Bootstrap Tutorial

Welcome to the Bootstrap part of this tutorial! This part will add some jazz to your page, take you through how you import Bootstrap files and tools, and how to add them to your html pages, as well as some general tips for how to understand html pages. To access it switch to the folder Bootstrap of this tutorial.

For now we have created a basic website with separate HTML pages, one for each of the pages wanted in the website, all of which are interconnected directed to through the main.py file. This can happen through buttons or forms.

However, now we can forget about the main.py. To use Bootstrap, our intuitive and nice looking website tool, we have to understand html pages further, as the main.py does not have to change whatsoever!

NOTE: to run this part of the tutorial, make sure you are in FlaskTutorial/bootstrap. Then you will simply have to run << python3 main.py >> (or whatever python version you have) and you will get a link to go to in your browser.

Contents

1. How to import Bootstrap
2. Html pages and fields
3. Base page
4. CSS classes

# HOW TO IMPORT BOOTSTRAP

The first thing to use Bootstrap is to import and download the right files. This work has already been done for you in this tutorial, but before showing you where the files are hiding we will go through how to download them.

Bootstrap works by using css definitions of specific classes. There are many classes for buttons, columns, titles, everything you can think of. These css classes have parameters such as width, height, padding, colour, and many many more. The idea is that once you have downloaded the style files, you can use the classes defined within them to easily make your html prettier and colourful.

In a world without this tutorial you would go to the website <https://getbootstrap.com/docs/4.3/getting-started/download/> and download all files. Once downloaded, you would simply have to copy the inside folders of css and js, make a new directory in your project directory called ‘static’ and paste the css and js folders there. If you look at base\_full.html, the header section contains links and scripts referring to the bootstrap files and includes a SOURCE PATH – this is important! If you don’t add the right relative path, the bootstrap files won’t be imported (if this happens the message 404 will appear on your terminal). So we addd the paths and done!

# HTML PAGES AND FIELDS

To understand what Bootstrap does we have to understand HTML pages. The following is a quick description of what the sections/fields in HTML pages mean.

A main part of the html page is the <head> section, because that’s where we establish the style and script that will be used for the styling of the page. The style and script html fields are the ones we need to use bootstrap!

As you can see, we can also set the title for the page in the head section, as well as any icon we want the website to have. In the static folder you have some fun images to try.

Have a look at the file base\_try.html in the templates folder of this part. The head part has already been filled in to import the bootstrap files that have already been downloaded into the static folder. Inside the header lines we see tags such as “src” and “charset” as well as classes such as “title”, “link”, and “script”. The src for example, whenever it is used, indicates the path to where the code or image can be found.

Below you will see the <body> section, which is where the main content of the page will reside. Inside this you will see curly brackets {% block content %} that will be explained further on.

The important thing here is to realize we have three sections – the head, body and footer, and many classes and tags that we can add in these sections.

If you take a quick look at the file base\_full.html page, you will see what this page will eventually look like.

# base page

This section will also demonstrate a very useful Flask tool, the use of a BASE html page. Instead of creating many different html pages with the same settings, for example, all pages must have the same navigation bar and a search bar, we create a base html page that acts as the predetermined basic html page. All other pages will have this as a base, and will have an easier time adding their personalized changes, reducing the amount of repeated code substantially.

To have a look at the finished product, look at the html page base\_full.html. You will see we have included a navigation bar as well as a footer section.

So what does the base page have? Our very important <head> section to import the bootstrap files, the icon we wish to use for our page, the <body> section that will eventually contain the individual pages, and a <footer> section.

And how do we use the base page? It is surprisingly easy! Take a look at the index.html page. This is a very simple HTML page, and we don’t like it. Now take a look at the index\_full.html page. The main things to note are the additions of {% extends “base\_full.html” %} and the {% block content %} and {% endblock %}.

What these simple html lines do is firstly, extend the base\_full.html page (or whatever name you make your file) and secondly, they two last lines mentioned encircle the body of the individual html page. This section is what will fill in the section in the base.html that is named the same. And that’s how we use the base html page!

# CSS CLASSES

Now the real Bootstrap starts – we’ve downloaded it and imported it in the header section and seen the base html page. But how do we make everything look good? As mentioned before the way to do that is to play with the predetermined css classes. For example, let’s look at some different buttons! For this, run the website and add /buttons to your url to see some button options, or click the first button of the index page.

If you go to look at this html page, buttons.html, you will see that the only things that changed to make the headings and buttons different was the “class” tag of the html object. To see what these mean, take a look at the file static/css/bootstrap.css. Here, if you can use Ctrl-F to find classes – try searching ‘btn’ and look through the options. There are lots of different classes and each has its own style settings. There are so many for every type of object that you’ll usually have what you need.

Fun fact: you can make your own! Let’s try making our own class. In the style section of the header you can see some style options for the input type test. Below the last button, add the following line to see how to use your own class:

<input *type*="test" *value*="New Button" />

Go to the style section to change all the padding settings to experiment how changing the class can change everything.