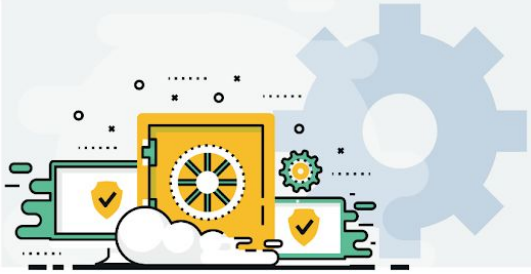
INFORMATORIO

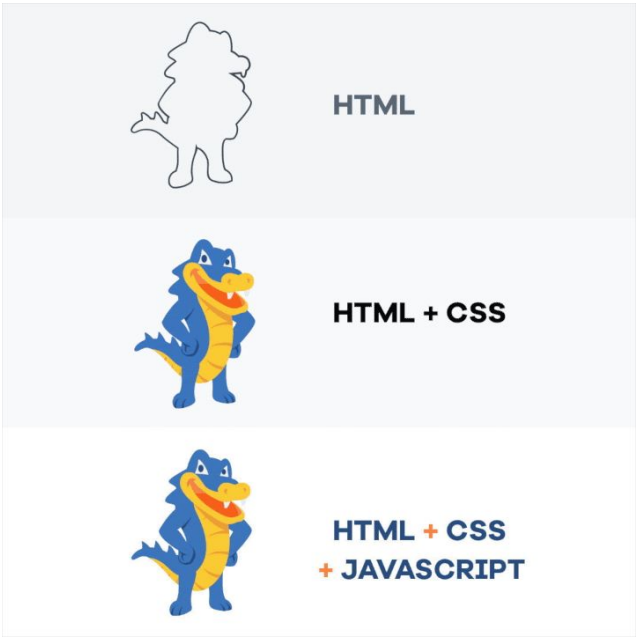
Taller de Programación Web

DJANGO: Static Files



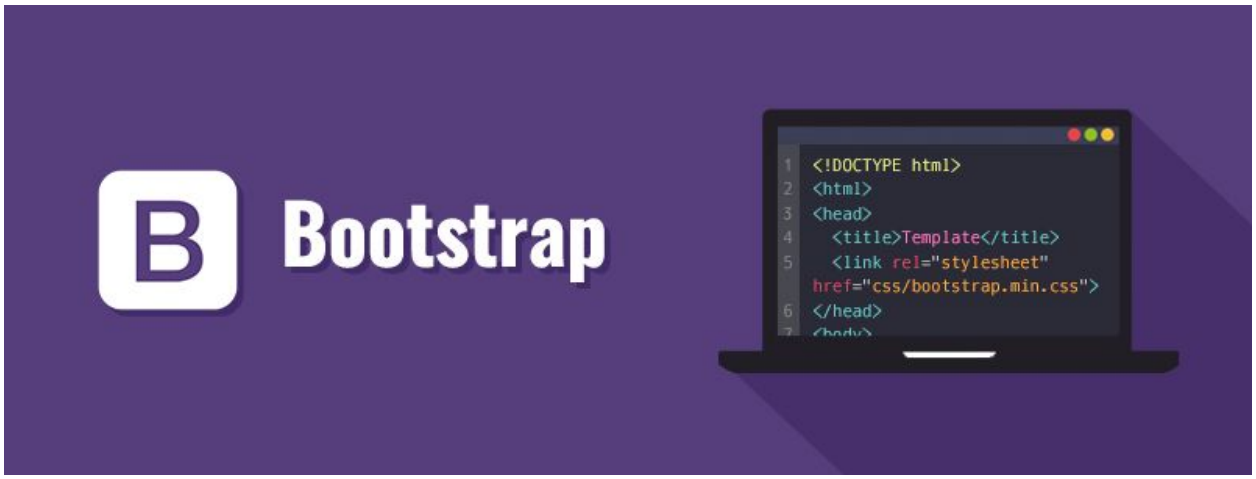
Archivos estáticos y template tags de los mismos

Una vez que ya tenemos nuestra aplicación y nuestro html desplegado en nuestro navegador debemos tener en cuenta que las aplicaciones Web, por lo general, contienen archivos adicionales tales como imágenes, JavaScript o CSS necesarios para renderizar la página web completa. En Django, nos referimos a estos archivos como **«archivos estáticos»**.



¿Qué es CSS?

El lenguaje **CSS** (las siglas en inglés de Hojas de Estilos en Cascada, o Cascading Style Sheets) sirve para describir la apariencia de un sitio web escrito en un lenguaje de marcado (como HTML). Es como la capa de pintura para nuestra página web.



Usemos Bootstrap

Bootstrap es una de las librerías de HTML y CSS más populares para desarrollar sitios web con estilo!: https://getbootstrap.com/

Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más.

Además de todas las características que ofrece, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles. Esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada.

