Un sitio público que le permite a las personas ver sondeos y votar en ellos.
- ✔ Un sitio admin que le permite añadir, modificar y borrar sondeos.

Primero: ¿Cuál es la diferencia entre un proyecto y una aplicación?

➔ Proyectos vs. aplicaciones

Una app es una aplicación web que hace algo, por ejemplo, un sistema de blog, una base de datos de registros públicos o una aplicación de encuesta simple. Un proyecto es un conjunto de configuraciones y aplicaciones para un sitio web determinado. Un proyecto puede contener aplicaciones múltiples. Una aplicación puede estar en varios proyectos.

- ✗ Antes de comenzar lo primero que debemos hacer es instalar algunos programas en nuestra maquina para empezar a jugar con Django. La configuracion basica consiste en instalar Python,Virtualenv y Django
- ✗ Usar entornos virtuales no es obligatorio, pero es muy recomendado.Usando entornos virtuales, cada proyecto que desarroles tiene su ambiente aislado asi que las deependencias no haran conflicto.

¿COMENZAMOS!?

➢ Escribiendo La primera aplicación en Django, parte 1

En la terminal:

$ django-admin startproject mysite

veamos que creo

mysite/

    manage.py

    mysite/

__init__.py

settings.py

urls.py

wsgi.py

- manage.py: Un atajo para usar la utilidad de línea de comando django-admin. Es usado para ejecutar comandos de administración relacionados con nuestro proyecto. Lo usaremos para ejecutar el desarrollo en servidos, pruebas, migraciones y muchos más.

- init.py: Este archivo vacío le dice a Python que esta carpeta es un paquete.
- settings.py: Este archivo contiene la configuración de todo el proyecto. ¡Lo estaremos usando todo el tiempo!
- urls.py: Este archivo es el responsable de mapear las rutas y caminos (paths) en nuestro proyecto. Por ejemplo, si quieres mostrar algo en la URL /about/, tienes que mapearlo aquí primero.
- wsgi.py: Este archivo es simplemente una interfaz de puerta de enlace usada para despliegues. No debes preocuparte por él, déjalo así por ahora

➢ El servidor de desarrollo

Comprobemos que su proyecto Django funciona.

$ python manage.py runserver

Recarga automática del comando runserver

El servidor de desarrollo recarga de forma automática el código Python para cada petición cuando sea necesario. No es necesario reiniciar el servidor para que los cambios de código surtan efecto. Sin embargo, algunas acciones como la adición de archivos no provoca un reinicio, por lo que tendrá que reiniciar el servidor en estos casos.

- ✗ Verá la siguiente salida en la línea de comandos:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until
they are applied.
Run 'python manage.py migrate' to apply them.

octubre 16, 2019 - 15:50:53
Django version 2.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

➢ Creando la aplicación encuestas

$ python manage.py startapp polls

Eso va a crear un directorio polls que se presenta de la siguiente forma:

polls/

__init__.py

admin.py

apps.py

migrations/

__init__.py

models.py

tests.py

views.py

Esta estructura de directorios almacenará la aplicación encuesta.

Algunas de estas aplicaciones utilizan al menos una tabla de base de datos, por lo que necesitamos crear las tablas en la base de datos antes de poder utilizarlas.

$ python manage.py migrate

El comando migrate analiza la configuración INSTALLED_APPS y crea las tablas de base de datos necesarias según la configuración de base de datos en su archivo mysite/settings.py y las migraciones de base de datos distribuidas con la aplicación (trataremos este tema más tarde). Verá un mensaje para cada migración que aplique. Si está interesado, ejecute el cliente de línea de comandos para su base de datos y teclee \dt (PostgreSQL), SHOW TABLES; (MySQL), .schema (SQLite), o SELECT TABLE_NAME FROM USER_TABLES; (Oracle) para desplegar las tablas que creó Django.

