### Editar entradas

Ahora crearemos una página para que los usuarios puedan editar las entradas que han añadido.

#### La URL editar\_entrada

La URL de la página tiene que pasar el ID de la entrada que se va a editar. Aquí tienes *learning\_logs/urls.py*:

**urls.py**

--snip--  
urlpatterns = [  
 --snip--  
 # Page for editing an entry.  
 path('edit\_entry/<int:entry\_id>/', views.edit\_entry, name='edit\_entry'),  
]

Este patrón de URL coincide con URL como *http://localhost:8000/edit\_entry/id/*. Aquí el valor de id se asigna al parámetro entry\_id. Django envía las solicitudes que coinciden con este formato a la función de vista edit\_entry().

#### La función de vista edit\_entry()

Cuando la página edit\_entry recibe una solicitud GET, la función edit\_entry() devuelve un formulario para editar la entrada. Cuando la página recibe una petición POST con el texto de la entrada revisado, guarda el texto modificado en la base de datos:

**views.py**

from django.shortcuts import render, redirect  
  
from .models import Topic, Entry  
from .forms import TopicForm, EntryForm  
--snip--  
  
def edit\_entry(request, entry\_id):  
 """Edit an existing entry."""  
❶ entry = Entry.objects.get(id=entry\_id)  
 topic = entry.topic  
  
 if request.method != 'POST':  
 # Initial request; pre-fill form with the current entry.  
❷ form = EntryForm(instance=entry)  
 else:  
 # POST data submitted; process data.  
❸ form = EntryForm(instance=entry, data=request.POST)  
 if form.is\_valid():  
❹ form.save()  
❺ return redirect('learning\_logs:topic', topic\_id=topic.id)  
  
 context = {'entry': entry, 'topic': topic, 'form': form}  
 return render(request, 'learning\_logs/edit\_entry.html', context)

Primero importamos el modelo Entry. A continuación, obtenemos el objeto entrada que el usuario quiere editar ❶ y el tema asociado a esta entrada. En el bloque if  , que se ejecuta para una solicitud GET, creamos una instancia de EntryForm con el argumento instance=entry ❷. Este argumento indica a Django que cree el formulario, rellenado previamente con la información del objeto de entrada existente. El usuario verá sus datos existentes y podrá editarlos.

Al procesar una solicitud POST, pasamos los argumentos instance=entry y data=request.POST ❸. Estos argumentos indican a Django que cree una instancia de formulario basada en la información asociada al objeto de entrada existente, actualizada con cualquier dato relevante de request.POST. A continuación, comprobamos si el formulario es válido; si lo es, llamamos a save() sin argumentos porque la entrada ya está asociada al tema ❹ correcto . A continuación, redirigimos a la página topic, donde el usuario debería ver la versión actualizada de la entrada que editó ❺.

Si mostramos un formulario inicial para editar la entrada o si el formulario enviado no es válido, creamos el diccionario context y renderizamos la página utilizando la plantilla *edit\_entry.html*.

#### La plantilla edit\_entry

A continuación, creamos una plantilla *edit\_entry.html* , que es similar a *new\_entry.html*:

**edit\_entry.html**

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

El argumento action devuelve el formulario a la función edit\_entry() para su procesamiento ❶. Incluimos el entry.id como argumento en la etiqueta {% url %}, para que la función vista pueda modificar el objeto de entrada correcto. Etiquetamos el botón de envío como Save changes para recordar al usuario que está guardando ediciones, no creando una nueva entrada ❷.

#### Enlace a la página edit\_entry

Ahora tenemos que incluir un enlace a la página edit\_entry para cada entrada de la página de temas:

**topic.html**

--snip--  
 {% for entry in entries %}  
 <li>  
 <p>{{ entry.date\_added|date:'M d, Y H:i' }}</p>  
 <p>{{ entry.text|linebreaks }}</p>  
 <p>  
 <a href="{% url 'learning\_logs:edit\_entry' entry.id %}">  
 Edit entry</a></p>  
 </li>  
--snip--

Incluimos el enlace de edición después de mostrar la fecha y el texto de cada entrada. Utilizamos la etiqueta de plantilla {% url %} para determinar la URL del patrón de URL con nombre edit\_entry, junto con el atributo ID de la entrada actual en el bucle (entry.id). El texto del enlace Edit entry aparece después de cada entrada de la página. [La Figura 19-3](#figure19-3) muestra el aspecto de la página del tema con estos enlaces.

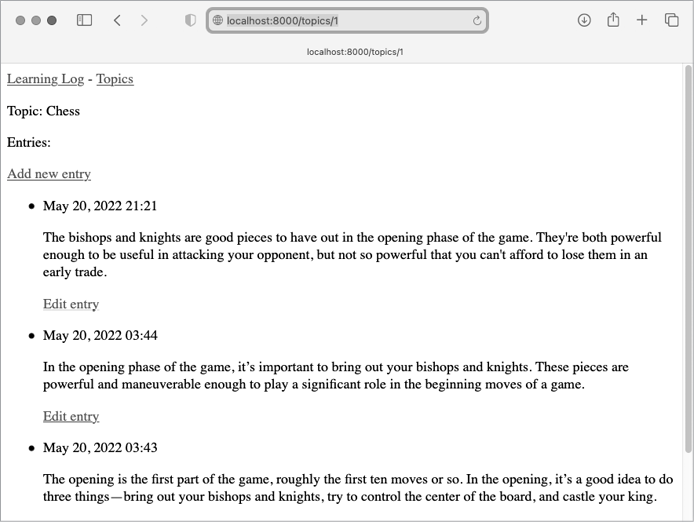


Figura 19-3: Cada entrada tiene ahora un enlace para editarla.

El Registro de Aprendizaje tiene ahora la mayor parte de la funcionalidad que necesita. Los usuarios pueden añadir temas y entradas, y pueden leer cualquier conjunto de entradas que deseen. En la próxima sección, implementaremos un sistema de registro de usuarios para que cualquiera pueda crear una cuenta en Learning Log y crear su propio conjunto de temas y entradas.

[anterior](c19_4.html)[Subtema 5 de 23: (Ver todo)](c19.html)[siguiente](c19_6.html)