### Añadir nuevos temas

Ahora que el usuario puede añadir un nuevo tema, también querrá añadir nuevas entradas. Definiremos de nuevo una URL, escribiremos una función de vista y una plantilla, y enlazaremos con la página. Pero antes, añadiremos otra clase a *forms.py*.

#### El modelo de formulario de entrada

Necesitamos crear un formulario asociado al modelo Entry, pero esta vez, con un poco más de personalización que TopicForm:

**forms.py**

from django import forms  
  
from .models import Topic, Entry  
  
class TopicForm(forms.ModelForm):  
 --snip--  
  
class EntryForm(forms.ModelForm):  
 class Meta:  
 model = Entry  
 fields = ['text']  
❶ labels = {'text': ''}  
❷ widgets = {'text': forms.Textarea(attrs={'cols': 80})}

Actualizamos la declaración import para incluir tanto Entry como Topic. Creamos una nueva clase llamada EntryForm que hereda de forms.ModelForm. La clase EntryForm tiene una clase anidada Meta que enumera el modelo en el que se basa y el campo a incluir en el formulario. Volvemos a dar al campo 'text' una etiqueta en blanco ❶.

Para EntryForm, incluimos el atributo widgets ❷. Un *widget* es un elemento de formulario HTML, como un cuadro de texto de una línea, un área de texto multilínea o una lista desplegable. Al incluir el atributo widgets, puedes anular las opciones de widget predeterminadas de Django . Aquí le estamos diciendo a Django que utilice un elemento forms.Textarea con una anchura de 80 columnas, en lugar de las 40 columnas por defecto. Esto da a los usuarios espacio suficiente para escribir una entrada con sentido.

#### La URL de la nueva\_entrada

Las nuevas entradas deben asociarse a un tema concreto, por lo que necesitamos incluir un argumento topic\_id en la URL para añadir una nueva entrada. Aquí tienes la URL, que debes añadir a *learning\_logs/urls.py*:

**learning\_logs/urls.py**

--snip--  
urlpatterns = [  
 --snip--  
 # Page for adding a new entry.  
 path('new\_entry/<int:topic\_id>/', views.new\_entry, name='new\_entry'),  
]

Este patrón de URL coincide con cualquier URL de la forma *http://localhost:8000/new\_entry/id/*, donde id es un número que coincide con el ID del tema. El código <int:topic\_id> captura un valor numérico y lo asigna a la variable topic\_id. Cuando se solicita una URL que coincide con este patrón, Django envía la solicitud y el ID del tema a la función de vista new\_entry().

#### La función de vista new\_entry()

La función de vista para new\_entry es muy parecida a la función para añadir un nuevo tema. Añade el siguiente código a tu archivo *views.py*:

**views.py**

from django.shortcuts import render, redirect  
  
from .models import Topic  
from .forms import TopicForm, EntryForm  
  
--snip--  
def new\_entry(request, topic\_id):  
 """Add a new entry for a particular topic."""  
❶ topic = Topic.objects.get(id=topic\_id)  
  
❷ if request.method != 'POST':  
 # No data submitted; create a blank form.  
❸ form = EntryForm()  
 else:  
 # POST data submitted; process data.  
❹ form = EntryForm(data=request.POST)  
 if form.is\_valid():  
❺ new\_entry = form.save(commit=False)  
❻ new\_entry.topic = topic  
 new\_entry.save()  
❼ return redirect('learning\_logs:topic', topic\_id=topic\_id)  
  
 # Display a blank or invalid form.  
 context = {'topic': topic, 'form': form}  
 return render(request, 'learning\_logs/new\_entry.html', context)

Actualizamos la declaración import para incluir el EntryForm que acabamos de hacer. La definición de new\_entry() tiene un parámetro topic\_id para almacenar el valor que recibe de la URL. Necesitaremos el tema para renderizar la página y procesar los datos del formulario, así que utilizamos topic\_id para obtener el objeto tema ❶ correcto .

A continuación, comprobamos si el método de solicitud es POST o GET ❷. El bloque if se ejecuta si es una petición GET, y creamos una instancia vacía de EntryForm ❸.