➢ Creando modelos

El modelo define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato debe ser almacenado se encuentra en una variable con ciertos parametros posee metodo tambien.Todo esto perimte indicar y controlar el comportamiento de los datos

En nuestra sencilla aplicación encuesta, vamos a crear dos modelos: Question y Choice . Una Question tiene una pregunta y una fecha de publicación. Una Choice tiene dos campos: el texto de la elección y un conteo de votos. Cada Choice se asocia a una

```
$ python manage.py makemigrations polls
```

✗ Usted debe ver algo similar a lo siguiente:

```
Migrations for 'polls':
  polls/migrations/0001_initial.py:
    - Create model Choice
    - Create model Question
    - Add field question to choice
```

Question

Al ejecutar makemigrations, usted le indica a Django que ha realizado algunos cambios a sus modelos (en este caso, ha

realizado cambios nuevos) y que le gustaría que los guarde como una *migración*.

```
$ python manage.py migrate
```

✗ Usted debe ver algo similar a lo siguiente:

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Rendering model states... DONE
  Applying polls.0001_initial... OK
```

El comando migrate toma todas las migraciones que no han sido aplicadas (Django rastrea cuales se aplican utilizando una tabla especial en su base de datos llamada django_migrations) y las ejecuta contra su base de datos; básicamente, sincronizando los cambios que usted ha realizado a sus modelos con el esquema en la base de datos.

Las migraciones son muy potentes y le permiten modificar sus modelos con el tiempo, a medida que desarrolla su proyecto, sin necesidad de eliminar su base de datos o las tablas y hacer otras nuevas. Este se especializa en la actualización de su base de datos en vivo, sin perder datos. Vamos a hablar de ellas en mayor profundidad más tarde en el tutorial, pero por ahora, recuerde la guía de tres pasos para hacer cambios de modelo:

- Cambie sus modelos (en models.py).
- Ejecute el comando python manage.py makemigrations para crear migraciones para esos cambios
- Ejecute el comando python manage.py migrate para aplicar esos cambios a la base de datos.

➢ Vistas

La vista se presenta en forma de funciones en Python, su proposito es determinar que datos seran visualizados, se encarga de tareas conocidas como la validacion de datos a traves de formularios. Lo mas importante a entender respecto a la vista es que no tiene nada que ver con el estilo de presentacion de los datos, solo se encarga de los datos , la presentacion es tarea de la plantilla

➢ Url’s

De esta manera tu aplicación sabe lo que debe mostrar a un usuario que abre una URL. En Django utilizamos algo que se llama URLconf (configuración de URL). URLconf es un conjunto de patrones que Django intentará comparar con la URL recibida para encontrar la vista correcta.

➢ Plantillas

Otra cosa buena que tiene Django es la extension de plantillas. Significa que se puede reusar partes del HTML para diferentes paginas del sitio web. Las plantillas son utiles cuando quieres utilizar la misma informacion o el mismo diseño en mas de un lugar.

➢ Sitio administrativo de Django

La generación de sitios administrativos para su personal o clientes para agregar, modificar y eliminar contenido es un trabajo aburrido que no requiere mucha creatividad. Por esa razón, Django automatiza completamente la creación de interfaces de sitios administrativos para los modelos.

Primero tendremos que crear un usuario que pueda iniciar sesión en el sitio administrativo. Ejecute el siguiente comando:

```
$ python manage.py createsuperuser

Introduzca su nombre de usuario deseado y pulse enter.

Username: admin

A continuación se le solicitará su dirección de correo electrónico deseada:

Email address: ejemplo@ejemplo.com

El paso final es introducir su contraseña. Se le pedirá que introduzca su contraseña dos veces, la segunda vez como confirmación de la primera.

