Práctica 10: Introducción a HTML

Te estarás preguntando, ¿Qué es una plantilla?. Una plantilla es un archivo que podemos reutilizar para presentar información diferente de forma consistente – por ejemplo, se podría utilizar una plantilla para ayudarte a escribir una carta, porque aunque cada carta puede contener un mensaje distinto y dirigirse a una persona diferente, compartirán el mismo formato.

El formato de una plantilla de Django se describe en HTML.

¿Qué es HTML?

HTML es un simple código que es interpretado por tu navegador web - como Chrome, Firefox o Safari - para mostrar una página web al usuario.

HTML significa HyperText Markup Language – en español, Lenguaje de Marcado de HyperTexto. **HyperText** significa que es un tipo de texto que soporta hipervínculos entre páginas. **Markup** significa que hemos tomado un documento y lo hemos marcado con código para decirle a algo (en este caso, un navegador) cómo interpretar la página. El código HTML está construido con **tags**, cada una comenzando con `<` y terminando con `>`. Estas etiquetas de marcado son llamadas **elements**.

¡Tu primera plantilla!

Crear una plantilla significa crear un archivo de plantilla. Todo es un archivo, probablemente hayas notado esto ya.

Las plantillas se guardan en el directorio de `blog/templates/blog`. Así que primero crea un directorio llamado `templates` dentro de tu directorio blog. Luego crea otro directorio llamado `blog` dentro de tu directorio de templates:

```
blog
└── templates
    └── blog
```

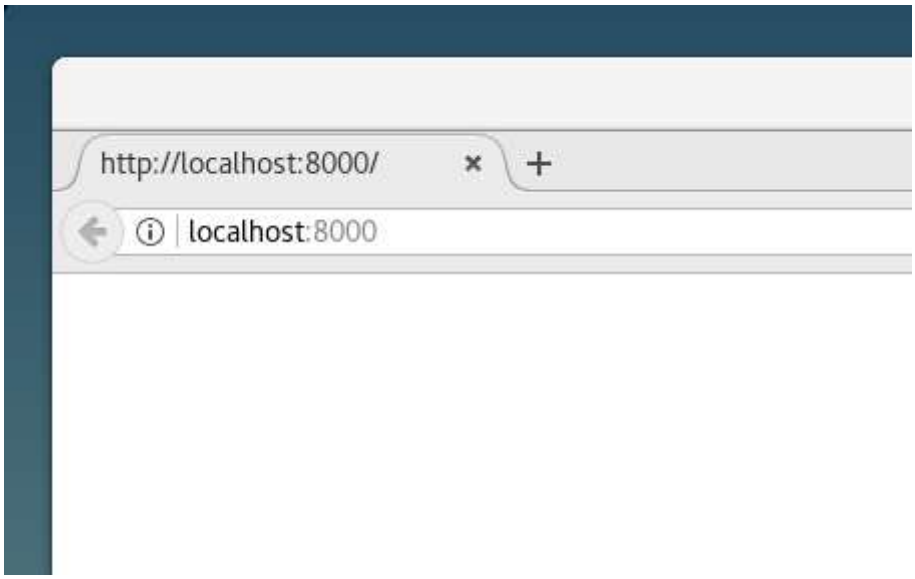
(Tal vez te preguntes por qué necesitamos dos directorios llamados `blog` – como descubrirás más adelante, esto es simplemente una útil convención de nomenclatura que hace la vida más

fácil cuando las cosas empiezan a complicarse más.

Y ahora crea un archivo `post_list.html` (déjalo en blanco por ahora) dentro de la carpeta `blog/templates/blog`.

Mira cómo se ve su sitio web ahora: http://127.0.0.1:8000/

> Si todavía tienes un error `TemplateDoesNotExists`, intenta reiniciar el servidor. Ve a la línea de comandos, detén el servidor pulsando Ctrl + C (teclas Control y C juntas) y comienza de nuevo mediante la ejecución del comando `python manage.py runserver`.

