### Definir el modelo de entrada

Para que un usuario pueda registrar lo que ha estado aprendiendo sobre ajedrez y escalada, necesitamos definir un modelo para los tipos de entradas que los usuarios pueden hacer en sus registros de aprendizaje. Cada entrada debe estar asociada a un tema concreto. Esta relación se denomina *many-to-one relationship*, lo que significa que muchas entradas pueden asociarse a un tema.

Aquí tienes el código del modelo Entry. Colócalo en tu archivo *models.py* :

**models.py**

from django.db import models  
  
class Topic(models.Model):  
 --snip--  
  
❶ class Entry(models.Model):  
 """Something specific learned about a topic."""  
❷ topic = models.ForeignKey(Topic, on\_delete=models.CASCADE)  
❸ text = models.TextField()  
 date\_added = models.DateTimeField(auto\_now\_add=True)  
  
❹ class Meta:  
 verbose\_name\_plural = 'entries'  
  
 def \_\_str\_\_(self):  
 """Return a simple string representing the entry."""  
❺ return f"{self.text[:50]}..."

La clase Entry hereda de la clase base Model de Django, al igual que Topic ❶. El primer atributo, topic, es una instancia de ForeignKey ❷. Un *foreign key* es un término de base de datos; es una referencia a otro registro de la base de datos. Es el código que conecta cada entrada con un tema concreto. A cada tema se le asigna un *key*, o ID, cuando se crea. Cuando Django necesita establecer una conexión entre dos datos, utiliza las claves asociadas a cada información. En breve utilizaremos estas conexiones para recuperar todas las entradas asociadas a un determinado tema. El argumento on\_delete=models.CASCADE indica a Django que cuando se elimine un tema, se eliminen también todas las entradas asociadas a ese tema. Esto se conoce como *cascading delete*.

A continuación hay un atributo llamado text, que es una instancia de TextField ❸. Este tipo de campo no necesita un límite de tamaño, porque no queremos limitar el tamaño de las entradas individuales. El atributo date\_added nos permite presentar las entradas en el orden en que fueron creadas, y colocar una marca de tiempo junto a cada entrada.

La clase Meta está anidada dentro de la clase Entry ❹. La clase Meta contiene información adicional para gestionar un modelo; aquí, nos permite establecer un atributo especial que indica a Django que utilice Entries cuando necesite hacer referencia a más de una entrada. Sin esto, Django se referiría a varias entradas como Entrys.

El método \_\_str\_\_() indica a Django qué información debe mostrar cuando hace referencia a entradas individuales. Como una entrada puede ser un cuerpo de texto largo, \_\_str\_\_() devuelve sólo los primeros 50 caracteres de text ❺. También añadimos una elipsis para aclarar que no siempre mostramos la entrada completa.

[anterior](c18_21.html)[Subtema 22 de 41: (Ver todo)](c18.html)[siguiente](c18_23.html)