Práctica 13: CSS

Nuestro blog todavía se ve bastante feo, es hora de hacerlo bonito. Vamos a usar CSS para eso.

¿Qué es CSS?

Cascading Style Sheets (CSS), que significa 'hojas de estilo en cascada'. Es un lenguaje utilizado para describir el aspecto y el formato de un sitio web escrito en lenguaje de marcado (como HTML). Trátalo como maquillaje para páginas web.

Pero no queremos empezar de cero otra vez. Una vez más, usaremos algo que ya ha sido realizado por programadores y publicado en Internet de forma gratuita. Ya sabes, debemos de utilizar todas las herramientas disponibles y no reinventar la rueda.

Usando Bootstrap

Bootstrap es uno de los frameworks HTML y CSS más populares para desarrollar webs agradables al ojo: https://getbootstrap.com/

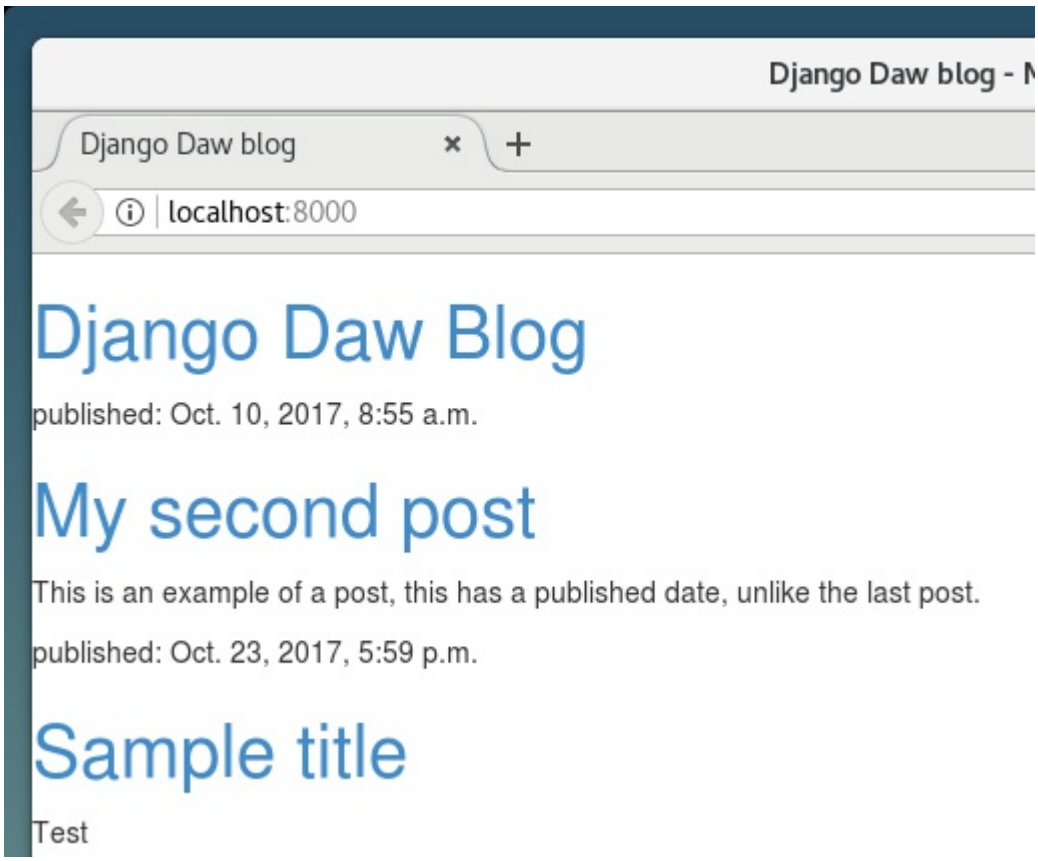
Lo escribieron programadores que trabajaban para Twitter y ahora lo desarrollan voluntarios de todo el mundo.

Instalar Bootstrap

Para instalar Bootstrap tienes que añadir esto al `<head>` de tu fichero `.html` (`blog/templates/blog/post_list.html`):

```
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
```

Esto no añade ningún fichero a tu proyecto. Simplemente apunta a ficheros que existen en Internet. Adelante, abre tu sitio web y actualiza la página. Aquí está:



Se ve mucho mejor.

Ficheros estáticos en Django

Finalmente nos vamos a fijar en estas cosas que hemos estado llamando **ficheros estáticos**. Los ficheros estáticos son todos tus CSS e imágenes; ficheros que no son dinámicos, por lo que su contenido no depende del contexto de la petición y serán iguales para todos los usuarios.