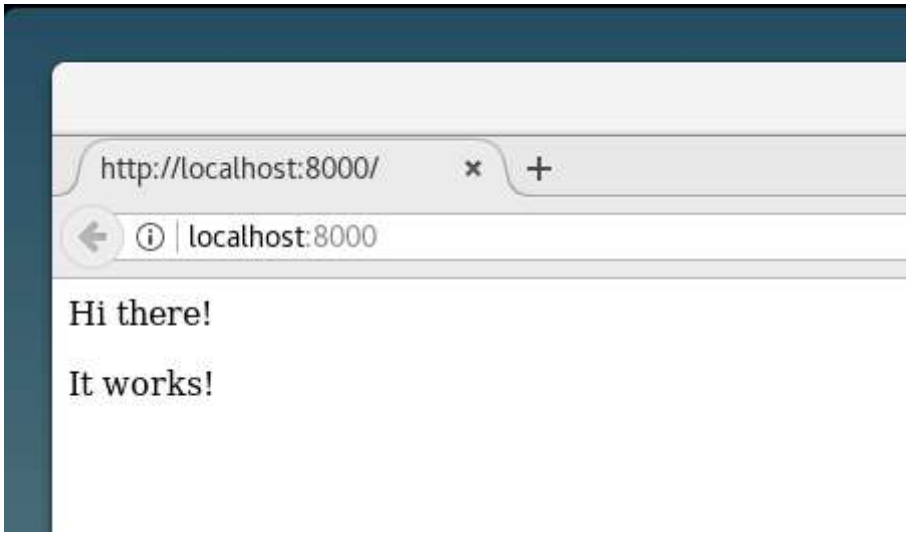


Ningún error. Sin embargo, por ahora, tu sitio web no está publicando nada excepto una página en blanco, porque la plantilla también está vacía. Tenemos que arreglarlo.

Añade lo siguiente a tu archivo de plantilla:

```
<html>
<body>
    <p>Hi there!</p>
    <p>It works!</p>
</body>
</html>
```

¿Cómo luce ahora tu sitio web? Haz click para ver: http://127.0.0.1:8000/



Funcionó.

- La etiqueta más básica, `<html>` , es siempre el principio de cualquier página web y `</html>` es siempre el final. Como puedes ver, todo el contenido de la página web va desde el principio de la etiqueta `<html>` y hasta la etiqueta de cierre `</html>`
- `<p>` es una etiqueta para los elementos de párrafo; `</p>` cierra cada párrafo

Encabezado y cuerpo

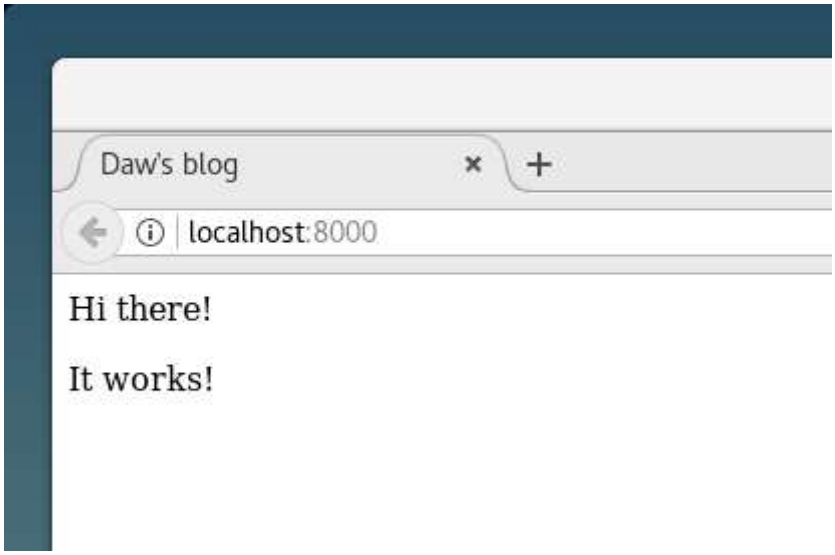
Cada página HTML también se divide en dos elementos: **head** y **body**.

- **head** es un elemento que contiene información sobre el documento que no se muestra en la pantalla.
- **body** es un elemento que contiene todo lo que se muestra como parte de la página web. Usamos para decirle al navegador acerca de la configuración de la página y para decir lo que realmente está en la página.

Por ejemplo, puedes ponerle un título a la página web dentro de la `<head>` , así:

```
<html>
    <head>
        <title>DAW's blog</title>
    </head>
    <body>
        <p>Hi there!</p>
        <p>It works!</p>
    </body>
</html>
```

Guarda el archivo y actualiza tu página.



¿Observas cómo el navegador ha comprendido que “DAW’s blog” es el título de tu página? Ha interpretado `<title>DAW's blog</title>` y colocó el texto en la barra de título de tu navegador (también se utilizará para marcadores y así sucesivamente).

Probablemente también hayas notado que cada etiqueta de apertura coincide con una *etiqueta de cierre*, con un `/`, y que los elementos son *anidados* (es decir, no puedes cerrar una etiqueta particular hasta que todos los que estaban en su interior se hayan cerrado también).

Es como poner cosas en cajas. Tienes una caja grande, `<html></html>`; en su interior hay `<body></body>`, y que contiene las cajas aún más pequeñas: `<p></p>`.

