Vistas, URLs, y Templates en Django

Django MTV

Elizabeth León Guzman, Ph.D.
eleong@unal.edu.co

Carlos Andrés Sierra, M.Sc.
casierrav@unal.edu.co

Joseph Gallego, M.Sc.
jagallegom@unal.edu.co

Jhon López, M.Sc.(C)
jalopezfa@unal.edu.co

Grupo de Investigación en Minería de Datos – (MIDAS)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

Agenda

Archivos static & templates

Django maneja dos nociones de archivos distintos a los de tipo Python. Por facilidad de manejo, para cada noción se busca una carpeta de almacenamiento diferente y se manejan rutas relativas. Estas nociones son:

- `Template`: Parte del patrón de desarrollo, Corresponde a los archivos *HTML*.
- `Static`: Corresponde a archivos como imágenes, estilo (CSS), dinámicas con javascript, entre otros, que son usados por los *templates*.

Las rutas relativas se toman basandose en la carpeta de cada elemento como la raíz de la ruta.

Relación de las vistas

Para relacionar las vistas en la aplicación, se hace uso del archivo `views.py`. Para cada vista a relacionar, se debe crear una función que tiene como entrada un *request* y como salida el renderizado de un *template*.

A continuación se muestra un código que ejemplifica dicha definición:

```
from django.shortcuts import render

def nombre_funcion(request):
.    return render(request, 'aplicacion/archivo.html')
```

Agenda

Mapeo de la aplicación

Para cada una de las vistas relacionadas en la aplicación, se debe realizar un mapeo que relacione esta vista con una URL específica única.
Para ello, en cada aplicación se recomienda crear un archivo `urls.py`, en donde debe existir una lista llamada `urlpatterns`, y cada URL será un elemento independiente de esta lista.
Este sería un ejemplo del contenido del archivo *urls.py*:

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

urlpatterns = [
.   path('url_path', views.nombre_funcion,
name="vista_nombre"),
.   ..., ]
```

Mapeo del proyecto

Recuerde que usted puede tener tantas aplicaciones como quiera en el proyecto de **Django**, y en cada una tendrá un mapeo particular. Estos mapeos deben ser relacionados en el archivo de `urls.py` que se crea por defecto en el proyecto global en donde está trabajando. En este archivo ya existe una lista de URLs mapeadas a nivel de proyecto, y se debe agregar a esta lista cada uno de los archivos de mapeo de las aplicaciones, como se muestra a continuacuón:

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

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

Agenda

Definición

Toda aplicación web está casi obligada a usar de alguma manera formularios. Django proporciona una manera sencilla de trabajar con formularios, pero se le recomienda que en cada aplicación que construya, cree un archivo llamado `forms.py` donde guarde todas las definiciones de sus formularios. Un ejemplo del código posible en este archivo se muestra a continuación:

```
from django import forms

class nombre_formulario(Form):
.    opciones = (...)
.    desplegable = ChoiceField(label = "nombre_1",
required=True, choices=opciones)
.    texto = CharField(label = "nombre_2", required=False)
```

Campos comunes

A continuación se enlistan los tipos de campos comunes que puede ser utilizados en un formulario de Django:

- BooleanField
- CharField
- ChoiceField
- DateField
- DateTimeField
- DecimalField
- EmailField
- FileField
- ImageField
- JSONField
- MultipleChoiceField
- URLField

Agenda

Ajuste de vínculos entre Templates

A nivel de los *templates* es normal que existan vínculos que permiten la navegación entre las distintas pantallas de la aplicación. Para hacer correctamente estos vínculos, se hace uso de la URL proporcionada en el archivo de mapeo definido previamente.
Siendo esto así, el siguiente sería un segmento válido de código *HTML* para este propósito:

```
<a href="/aplicacion/url_path/">
```

Tranferencia desde las Vistas

Una tarea común a hacer en Django es el envío de información desde la *vista* al *template* como parte del proceso natural de respuesta de una petición.
Esto se resuelve de una manera sencilla, ya que la función de *render* tiene como tercer posible argumento la recepción de un diccionario con toda la información que se quiera enviar al *template*, como se muestra a continuación:

```
def nombre_funcion(request):
.    parametros={'clave_1':'valor_1',..,'clave_n':'valor_n'}
.    render(request, 'aplicacion/archivo.html', parametros)
```

Inyección de contenido

Desde el punto de vista del *template*, cuando se recibe información, existe una forma de agregarla al código HTML usando una sintaxis parecida a la típica de Python, pero con par de cambios: (i) el uso de {{}} para imprimir variables, y (ii) el uso de {%%} para colocar elementos como condicionales o ciclos.

```
{% block content %}
.    {% for elemento in lista %}
.        {{ elemento.campo_1 }}
.        {{ elemento.campo_2 }}
.        ...
.        {{ elemento.campo_n }}
.    {% endfor %}
{% endblock % }
```

Inyección de formulario

En cuanto a los formularios, la forma más simple de agregarlos a un template se muestra a continuación.:

```
{% block content %}
<form method="POST">{% csrf_token %}
.    {{ formulario.as_p }}
.    <input type="submit" value="Texto de Botón" id="btn"
name="btn">
</form>
{% endblock %}
```

Esta forma solo tiene un inconveniente: el formato de presentación lo coloca por defecto Django, y rara vez será como usted lo desea observar.

Gracias!

¿Preguntas?