Si el método de solicitud es POST, procesamos los datos creando una instancia de EntryForm, rellenada con los datos POST del objeto request ❹. A continuación, comprobamos si el formulario es válido. Si lo es, tenemos que establecer el atributo topic del objeto de entrada antes de guardarlo en la base de datos. Cuando llamamos a save(), incluimos el argumento commit=False ❺ para decirle a Django que cree un nuevo objeto de entrada y lo asigne a new\_entry, sin guardarlo aún en la base de datos. Establecemos el atributo topic de new\_entry al tema que sacamos de la base de datos al principio de la función ❻. A continuación, llamamos a save() sin argumentos, guardando la entrada en la base de datos con el tema asociado correcto.

La llamada a redirect() requiere dos argumentos: el nombre de la vista a la que queremos redirigir y el argumento que requiere la función vista ❼. Aquí, estamos redirigiendo a topic(), que necesita el argumento topic\_id. Esta vista muestra la página del tema para el que el usuario hizo una entrada, y debería ver su nueva entrada en la lista de entradas.

Al final de la función, creamos un diccionario context y mostramos la página utilizando la plantilla *new\_entry.html*. Este código se ejecutará para un formulario en blanco, o para un formulario que ha sido enviado pero resulta ser inválido.

#### La plantilla new\_entry

Como puedes ver en el código siguiente, la plantilla para new\_entry es similar a la plantilla para new\_topic:

**new\_entry.html**

{% extends "learning\_logs/base.html" %}  
  
{% block content %}  
  
❶ <p><a href="{% url 'learning\_logs:topic' topic.id %}">{{ topic }}</a></p>  
  
 <p>Add a new entry:</p>  
❷ <form action="{% url 'learning\_logs:new\_entry' topic.id %}" method='post'>  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name='submit'>Add entry</button>  
 </form>  
  
{% endblock content %}

Mostramos el tema en la parte superior de la página ❶, para que el usuario pueda ver a qué tema está añadiendo una entrada. El tema también actúa como enlace a la página principal de ese tema.

El argumento action del formulario incluye el valor topic.id en la URL, para que la función de vista pueda asociar la nueva entrada con el tema ❷ correcto . Aparte de eso, esta plantilla tiene el mismo aspecto que *new\_topic.html*.

#### Enlace a la página nueva\_entrada

A continuación, tenemos que incluir un enlace a la página new\_entry desde cada página de tema, en la plantilla de tema:

**topic.html**

{% extends "learning\_logs/base.html" %}  
  
{% block content %}  
  
 <p>Topic: {{ topic }}</p>  
  
 <p>Entries:</p>  
 <p>  
 <a href="{% url 'learning\_logs:new\_entry' topic.id %}">Add new entry</a>  
 </p>  
  
 <ul>  
 --snip--  
 </ul>  
  
{% endblock content %}

Colocamos el enlace para añadir entradas justo antes de mostrar las entradas, porque añadir una nueva entrada será la acción más común en esta página. [La Figura 19-2](#figure19-2) muestra la página new\_entry. Ahora los usuarios pueden añadir nuevos temas y tantas entradas como quieran para cada tema. Prueba la página new\_entry añadiendo algunas entradas a algunos de los temas que has creado.

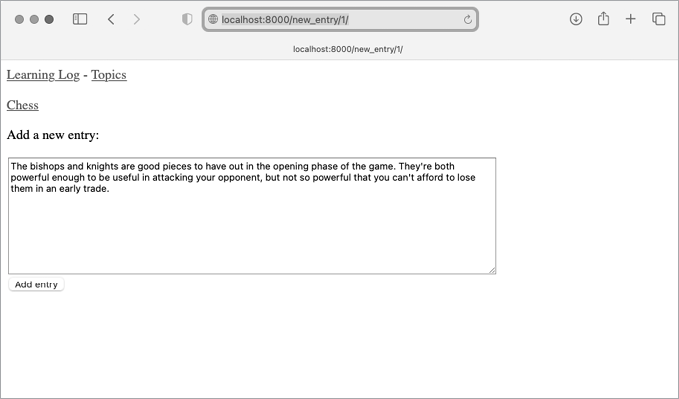


Figura 19-2: La

página



new\_entry

[anterior](c19_3.html)[Subtema 4 de 23: (Ver todo)](c19.html)[siguiente](c19_5.html)