Dónde poner los ficheros estáticos para Django

Como has visto cuando hemos ejecutado `collectstatic` en el servidor, Django ya sabe dónde encontrar los ficheros estáticos para la aplicación "admin". Ahora necesitamos añadir algunos ficheros estáticos para nuestra propia aplicación, `blog`.

Esto lo conseguimos creando una carpeta llamada `static` dentro de la aplicación blog:

```
django-daw
├── blog
│   ├── migrations
│   └── static
└── mysite
```

Django encontrará automáticamente cualquier carpeta que se llame "static" dentro de las carpetas de tus aplicaciones y podrá utilizar su contenido como ficheros estáticos.

Tu primer fichero CSS

Crea un nuevo directorio llamado `css` dentro de tu directorio `static`. Para añadir tu propio estilo a tu página web, crea un nuevo fichero llamado `blog.css` dentro de este directorio `css`. ¿Listo?

```
django-daw
└── blog
    └── static
        └── css
            └── blog.css
```

Es hora de escribir algo de CSS. Abre el fichero `blog/static/css/blog.css` en tu editor de código (sublimetext o nano).

No vamos a entrar mucho en la personalización y el aprendizaje de CSS aquí porque es bastante fácil y lo puedes aprender por tu cuenta después de está práctica. Recomiendo enormemente hacer este curso de HTML y CSS en Codecademy para aprender todo lo que necesitas saber sobre cómo hacer tus sitios web más bonitos con CSS.

Pero vamos a hacer un poco al menos. ¿Tal vez podríamos cambiar el color de nuestro título? Los ordenadores utilizan códigos especiales (hexadecimal) para entender los colores. Empiezan con `#` y les siguen 6 letras (A-F) y números (0-9). Puedes encontrar códigos de color, por ejemplo, aquí: http://www.colorpicker.com/. También puedes utilizar colores predefinidos utilizando su nombre en inglés, como `red` y `green`.

En tu fichero `blog/static/css/blog.css` deberías añadir el siguiente código:

```
h1 a {
    color: #FCA205;
}
```

`h1 a` es un selector CSS. Quiere decir que estamos aplicando nuestros estilos a cualquier elemento `a` que se encuentre dentro de un elemento `h1` (por ejemplo cuando tenemos en código algo como: `<h1><a href="">enlace</a></h1>`). En este caso le estamos diciendo que cambie el color a `#FCA205`, que es naranja. Por supuesto, ¡puedes poner tu propio color aquí!

En un fichero CSS definimos los estilos para los elementos del fichero HTML. Los elementos se identifican por el nombre del elemento (es decir, `a`, `h1`, `body`), el atributo `class` (clase) o el atributo `id` (identificador). Class e id son nombres que le asignas tú mismo al elemento. Las clases definen grupos de elementos y los ids apuntan a elementos específicos. Por ejemplo, la siguiente etiqueta se puede identificar con CSS usando el nombre `a`, la clase `external_link` o el id `link_to_wiki_page`:

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

Puedes leer más sobre selectores de CSS en w3schools.

También necesitamos decirle a nuestra plantilla HTML que hemos añadido CSS. Abre el fichero `blog/templates/blog/post_list.html` y añade esta línea justo al principio:

```
{% load staticfiles %}
```

Aquí sólo estamos cargando ficheros estáticos. Luego, entre el `<head>` y `</head>`, después de los enlaces a los ficheros CSS de Bootstrap (el navegador lee los ficheros en el orden en que están, así que nuestro fichero podría sobrescribir partes del código de Bootstrap), añade la siguiente línea:

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

Le acabamos de decir a nuestra plantilla dónde se encuentra nuestro fichero CSS.

