COURSEWARE

Introduction to frameworks and Flask

Contents

Contents

Overview

In this module, we will introduce the idea of a web framework and introduce a micro-framework called Flask.

What is a framework?

A framework is a structure, it acts as skeletal support used as the basis for something being constructed. A web framework acts as such for the development of web applications.

Web frameworks are pre-written code which enables the developer to design the objects, structure and interactions of their web application. There are many web frameworks available for different programming languages and needs.

Flask

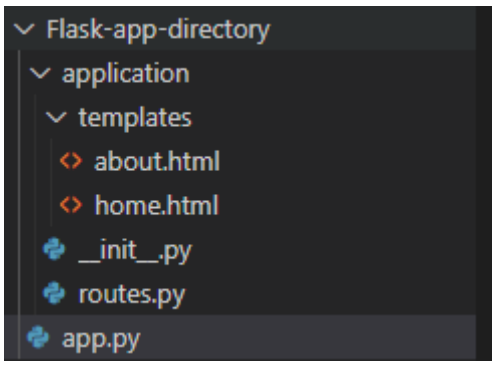
Flask is a Python driven `micro-framework`. 'micro' indicates that Flask aims to keep the core simple but extensible. This gives us the means to create very simple apps written in just a few lines of code as well as complex enterprise-level applications.

Added functionality such as forms, database interactions and secrets handling is enabled in Flask using extensions, which are modules that can be imported as needed.

Structure

It is important to adhere to a clear and sensible structure so that as it becomes more complex, it remains coherent for debugging and implementing stages.

A typical app structure looks something like this:



`app.py` is the main file (the one that runs to start the app).

`__init__.py` contains various objects that the app will use.

The `application` folder will contain the Python files with the app's code.

The `templates` folder contains the HTML files for each page of the web application.

Installation

Flask can be easily installed using pip by running the command:

```
pip3 install flask
```

Tutorial

In this tutorial, we will install Flask and create a basic app that displays a message saying `Hello internet!`.

This tutorial assumes you are working on an Ubuntu machine, either `18.04 LTS` or `20.04 LTS`.

Setup

First, run the following to install the necessary Python requirements:

```
sudo apt update
sudo apt install python3 python3-venv python3-pip
```

Create a directory named `flask-introduction` and make it your current working directory:

```
mkdir flask-introduction && cd $_
```

We now need to create a Python virtual environment to install our pip requirements in. Create a new virtual environment named `venv` and activate it:

```
python3 -m venv venv
source venv/bin/activate
```

Your terminal should now look something like this:

```
(venv) my-user@my-machine:~/flask-introduction$
```

Install Flask with pip:

```
pip3 install flask
```

With our dependencies installed, we are now ready to create our Flask application.

Creating the App

Create a Python file named `app.py`:

```
touch app.py
```

Using a text editor of your choice, enter the following into `app.py`:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_internet():
    return "Hello Internet!"

if __name__=='__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

This is a Flask app! Because Flask is a *microframework*, it requires very little configuration to create – in this case, only 7 lines of code! Let's break down what's happening here.

The first line imports the Flask module, which is what will enable us to instantiate the Flask app.

```
app = Flask(__name__)
```

The above line is where the Flask app object is created. The methods and attributes of this object are the internal operations that result in a working application.

This line of code provides some functionality to the Flask app in the form of a `route`:

```
@app.route('/')
def hello_internet():
    return "Hello Internet!"
```

It provides a URL location for HTTP requests to be sent to, and describes the code that should be run when the request is received. Here we are returning the phrase `Hello Internet!`, which will be displayed on our browser.

Finally, this block of code is what allows us to run the app by running `app.py` from the command line.

```
if __name__=='__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

The `if` statement checks to see if this module is the one that is being run by the Python interpreter or not. If this module were being imported by another, this `if` statement prevents Flask from running unnecessarily.

Running the App

Run the app using this command:

```
python3 app.py
```

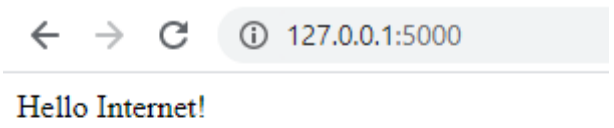
Your terminal should display an output that looks like this:

```
* Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production
deployment.
 * Running on http://10.0.0.4:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 699-679-452
```

We're now going to access the application via your browser. Find your machine's public IP address and copy it into your web browser with `:5000` at the end (e.g. `51.42.6.23:5000`) to access the app on port `5000`.

Note: you may have to allow incoming network traffic on port 5000.

You should see `Hello Internet!` appear in your browser like so:



Clean Up

To stop your Flask application running, navigate back to your terminal and press `Ctrl+C`. You should now have control over your terminal again.

To deactivate the virtual environment, run:

```
deactivate
```

If you wish to delete the virtual environment, run:

```
rm -rf venv
```

Exercises

There are no exercises for this module.