Password: **********
Password (again): *********
Superuser created successfully.
```

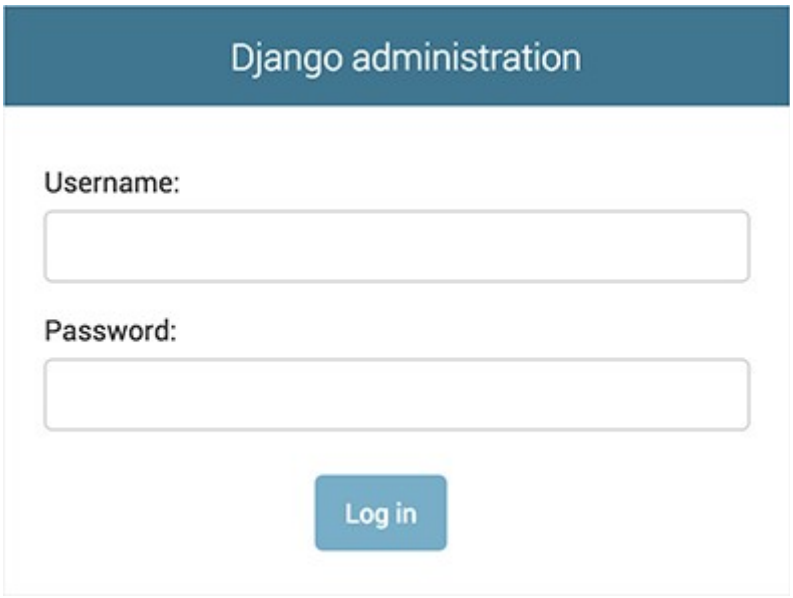
➢ Inicie el servidor de desarrollo

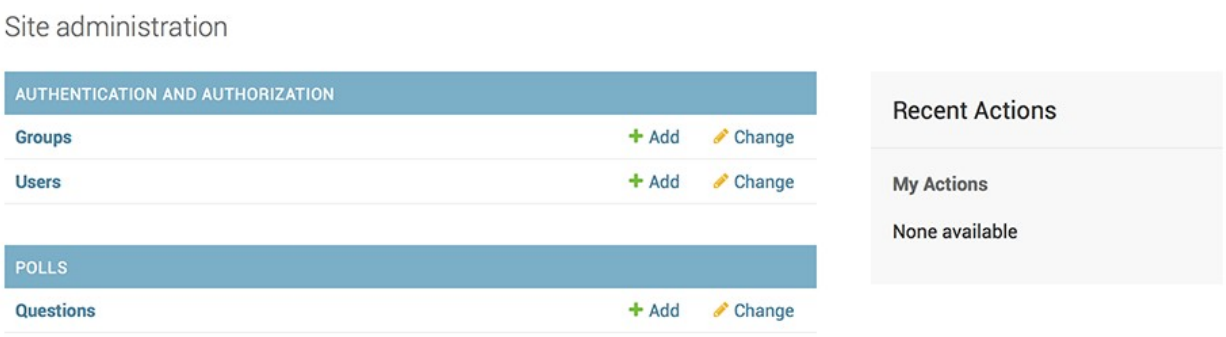
El sitio administrativo de Django se activa de forma predeterminada. Vamos a iniciar el servidor de desarrollo y a explorarlo.

Si el servidor no está en marcha, inícielo de la siguiente forma:

$ python manage.py runserver

A continuación, abra un navegador Web y vaya a «/admin/» en su dominio local, por ejemplo, http://127.0.0.1:8000/admin/. Usted debe ver la pantalla de inicio de sesión del sitio administrativo:


Django administration
Username:
Password:
Log in


Site administration
AUTHENTICATION AND AUTHORIZATION
Groups
Add
Change
Users
Add
Change
POLLS
Questions
Add
Change
Recent Actions
My Actions
None available

➢ Usando el sistema de plantillas

La plantilla basicamente es una pagina HTML con algunas etiquetas extras propias de Django.No solamente crea contenido en HTML (tambien css)

configuración de TEMPLATES de su proyecto describe cómo Django cargará y creará las plantillas. El archivo de configuración predeterminada configura un backend DjangoTemplates cuya opción APP_DIRS está configurada como True. Convencionalmente, DjangoTemplates busca un subdirectorio «templates» en cada una de las INSTALLED_APPS.

➢ Cabiamos a vistas genéricas: Menos código es mejor

¿Por qué el intercambio de código?

Generalmente, cuando se escribe una aplicación de Django, usted podrá evaluar si las vistas genéricas son una buena opción para su problema y las utilizará desde el principio en lugar de reestructurar su código a medio camino. Sin embargo, este tutorial intencionadamente se ha centrado hasta ahora en escribir las

vistas «de la manera difícil» para centrarse en los conceptos esenciales.

Debe saber matemáticas básicas antes de comenzar a utilizar una calculadora.

➢ Pruebas

El por qué necesita crear pruebas:

- ✗ Las pruebas no solo identifican los problemas, los previenen

Es un error pensar en las pruebas sólo como un aspecto negativo del desarrollo.

Sin las pruebas, el propósito o el comportamiento previsto de una aplicación podría ser bastante incomprensible. Aún cuando es su propio código, usted a veces se encontrará buscando en él tratando de averiguar qué es exactamente lo que hace.

Las pruebas cambian eso, ellas iluminan su código desde adentro, y cuando algo sale mal, ellas se centran en la parte que salió mal; *incluso si usted no se ha dado cuenta de que salió mal*

➢ Personalizamos la apariencia de su aplicación

En primer lugar, cree un directorio llamado static en su directorio polls. Django buscará archivos estáticos allí, del mismo modo cómo Django encuentra las plantillas dentro de polls/templates/.

Quedaria algo como esto:

style.css

