How Do We Serve Static Files?

In this lesson, we will study how we can host the static files present in our Flask application.

WE'LL COVER THE FOLLOWING

What are static files?

Static files, or assets, are the files that the server sends to the clients *"as-it-is"*, i.e., without any intervention. For example, any `CSS` files, background **images**, or `JavaScript` files that we might have on our website are sent to the client without modification.

Steps to Serve Static Files

The following steps need to be taken to host a static asset.

Create a `\static` directory

First, similar to the templates, we need to create a directory called `\static` in our project for `Flask` to recognize it. The *same file structure strategies* that apply to the *templates* will also apply here. You can recall them from the "How to Use HTML Templates" lesson.

Create a URL for static files

In the case of a template, we created the `URL` through the views and the routes

In the case of a template, we created the `URL` through the views and the routes associated with them. However, we do not use views for static files. So, to

create a separate `URL` for them, we use the `url_for()` function.

`url_for()` function

The `url_for()` function is used to build a `URL` for a certain endpoint. We can use it to build `URL`s for view functions as well. It takes the name of the endpoint and any associated variables as arguments. **For example:** The following line will generate the `URL` for the defined view.

```
url_for('view_function_name', variable_name = 'value_of_variable')
```

> **Why do we need to use `url_for()` when we can hardcode the `URL`?**
>
> 1. The `url_for()` function always returns the absolute `URL` for the endpoint. Thus, we call it anywhere in the project, and we will not have issues with relative paths.
> 2. If we end up changing the path for an endpoint or the route for a view, we do not have to change the hardcoded `URL` in the whole project.

The `URL` for a static file can be created using the `static` endpoint as following:

```
url_for('static', filename = 'name_of_file')
```

Example application

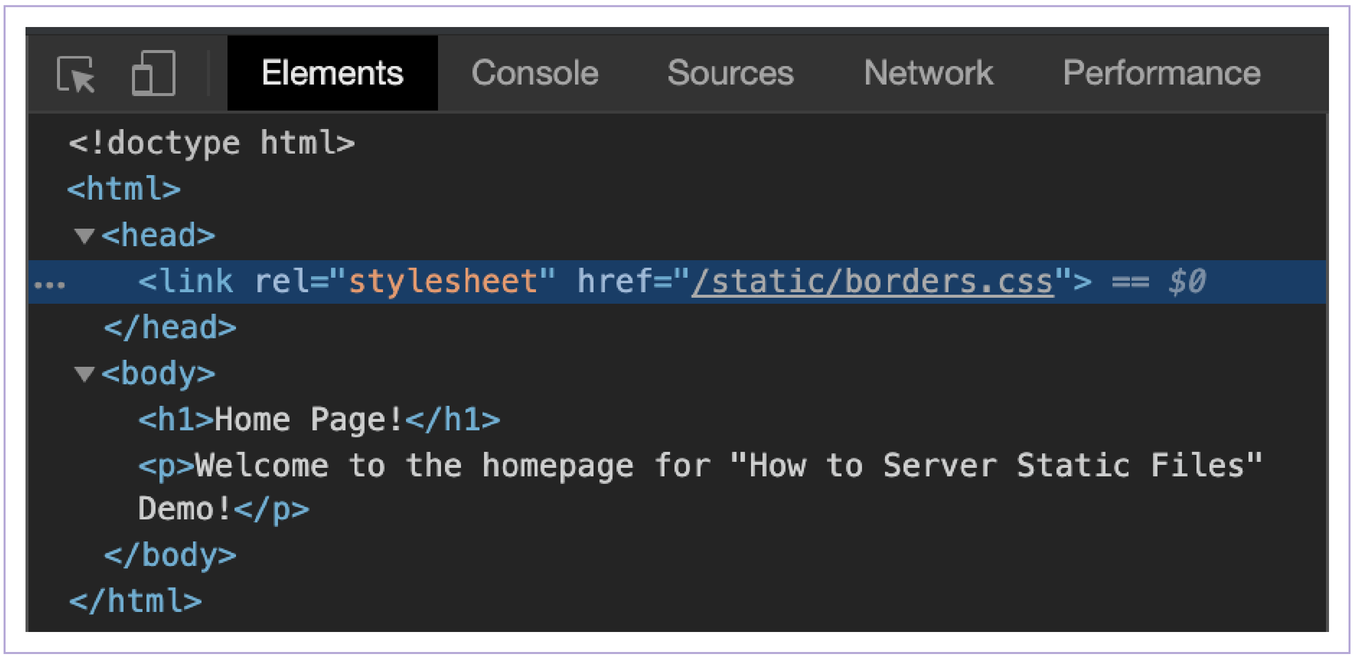
```
body {
    border: 5px solid grey;
    padding: 5px;
}

h1 {
    text-align: center;
}

p {
    border: 2px solid;
    padding: 5px;
    margin: 5px;
}
```

Explanation

In the code snippet given above, we linked the `CSS` stylesheet with `home.html` using the `<link>` and `</link>` tags. Instead of a relative path in the `href` attribute, we used `url_for()` to generate an absolute path for the static file.

> **You must be wondering: "what are the double braces `{{ }}` around the function?"**
> These braces are actually part of **Jinja's** syntax! We will learn about them in detail in the next chapter. For now, all you need to know is that whatever Python variable or function call we place inside them, its value will be rendered in the template when rendered on the client-side.

You can inspect the source code of the web page under Developer Tools in your browser to see that the absolute path to the `borders.css` has replaced the **Jinja** syntax. In fact, this is the `URL` that the function returned.



```
<!doctype html>
<html>
 ▼<head>
...    <link rel="stylesheet" href="/static/borders.css"> == $0
   </head>
 ▼<body>
     <h1>Home Page!</h1>
     <p>Welcome to the homepage for "How to Server Static Files"
     Demo!</p>
   </body>
</html>
```

Source code in Developer Tools

In the next lesson, we will solve a challenge related to static files. Stay tuned!