LECTURE 31

HTTP AND FLASK

MCS 275 Spring 2023

David Dumas

LECTURE 31: HTTP AND FLASK

Reminders and announcements:

- Project 3 graded
- Project 4 announced
- Please install **Flask**, e.g. with

```
python3 -m pip install Flask
```

  in preparation for using it in upcoming assignments.

MOCKUPS

First, let's check in on page mockups for the work order app.

Reminder: You can always get these from the sample code repository. You can also browse history of previous versions there.

CSS: WHAT TO KNOW

- Selectors
  - by tag name (e.g. `div { ... }`)
  - by class (e.g. `p.intro { ... }`)
- Attributes
  - font-size
  - font-family
  - color
  - background

FILE PROTOCOL

So far, I've been opening files in the web browser, using URLs with the `file` protocol.

There's no network communication here. The browser just opens the file using the OS interface.

To make an actual web site or application, we need an HTTP server.

PYTHON'S BUILT-IN HTTP SERVER

```
python3 -m http.server
```

Opens a web server that serves files in the current directory and its subdirectories.

Visit `http://localhost:8000/` in a browser (or substitute other port number shown in startup message) to see `index.html`.

Firewall rules typically prevent incoming connections from the internet (and maybe the local network too). That's good! Or

```
python3 -m http.server --bind 127.0.0.1
```

INDEX.HTML

Most HTTP servers that deliver resources from a filesystem will look for a file called `index.html` and send it in response to a request that ends in a `/`.

(i.e. if no filename is given, `index.html` is used.)

FLASK

Flask is a Python **web framework**. It makes it easy to write Python programs that respond to HTTP requests (e.g. web applications, APIs).

Competitors include:

- Bottle — minimalist like Flask
- Django — huge and full-featured

MINIMAL FLASK

```
from flask import Flask

app = Flask(__name__)

@app.route("/positivity/")  # URL (public name for this action
def message():              # Logical name for this action
    return """<!doctype html>
    <html>
        <body>
            You can do it!
        </body>
    </html>
    """

app.run()
```

STATIC FILES

A flask app may not generate all the content it serves; some things (like CSS files or images) may be prepared in advance and used unchanged.

If you create a subdirectory `static` of the directory where your Flask application runs, you can put such "static" files there.

Refer to files there using URLs like `static/image.png`.

TEMPLATES

You can make an HTML file in which some parts are to be replaced by values of variables.

Surround the parts to be replaced by `{{` and `}}`.

Put these in subdirectory `templates` and then call `flask.render_template` to generate the HTML as a return value.

Pass the variables to be given to the template as kwargs.

```
@app.route("/foo")
def bar():
    # Find "templates/david.html"
    # In it, replace {{ x }} with 12 and {{ y }} with 15
    # then serve the result as the HTML for this route
    return render_template("david.html",x=12,y=15)
```

REFERENCES

- jsfiddle - Write and test HTML+CSS quickly in browser
- HTML tutorial from w3schools
- CSS tutorial from w3schools
- The Flask tutorial

REVISION HISTORY

- 2022-04-07 Last year's lecture on this topic finalized
- 2023-04-05 Updated for 2023