CODING NOMADS

5) Build Your First Flask Apps Lesson

How to Build a Flask Hello World App

13 min to complete · By Brandon Gigous

Contents

Now that you installed Flask in the previous section, things are starting to get interesting! It's time to start developing your first Flask app. In this lesson, you'll see step-by-step how to build a Flask hello world application.

Flask Hello World App Setup

- Run VS Code and go to `File -> Open Folder`
- Navigate to your `flask-webdev` repository and open it up!

- Bring up VS Code's handy little terminal, which you can open from `View -> Terminal` or by pressing `Ctrl + ` `, or `Cmd + ` ` for Mac.
- If the terminal's working directory isn't already your `flask-webdev` directory, change it.
- Make sure your virtual environment is activated.
- Then in VS Code, make a new file called `hello.py`. Now let's get those fingers moving with some code!

Note: This is one more friendly reminder that you'll be developing in the `flask-webdev`` directory from now on, unless otherwise specified. Assume that you'll have the steps above completed every time you come back for more Flask development.

Alert: If you have multiple virtual environments you're juggling these days, make sure you have the correct one activated. You can issue the command `deactivate` in the CLI to close the session you currently have open in order to activate a new one.

Write the Code

In your new `hello.py` file, just type these code blocks in order. Here goes!

Initialization

```
from flask import Flask

app = Flask(__name__)
```

The first couple of lines imports the Flask class from the flask module, and then a variable `app` is created as an instance of a Flask object. To make such a Flask instance, the `Flask` class wants the name of the main module or package of the application. Generally, `__name__` passed in as an argument is the correct value, and Flask wants it so

it can find all the templates and images you make for it to render! (Which will come later ;))

Routing

```
@app.route('/')
def index():
    return "Hello Web World!"
```

This one is fun, because decorators are fun. The `app.route` decorator here, with `'/'` as its argument, translates to "whenever someone visits the root URL, call this `index()` function to return a response."

Put another way, the `index()` function is a **handler** for the root URL. If the application is deployed at *www.example.com*, navigating to https://www.example.com/ with nothing else would trigger the `index()` function to be called and run on the server. Routes will be covered a little more later.

What about app.run()?

You now have your 7-or-so line `hello.py` file and I'm sure you're wondering what to do with it and how to get it running as a web app. It's not quite as simple as using an online development tool like Repl.it, since now you're in a real web development scenario, but this Python Flask tutorial will get you on your way!

You may even be thinking, "Hey wait a sec, didn't you leave something out in that file? That `app.run()` thing?" (Thinking in code, I love it!)

Ah, you'd be right. There was such a `app.run()` function in the example a couple sections ago, but you don't actually need it this time. Isadora might be a fortune teller, but can she see why developers love web development using Flask?

Cinnamon Toast Crunch Commercial - Fortune Teller (2002)

Set the Flask App Environment Variable

Flask can get applications up and running lickety-split (as you've seen), and Flask makes it *convenient* to develop with. You truly have a working app inside `hello.py`, but for now it is just text. Even running `python hello.py` will do seemingly nothing.

Flask has its very own development web server hidden inside, all waiting to come out with a simple command! Not even a fortune teller could've figured that out at first glance.

To be able to kick off this web server, all that Flask needs from you is for you to set the `FLASK_APP` environment variable. This setting lets Flask know the name of the file where the application instance lives. In other words, the file specified is one you want to test, debug, or otherwise just appreciate.

With that said, in your CLI, type and execute the following (keeping the spacing as it is):

```
(env) $ export FLASK_APP=hello.py
```

Run your Flask Hello World App

Now hold your breath, you're about to launch your web app as if by magic. Next, type in these two words and imagine they have the same effect as the words "hocus pocus":

```
(env) $ flask run
 * Serving Flask app "hello.py"
 ...
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

If you see output similar to the above, your magic trick... er, the command worked! Open up your favorite browser and navigate to `localhost:5000` and you'll be greeted with a "Hello Web World!"

Run Flask for Development Mode

You might see an ominous warning regarding production deployment; that's because Flask by default runs in "production" mode. However, that's not what should usually be used for development. Instead, you can set another environment variable to run your Python Flask app in development mode:

```
(env) $ export FLASK_ENV=development
```

Enable Debug Mode

With that set, the next time you try `flask run`, it will show as running in a development environment. This will also automatically enable debug mode. Regardless of the environment, debug mode can be set separately with:

```
(env) $ export FLASK_DEBUG=1
```

With debug mode enabled, you'll get to utilize two very convenient modules:

- **Reloader** - It's pretty nifty. With this, Flask will scan all your source files for changes and will automatically restart the server if it detects that any of them have been modified. That means whenever you have a server running and you modify a line or two in a file and save it, you'll see from your terminal that your server will restart, which means it has already accounted for the changes you made. Told ya it was nifty!
- **Debugger** - Whenever one of those changes might break your web app, Flask has this other cool feature that will display whatever error it encountered *right in your webpage*. That is, instead of an ugly page that gives you nothing to go by to fix the error. Even more swell is the fact that you can interact with this page and see the source code that may have caused the problem.

Alert: All this boring command line stuff is actually pretty important. Flask's debug modes even screwed me up when I was developing! So if it helps you, definitely write some of this down to commit it to memory. It may be helpful to bookmark this page, too.

You can also get some documentation for Flask commands with `--help`, for example, `flask --help` or `flask run --help`

Note: While you *can* still use `app.run()` for bringing up your Flask app, keep in mind that setting `FLASK_APP`, `FLASK_DEBUG`, and `FLASK_ENV` environment variables won't have any effect if you go that route (pun intended).

Summary: How to Build a Flask Hello World App

1. Write the Flask hello world code

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "Hello Web World!"
```

2. Run the Flask app in your terminal

```
(env) $ flask run
```

3. Enable development and debugger mode

```
(env) $ export FLASK_ENV=development
```

```
(env) $ export FLASK_DEBUG=1
```

Now that you've learned the nitty-gritty of development using Flask, and since you already have the grit it takes to be a Flask developer, you're set to keep going with your app. But first, an existential question... what are URLs, anyway? Click on to the next lesson to find out!

Previous Next → What are URLs and Routes?