```css
li {
    color: black;
    font-family: Lobster;
}

h1 {
    text-align: center ;
}

body {
    text-align: left ;
    font: 150% Times New Roman, sans-serif;
    line-height:1 ; /**establece la distancia entre lineas de texto**/
    color: black;
    background-color: #FEECCF;
    margin: 0;
    padding: 0;
}
```

➢ Cambios Apartes de los pasos del tutorial:

-Solo apariencia. En el archivo css.

➢ Patrones de diseño

- Los patrones de diseño son la base para la busqueda de soluciones a problemas comunes en el desarrollo del softaware
- Un patron de diseño resulta ser una solucion a un problema de diseño

PATRONES IDENTIFICADOS EN EL PROYECTO

- *Patrones de diseño de Creacion*

Los patrones de creacion se centran en cómo crear una instancia de un objeto o grupo de objetos relacionados.

- ✔ Factory Method: proporciona una forma de delegar la logica de creacion de instancias a clases secundarias

  -En django

- ✔ Singleton: Asegura que solo se crea un objeto de una clase en particular

-En django algunos archivos instancian al None

- ✔ Abstract Factory: Una fabrica de fabricas; una fabrica que agrupa a las fabricas individuales pero relacionadas/dependientes juntas sin especificar sus clases concretas.

  -En django el administrador pueder crear preguntas y respuestas. Puesto que solo el puedo hacerlo logeandose

- *Patrones de comportamiento*

Son patrones de diseño que identifican patrones de comunicación comunes entre objetos y realizan estos patrones. Al hacerlo, estos patrones aumentan la flexibilidad al llevar a cabo esta comunicación.

- ✔ Chain of responsibility: Ayuda a construir una cadena de objetos. La solicitud ingresa desde un extremo y continua yendo de un objeto a otro hasta encontrar el conrolador adecuado

  -En Django las clases siguen una serie de pasos en base a lo solicitado por el usuario

- ✔ Iterator: Presenta un forma de acceder a los elementos de un objeto sin exponer la presentacion subyacente

-Django nos presenta una forma sencilla de votar en una plicacion encuesta

- ✔ Mediator: El patron del mediador agrega un objeto de terceros (llamado mediador) para controlar la interaccion entre dos objetos (llamados colegas). Ayuda a reducir el acoplamiento entre las clases que se comunican entre si. Porque ahora no necesita tener el conocimiento de la implementacion del otro.

  - La interfaz de la pagina seria el mediador entre el usuario y el sistema

- ✔ Observer: Define una dependencia entre objetos paara que cada vez que cambie su estado, se notifiquen todos sus dependientes.

  -Cada vez que se selecciona una respuesta, se suma uno a los votos de las mismas

- ✔ Template Method: El metodo plantilla define el esqueleto del programa de un algoritmo de una operación, defiriendo algunos pasos a subclases. Le permite redifinir ciertos pasos de un algoritmo sin cambiar la estructura del algoritmo.

-Usamos sistema de plantillas para organizar como se mostraran los datos.

- Anexos

link trello : https://trello.com/b/kzAS7ltW/tutorial-django

link del repositorio de Github: https://github.com/valentina3101/TutorialDjango