### La Página de Temas

Ahora que tenemos un enfoque eficiente para construir páginas, podemos centrarnos en nuestras dos siguientes páginas: la página de temas generales y la página para mostrar las entradas de un único tema. La página de temas mostrará todos los temas que han creado los usuarios, y es la primera página que implicará trabajar con datos.

#### El patrón URL de los temas

En primer lugar, definimos la URL de la página de temas. Es habitual elegir un fragmento de URL simple que refleje el tipo de información que se presenta en la página. Utilizaremos la palabra *topics*, por lo que la URL *http://localhost:8000/topics/* devolverá esta página. Así es como modificamos *learning\_logs/urls.py*:

**learning\_logs/urls.py**

"""Defines URL patterns for learning\_logs."""  
--snip--  
urlpatterns = [  
 # Home page  
 path('', views.index, name='index'),  
 # Page that shows all topics.  
 path('topics/', views.topics, name='topics'),  
]

El nuevo patrón de URL es la palabra *topics*, seguida de una barra oblicua. Cuando Django examine una URL solicitada, este patrón coincidirá con cualquier URL que tenga la URL base seguida de *topics*. Puedes incluir u omitir una barra diagonal al final, pero no puede haber nada más después de la palabra *topics*, o el patrón no coincidirá. Cualquier solicitud con una URL que coincida con este patrón se pasará a la función topics() en *views.py*.

#### La vista Temas

La función topics() necesita recuperar algunos datos de la base de datos y enviarlos a la plantilla. Añade lo siguiente a *views.py*:

**views.py**

from django.shortcuts import render  
  
❶ from .models import Topic  
  
def index(request):  
 --snip--  
  
❷ def topics(request):  
 """Show all topics."""  
❸ topics = Topic.objects.order\_by('date\_added')  
❹ context = {'topics': topics}  
❺ return render(request, 'learning\_logs/topics.html', context)

Primero importamos el modelo asociado a los datos que necesitamos ❶. La función topics() necesita un parámetro: el objeto request que Django ha recibido del servidor ❷. Consultamos la base de datos pidiendo los objetos Topic, ordenados por el atributo date\_added ❸. Asignamos el conjunto de consultas resultante a topics.

A continuación, definimos un contexto que enviaremos a la plantilla ❹. Un *context* es un diccionario en el que las claves son nombres que utilizaremos en la plantilla para acceder a los datos que queremos, y los valores son los datos que necesitamos enviar a la plantilla. En este caso, hay un par clave-valor, que contiene el conjunto de temas que mostraremos en la página. Cuando construimos una página que utiliza datos, llamamos a render() con el objeto request, la plantilla que queremos utilizar y el diccionario context ❺.

#### La plantilla Temas

La plantilla para la página de temas recibe el diccionario context, para que la plantilla pueda utilizar los datos que proporciona topics(). Crea un archivo llamado *topics.html* en el mismo directorio que *index.html*. Así es como podemos mostrar los temas en la plantilla:

**topics.html**

{% extends 'learning\_logs/base.html' %}  
  
{% block content %}  
  
 <p>Topics</p>  
  
❶ <ul>  
❷ {% for topic in topics %}  
❸ <li>{{ topic.text }}</li>  
❹ {% empty %}  
 <li>No topics have been added yet.</li>  
❺ {% endfor %}  
❻ </ul>  
  
{% endblock content %}

Utilizamos la etiqueta {% extends %} para heredar de *base.html*, igual que hicimos en la página de inicio, y luego abrimos un bloque content. El cuerpo de esta página contiene una lista con viñetas de los temas que se han introducido. En HTML estándar, una lista con viñetas se denomina *unordered list* y se indica mediante las etiquetas </ul>. La etiqueta de apertura inicia la lista con viñetas de temas ❶.

A continuación utilizamos una etiqueta de plantilla que equivale a un bucle for, que recorre la lista topics desde el diccionario context ❷. El código utilizado en las plantillas difiere del de Python en algunos aspectos importantes. Python utiliza la sangría para indicar qué líneas de una sentencia for forman parte de un bucle. En una plantilla, cada bucle for necesita una etiqueta {% endfor %} explícita que indique dónde se produce el final del bucle. Por tanto, en una plantilla, verás bucles escritos así:

{% for item in list %}  
 do something with each item  
{% endfor %}

Dentro del bucle, queremos convertir cada tema en un elemento de la lista con viñetas. Para imprimir una variable en una plantilla, encierra el nombre de la variable entre llaves dobles. Las llaves no aparecerán en la página; sólo indican a Django que estamos utilizando una variable de plantilla. Así, el código {{ topic.text }} ❸ será sustituido por el valor del atributo text del tema actual en cada pasada por el bucle. La etiqueta HTML </li> indica un *list item*. Cualquier cosa entre estas etiquetas, dentro de un par de etiquetas </ul>, aparecerá como un elemento con viñetas en la lista.

También utilizamos la etiqueta de plantilla {% empty %} ❹, que indica a Django qué hacer si no hay elementos en la lista. En este caso, imprimimos un mensaje informando al usuario de que aún no se ha añadido ningún tema. Las dos últimas líneas cierran el bucle for ❺ y luego cierran la lista con viñetas ❻.

Ahora tenemos que modificar la plantilla base para incluir un enlace a la página de temas. Añade el siguiente código a *base.html*:

**base.html**

<p>  
❶ <a href="{% url 'learning\_logs:index' %}">Learning Log</a> -  
❷ <a href="{% url 'learning\_logs:topics' %}">Topics</a>  
</p>  
  
{% block content %}{% endblock content %}

Añadimos un guión después del enlace a la página de inicio ❶, y luego añadimos de nuevo un enlace a la página de temas utilizando la etiqueta de plantilla {% url %} ❷. Esta línea indica a Django que genere un enlace que coincida con el patrón de URL con el nombre 'topics' en *learning\_logs/urls.py*.

Ahora, cuando actualices la página de inicio en tu navegador, verás un enlace Temas. Cuando hagas clic en el enlace, verás una página similar a la de [la Figura 18-4](#figure18-4).

|  |
| --- |
| Figura |



18-4: Página

de temas

[anterior](c18_35.html)[Subtema 36 de 41: (Ver todo)](c18.html)[siguiente](c18_37.html)