Cinco. Modelos

Lo que queremos crear ahora es algo que almacene todas las entradas de nuestro blog. Pero para poder hacerlo tenemos que hablar un poco sobre algo llamado objetos.

Objetos

Hay un concepto en el mundo de la programación llamado programación orientada a objetos. La idea es que en lugar de escribir todo como una aburrida secuencia de instrucciones de programación podemos modelar cosas y definir cómo interactúan entre ellas.

Es un conjunto de propiedades y acciones. Suena raro, pero te daremos un ejemplo.

Si queremos modelar un gato crearemos un objeto Gato que tiene algunas propiedades como: color, edad, temperamento (como bueno, malo, o dormilón ;)), y dueño (este es un objeto Persona o en caso de un gato callejero, esta propiedad está vacía).

Luego, el Gato tiene algunas acciones como: ronronear, arañar o alimentar (en cuyo caso daremos al gato algo de ComidaDeGato, el cual debería ser un objeto aparte con propiedades como sabor).

```
Gato
---------
color
edad
humor
dueño
ronronear()
rasguñar()
alimentarse(comida_de_gato)

ComidaDeGato
----------
sabor


ComidaDeGato
--------
sabor
```

Básicamente se trata de describir cosas reales en el código con propiedades (llamadas propiedades del objeto) y las acciones (llamadas métodos).

Aunque es posible usar Django sin una base de datos, Django cuenta con un mapeador objeto-relacional en el que es posible definir la estructura de la base de datos utilizando código Python.

La sintaxis de modelo de datos ofrece muchas formas de representar sus modelos – hasta ahora, ha resuelto una cantidad equivalente a años de problemas de esquemas de base de datos. Aquí hay un ejemplo rápido:

```python
from django.db import models

class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __str__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)

    def __str__(self):
        return self.headline
```

A continuación ejecute la utilidad de la línea de comandos de Django para crear las tablas de base de datos de forma automática:

```
$ python manage.py migrate
```

El comando migrate analiza todos sus modelos disponibles y crea cualquier tabla faltante en su base de datos, así como ofrece opcionalmente un control de esquema mucho más rico.

https://tutorial.djangogirls.org/es/django_models/

https://docs.djangoproject.com/es/2.1/intro/overview/