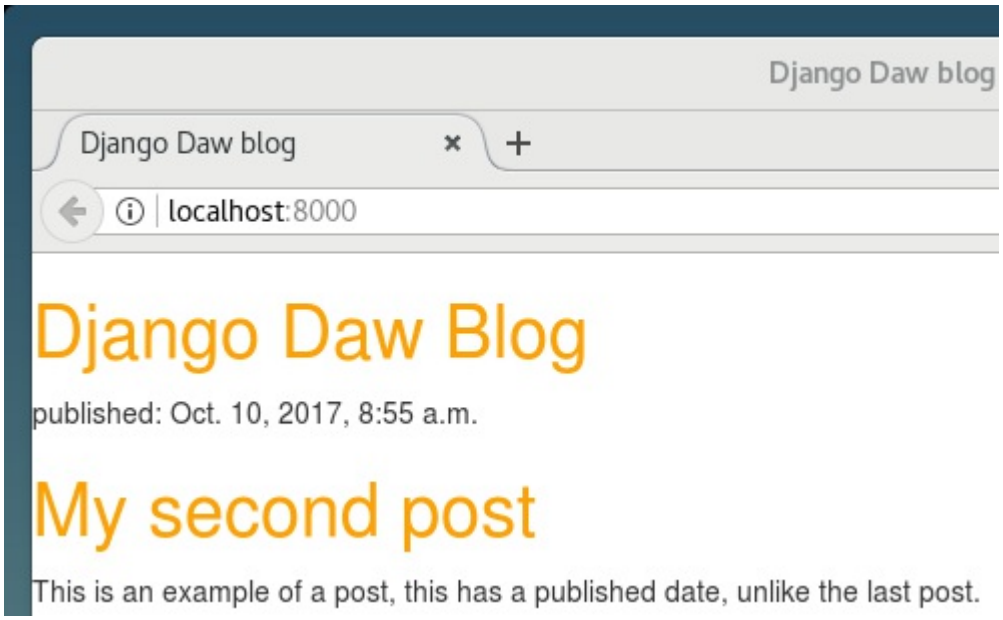
Ahora tu fichero debería tener este aspecto:

```
{% load staticfiles %}
<html>
```

```
        <head>
            <title>Django Daw blog</title>
            <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
            <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
            <link rel="stylesheet" href="{% static 'css/blog.css' %}">
        </head>
        <body>
            <div>
                <h1><a href="/">Django Daw Blog</a></h1>
            </div>

            {% for post in posts %}
                <div>
                    <p>published: {{ post.published_date }}</p>
                    <h1><a href="">{{ post.title }}</a></h1>
                    <p>{{ post.text|linebreaksbr }}</p>
                </div>
            {% endfor %}
        </body>
    </html>
```

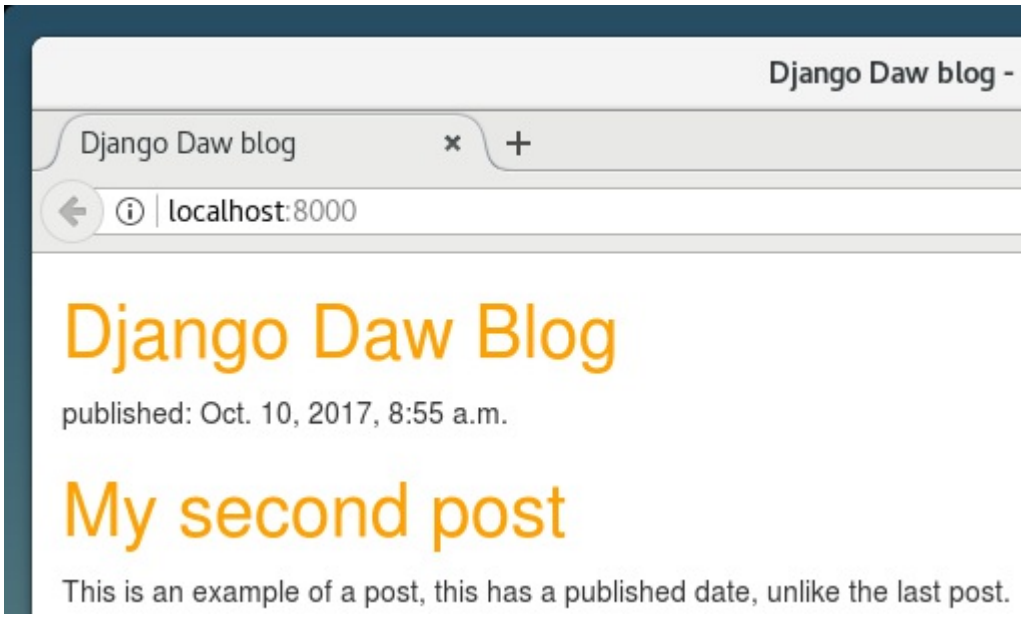
De acuerdo, guarda el fichero y actualiza el sitio.



Buen trabajo. ¿Quizá nos gustaría darle un poco de aire a nuestro sitio web y aumentar también el margen en el lado izquierdo?. Hagámoslo:

```
    body {
        padding-left: 15px;
    }
```

Añade esto a tu CSS, guarda el fichero y ¡mira cómo funciona!