Tienes que seguir estas reglas de etiquetas de *cierre* y de *anidación* de elementos – si no lo haces, el navegador puede no ser capaz de interpretarlos correctamente y tu página se mostrará de manera incorrecta.

Personaliza tu plantilla

Ahora, puedes divertirte un poco y tratar de personalizar tu plantilla. Aquí hay algunas etiquetas útiles para eso:

- `<h1>Un título</h1>` para tu título más importante
- `<h2>Un subtítulo</h2>` para el título del siguiente nivel
- `<h3>Un subsubtítulo</h3>` … y así hasta `<h6>`
- `<em>texto</em>` pone en cursiva tu texto
- `<strong>texto</strong>` pone en negrita tu texto
- `<br />` un salto de línea (no puedes colocar nada dentro de br)
- `<a href="https://localhost:8000">link</a>` crea un vínculo

- `<ul><li>primer elemento</li><li>segundo elemento</li></ul>` crea una lista, ¡igual que esta!
- `<div></div>` define una sección de la página

Aquí hay un ejemplo de una plantilla completa, copia y pega en `blog/templates/blog/post_list.html` :

```
    <html>
        <head>
            <title>Django DAW blog</title>
        </head>
        <body>
            <div>
                <h1><a href="">Django DAW Blog</a></h1>
             </div>
            <div>
                <p>published: 21.05.2017, 08:58</p>
                <h2><a href="">My first post</a></h2>
                <p>Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis
vestibulum. Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac
cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet
risus.</p>
            </div>
            <div>
                <p>published: 21.05.2017, 08:59</p>
                <h2><a href="">My second post</a></h2>
                <p>Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis
vestibulum. Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac
cursus commodo, tortor mauris condimentum nibh, ut f.</p>
            </div>
        </body>
    </html>
```

Aquí hemos creado tres secciones `div` .

- El primer elemento `div` contiene el título de nuestro blog – es un encabezado y un enlace
- Otros dos elementos `div` contienen nuestros blogposts con la fecha de publicación, `h2` con un título que es clickeable y dos `p` (párrafo) de texto, uno para la fecha y uno para nuestro blogpost.

Nos da este efecto:

Pero hasta el momento, nuestra plantilla sólo muestra exactamente la **misma** información – considerando que antes hablábamos de plantillas que nos permitían mostrar información **diferente** en el **mismo formato**.

Lo que queremos realmente es mostrar posts reales añadidos en nuestra página de administración de Django - y ahí es a donde vamos con la siguiente práctica.

Una cosa más: ¡despliega!

Sería bueno ver todo esto disponible en Internet. Hagamos otro despliegue en PythonAnywhere.

Haz un commit y sube tu código a GitHub

En primer lugar, veamos qué archivos han cambiado desde que hicimos el despliegue por última vez (corre estos comando de manera local, no en PythonAnywhere):

```
$ git status
```

Asegúrate de que estás en el directorio `django-daw` y vamos a decirle a `git` que incluya todos los cambios dentro de este directorio:

```
$ git add --all .
```

> `--all` significa que `git` también reconocerá si has eliminado archivos (por defecto, sólo reconoce archivos nuevos/modificados). También recuerda (de la práctica 3) qué `.` significa el directorio actual.

Antes de que subamos todos los archivos, vamos a ver qué es lo que `git` subirá (todos los archivos que `git` cargará deberían aparecer en verde):

```
$ git status
```

Ya casi acabamos, ahora es tiempo de decirle que guarde este cambio en su historial. Vamos a darle un “mensaje de commit” donde describimos lo que hemos cambiado. Puedes escribir cualquier cosa que te gustaría en esta etapa, pero es útil escribir algo descriptivo para que puedas recordar lo que has hecho en el futuro.

```
$ git commit -m "Cambie el HTML para el sitio."
```

> Asegúrate de usar comillas dobles alrededor del mensaje de commit.

Una vez que hicimos esto, subimos (push) nuestros cambios a GitHub:

```
$ git push
```

Descarga tu nuevo código a PythonAnywhere y actualiza tu aplicación web

- Abre la página de consolas de PythonAnywhere y ve a tu consola Bash (o comienza una nueva). Luego, ejecuta:

```
$ cd ~/<tu-pythonanywhere-username>.pythonanywhere.com
$ git pull
[...]
```

Y mira cómo tu código se descarga. Si quieres comprobar que ya ha terminado, puedes ir a la pestaña **Files** y ver tu código en PythonAnywhere.

- Finalmente, dirígete a la pestaña Web y selecciona **Reload** en tu aplicación web.

Tu actualización debería estar en línea. Actualiza tu sitio web en el navegador. Ahora deberías poder ver tus cambios.