PILLOW

Introducción

La librería Pillow se usa con Django si necesitamos manipular imágenes en tu sitio web. Pillow es una poderosa librería de procesamiento de imágenes que se utiliza en Python para abrir, manipular y guardar imágenes en diferentes formatos de archivo.

Tareas que se pueden realizar con Pillow::

- [ ] Redimensionar imágenes: puedes cambiar el tamaño de las imágenes para que se ajusten a diferentes tamaños de pantalla o para reducir el tamaño del archivo.
- [ ] Recortar imágenes: puedes recortar partes de una imagen para mostrar solo una porción específica.
- [ ] Aplicar filtros: puedes aplicar diferentes filtros a las imágenes, como efectos de sepia, blanco y negro, entre otros.
- [ ] Agregar marcas de agua: puedes agregar una marca de agua a tus imágenes para proteger tus derechos de autor.
- [ ] Optimizar imágenes: puedes reducir el tamaño de los archivos de imagen sin perder calidad para mejorar la velocidad de carga de tu sitio web.

Enlaces de documentación

- [ ] Página de documentación oficial → https://pillow.readthedocs.io/en/stable/
- [ ] Ejemplos en github → https://github.com/python-pillow/Pillow/tree/main/Tests
- [ ] Ejemplos con django de pillow → https://docs.djangoproject.com/en/3.2/topics/files/#handling-uploaded-files-with-pillow

Ejemplos

Redimensionar una imagen en una plantilla de Django

Supongamos que tenemos una imagen llamada "image.jpg" que queremos redimensionar en nuestra plantilla de Django. Podemos hacerlo utilizando el siguiente código:

```
{% load static %}

{% with image_path=static('images/image.jpg') %}
  {% with im=Image.open(image_path) %}
    {% with resized=im.resize((400, 400)) %}
      <img src="{{ resized.url }}">
    {% endwith %}
  {% endwith %}
{% endwith %}
```

Este código primero carga la imagen utilizando la etiqueta "static" de Django para obtener su ruta en el servidor. Luego, utiliza la librería Pillow para abrir la imagen y redimensionarla a un tamaño de 400 x 400 píxeles. Finalmente, se muestra la imagen redimensionada en la plantilla.

Aplicar un filtro a una imagen en una plantilla de Django

Supongamos que queremos aplicar un filtro de sepia a una imagen llamada "image.jpg". Podemos hacerlo utilizando el siguiente código:

```
{% load static %}

{% with image_path=static('images/image.jpg') %}
  {% with im=Image.open(image_path) %}
    {% with sepia=im.filter(ImageFilter.SEPIA) %}
      <img src="{{ sepia.url }}">
    {% endwith %}
  {% endwith %}
{% endwith %}
```

Este código carga la imagen y la abre utilizando la librería Pillow. Luego, aplica el filtro de sepia utilizando el método "filter" de la clase Image. Finalmente, se muestra la imagen con el filtro de sepia aplicado en la plantilla.

Crear miniaturas de imágenes (función)

Pillow nos permite crear miniaturas de una imagen de forma muy sencilla. Supongamos que tenemos una imagen llamada "image.jpg" y queremos crear una miniatura de tamaño 200 x 200 píxeles. Podemos hacerlo utilizando el siguiente código.

```
from PIL import Image

im = Image.open('image.jpg')
im.thumbnail((200, 200))
im.save('image_thumbnail.jpg')
```

Este código abre la imagen "image.jpg" y utiliza el método "thumbnail" para crear una miniatura de tamaño 200 x 200 píxeles. Luego, guarda la miniatura en un archivo llamado "image_thumbnail.jpg".

Rotar una imagen (función)

Pillow nos permite rotar una imagen en cualquier ángulo. Supongamos que tenemos una imagen llamada "image.jpg" y queremos rotarla 45 grados en sentido horario. Podemos hacerlo utilizando el siguiente código:

```
from PIL import Image

im = Image.open('image.jpg')
im_rotated = im.rotate(-45)
im_rotated.save('image_rotated.jpg')
```

Este código abre la imagen "image.jpg" y utiliza el método "rotate" para rotarla 45 grados en sentido horario (o -45 grados en sentido antihorario). Luego, guarda la imagen rotada en un archivo llamado "image_rotated.jpg".

Cambiar el tamaño de varias imágenes

Si necesitamos cambiar el tamaño de varias imágenes, podemos automatizar el proceso utilizando un ciclo "for". Supongamos que tenemos varias imágenes en la carpeta "images/" y queremos redimensionarlas a un tamaño de 400 x 400 píxeles. Podemos hacerlo utilizando el siguiente código

```
from PIL import Image
import os

folder = 'images/'
for filename in os.listdir(folder):
    if filename.endswith('.jpg'):
        im = Image.open(os.path.join(folder, filename))
        im_resized = im.resize((400, 400))
        im_resized.save(os.path.join(folder, f'resized_{filename}'))
```

Este código recorre todos los archivos en la carpeta "images/" y verifica que tengan extensión ".jpg". Para cada archivo, utiliza el método "resize" para redimensionar la imagen a un tamaño de 400 x 400 píxeles y la guarda en un archivo con el prefijo "resized_".

CRISPY FORM

```python
from django import forms
from crispy_forms.helper import FormHelper
from crispy_forms.layout import Layout, Submit, Row, Column
from .models import Contact

class ContactForm(forms.ModelForm):
    name = forms.CharField(label='Nombre', max_length=100)
    email = forms.EmailField(label='Correo electrónico')
    phone = forms.CharField(label='Teléfono', max_length=20)
    message = forms.CharField(label='Mensaje', widget=forms.Textarea)

    class Meta:
        model = Contact
        fields = ('name', 'email', 'phone', 'message')

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.helper = FormHelper()
        self.helper.form_method = 'post'
        self.helper.add_input(Submit('submit', 'Enviar', css_class='btn-pri

        self.helper.layout = Layout(
            Row(
                Column('name', css_class='form-group col-md-6 mb-0'),
                Column('email', css_class='form-group col-md-6 mb-0'),
                css_class='form-row'
            ),
            Row(
                Column('phone', css_class='form-group col-md-6 mb-0'),
                css_class='form-row'
            ),
            'message',
        )
```