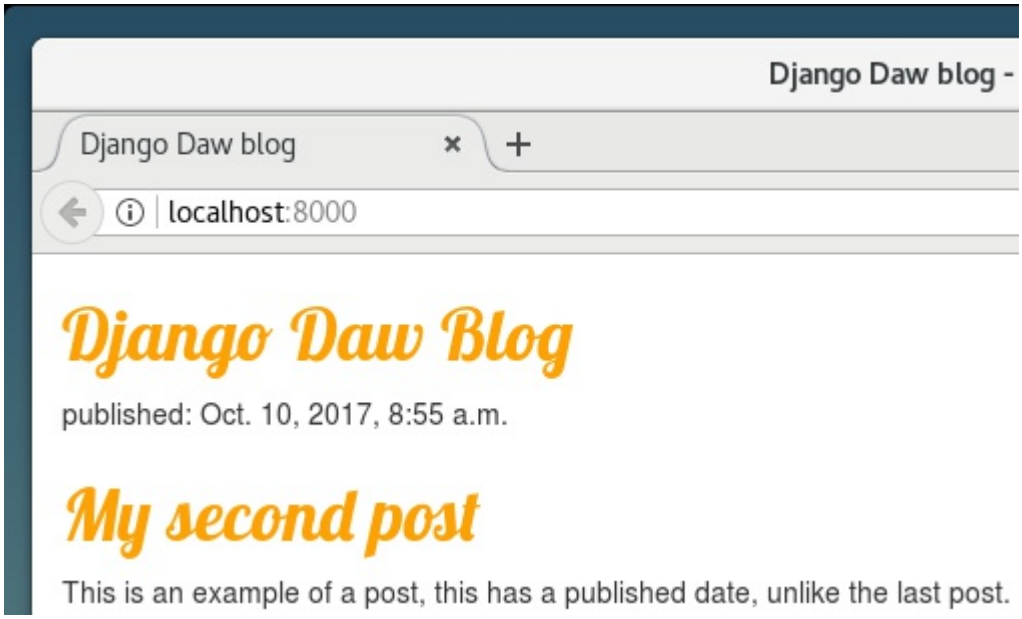
¿Quizá podríamos personalizar la tipografía del título? Pega esto en la sección `<head>` del fichero `blog/templates/blog/post_list.html`:

```
    <link href="https://fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext" rel="stylesheet" type="text/css">
```

Esta línea va a importar una tipografía llamada *Lobster* de Google Fonts (https://www.google.com/fonts).

Ahora añade la línea `font-family: 'Lobster';` en el fichero CSS `blog/static/css/blog.css` dentro del bloque de declaración `h1 a` (el código entre llaves `{` y `}`) y actualiza la página:

```
h1 a {
    color: #FCA205;
    font-family: 'Lobster';
}
```



Como se mencionó anteriormente, CSS tiene un concepto de clases que básicamente permite nombrar una parte del código HTML y aplicar estilos sólo a esta parte, sin afectar a otras. Es muy útil si tienes dos divs que hacen algo muy diferente (como el encabezado y la entrada) y no quieres que tengan el mismo aspecto.

Nombra algunas partes del código HTML. Añade una clase llamada `page-header` al `div` que contiene el encabezado, así:

```
    <div class="page-header">
        <h1><a href="/">Django Daw Blog</a></h1>
    </div>
```

Y ahora añade la clase `post` al `div` que contiene una entrada del blog.

```
    <div class="post">
```

```
        <p>published: {{ post.published_date }}</p>
        <h1><a href="">{{ post.title }}</a></h1>
        <p>{{ post.text|linebreaksbr }}</p>
    </div>
```

Ahora añadiremos bloques de declaración a diferentes selectores. Los selectores que comienzan con `.` hacen referencia a las clases. Hay muchos tutoriales y explicaciones sobre CSS en la web que te ayudarán a entender el siguiente código. Por ahora, simplemente copia y pega este bloque de código en tu fichero `blog/static/css/blog.css`:

```
.page-header {
    background-color: #ff9400;
    margin-top: 0;
    padding: 20px 20px 20px 40px;
}

.page-header h1, .page-header h1 a, .page-header h1 a:visited, .page-header h1 a:active {
    color: #ffffff;
    font-size: 36pt;
    text-decoration: none;
}

.content {
    margin-left: 40px;
}

h1, h2, h3, h4 {
    font-family: 'Lobster', cursive;
}

.date {
    float: right;
    color: #828282;
}

.save {
    float: right;
}

.post-form textarea, .post-form input {
    width: 100%;
}

.top-menu, .top-menu:hover, .top-menu:visited {
    color: #ffffff;
    float: right;
    font-size: 26pt;
    margin-right: 20px;
}

.post {
    margin-bottom: 70px;
}

.post h1 a, .post h1 a:visited {
    color: #000000;
}
```