Para instalar Bootstrap, abre tu fichero `.html` en el editor de código y añade esto a la sección `<head>`:

```
<!-- CSS only -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ
0Z" crossorigin="anonymous">

<!-- JS, Popper.js, and jQuery -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRk
fj" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7
lN" crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+
rV" crossorigin="anonymous"></script>
```

Lo bueno de utilizar este método de importación es que no añade ningún archivo a tu proyecto. Solo apunta a archivos que existen en Internet. Así que sigamos adelante, accede a tu sitio web y actualiza la página, deberías verlo con un estilo diferente.

Archivos estáticos (static files) en Django

Ahora nos vamos a fijar en esto que hemos estado llamando **archivos estáticos**. Como ya dijimos, los archivos estáticos son los archivos CSS e imágenes. Su contenido no depende del contexto de la petición y siempre será el mismo para todos los usuarios.

Por suerte Django nos provee de una librería **`django.contrib.staticfiles`** para ayudarnos a manejar estos recursos. Para hacer uso de los recursos estáticos debemos seguir los siguientes pasos:

1. Asegúrate que `django.contrib.staticfiles` está incluido en el `INSTALLED_APPS` del settings.py de tu proyecto.

2. En tu archivo de settings, debemos definir la variable STATIC_URL, por ejemplo:

```
STATIC_URL = '/static/'
```

3. Guarda tus recursos estáticos dentro de tu app en un directorio con nombre **static** para que Django pueda encontrarlo, y a su vez crear una carpeta por cada tipo de contenido(css, js, img,etc) , por ejemplo:

```
djangoproject
├──────myapp
│       ├──────migrations
│       ├──────static
│       ├──────templates
│       ├──────...
│
└──────mysite
```

4. Para poder utilizar los recursos estáticos debemos utilizar el *tag* `{% static 'img/icon.png' %}` . Para hacer uso del *tag* `{% static … %}` siempre debes cargarlo en tu template utilizando el *tag* `{% load … %}` dentro de nuestro .html ubicado en la carpeta template de tu app, ejemplo:

```
{% load static %}
<img src="{% static "my_app/example.jpg"
%}" alt="My image"/>
```

Creemos un CSS

Vamos a crear un archivo CSS, para añadir tu propio estilo a la página. Crea un nuevo directorio llamado `css` dentro de la carpeta `static`. A continuación, crea un nuevo archivo llamado `app.css` dentro de la carpeta `css`.

```
djangoproject
    └──────── myapp
              └──────── static
                        └──────── css
                                  └──────── app.css
```

Vamos a escribir algo de CSS. Abre el archivo `myapp/static/css/app.css` en el editor de código.

No vamos a profundizar demasiado en cómo personalizar y aprender CSS. Pero en las redes dispones de cursos, tutoriales y material infinitos, como por ejemplo: https://www.w3schools.com/css/

En el archivo `css` deberías añadir el siguiente código:

```
h1 a, h2 a {
    color: #C25100;
}
```

`h1 a` es un selector CSS. Esto significa que estamos aplicando nuestros estilos a cualquier elemento de `a` dentro de un elemento de `h1`; el selector `h2 a` hace lo mismo para los elementos de `h2`. Así, cuando tenemos algo como `<h1><a href="">link</a></h1>`, se aplicará el estilo `h1 a`. En este caso, le estamos diciendo que cambie el color a un `#C25100`, que es un naranjo oscuro.

En un archivo CSS se definen los estilos de los elementos que aparecen en el archivo HTML. La primera forma de identificar los elementos es por su nombre. Puede que los recuerdes como 'tags' de la sección de HTML. Cosas como `a`, `h1`, y `body` son algunos ejemplos de nombres de elementos. También podemos identificar elementos por el atributo `class` o el atributo `id`. Los valores de "class" e "id" son nombres que das al elemento para poderlo identificar. Con el atributo "class" identificamos grupos de elementos del mismo tipo y con el atributo "id" identificamos un elemento específico. Por ejemplo, el siguiente elemento lo podrías identificar por su nombre de "tag" `a`, por su "class" `external_link`, o por su "id" `link_to_wiki_page`:

```
<a  href="https://en.wikipedia.org/wiki/Django"  class="external_link" id="link_to_wiki_page">
```

Ahora necesitamos decirle a nuestra plantilla HTML que hemos añadido código CSS, agregando la línea `{% load static %}` al principio de todo, como ya lo dijimos anteriormente. Aquí solo estamos cargando archivos estáticos. Entre las etiquetas `<head>` y `</head>`, después de los enlaces a los archivos CSS de Bootstrap, añade esta línea:

```
<link rel="stylesheet" href="{% static 'css/myapp.css' %}">
```

Finalmente volvé al navegador y recargá la página para ver los nuevos estilos que desarrollamos. Ahora te toca a vos darle los estilos que quieras a tu página.