Luego rodea el código HTML que muestra las entradas con las declaraciones de clases. Sustituye esto:

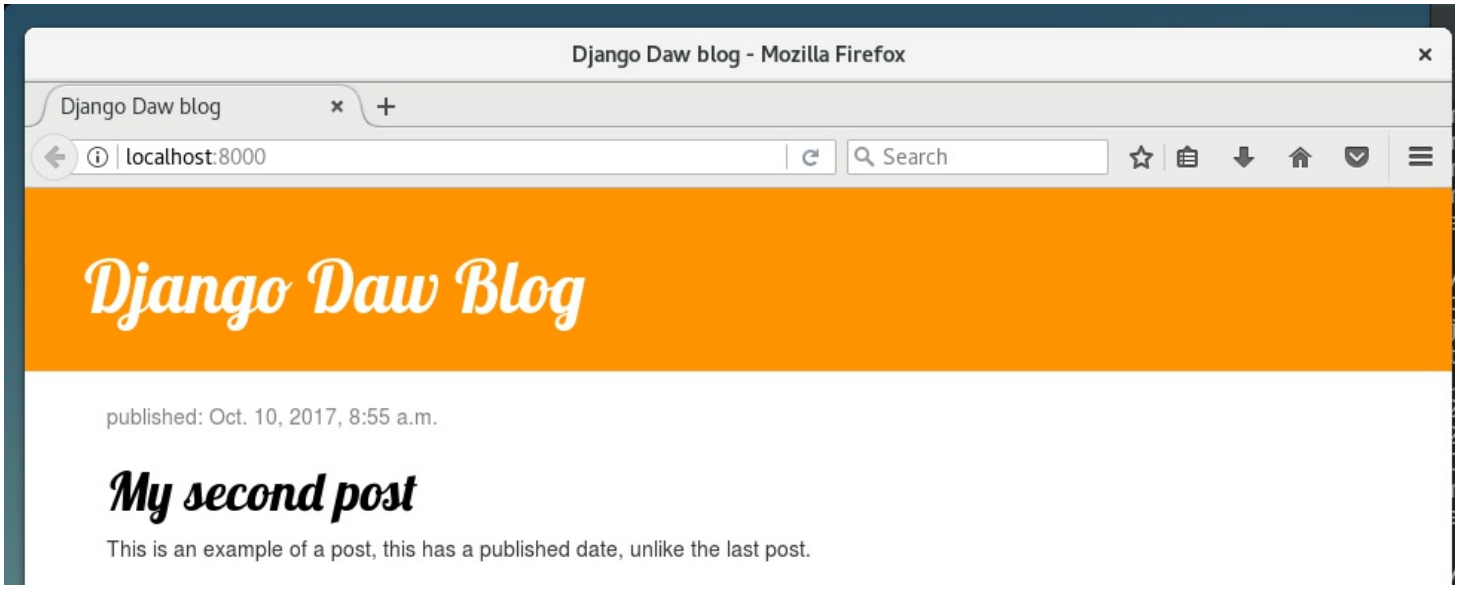
```
{% for post in posts %}
    <div class="post">
        <p>published: {{ post.published_date }}</p>
        <h1><a href="">{{ post.title }}</a></h1>
        <p>{{ post.text|linebreaksbr }}</p>
    </div>
{% endfor %}
```

en `blog/templates/blog/post_list.html` por esto:

```
<div class="content container">
    <div class="row">
        <div class="col-md-8">
            {% for post in posts %}
                <div class="post">
                    <div class="date">
                        {{ post.published_date }}
                    </div>
                    <h1><a href="">{{ post.title }}</a></h1>
                    <p>{{ post.text|linebreaksbr }}</p>
```

```
                </div>
            {% endfor %}
        </div>
    </div>
</div>
```

Guarda los ficheros y actualiza tu sitio.



Bien!, se ve mucho mejor. En realidad el código que acabamos de pegar no es tan difícil de entender y deberías ser capaz de entender la mayoría sólo con leerlo.

No tengas miedo de jugar un poco con este CSS e intentar cambiar algunas cosas. Si rompes algo, no te preocupes, siempre puedes deshacerlo.

De cualquier forma, te vuelvo a recomendar que hagas el curso de HTML y CSS de Codeacademy como una tarea post-clase para que aprendas todo lo que necesitas saber para hacer tus sitios web más bonitos con CSS.