Lab 2: Python Flask

GW CS 2541: Database Systems and Team Projects - 2022
Prof. Tim Wood, Ethan Baron, and Catherine Meadows

Front End vs Backend: What is the difference?

Front End vs Back End

Front End:

- The art and design of websites and web applications that render on the client side. Everything from the look and feel to the way you interact with a website.

Back End:

- The server side logic for an application controlling what happens with the data, how the client side rendering changes in response and how the data gets stored.

Full Stack:

- A developer that can work on both Front and Back End software

Python Flask

Flask is a Python “Web framework”

- Library to make it easier to write a web application
- Examples: Flask, Django, react.js, vue.js, Angular, Ruby on Rails, Drupal…

How is this different from a “web server”?

API for defining backend services

- Handles form input/output, cookie management, session data, DB connections, overall page formatting, etc

Flask is a microframework

- Provides a minimal set of functionality (with extras as needed)

Flask Hello World

Load the Flask library and setup your app

Define a **Route** and specify what is returned when we access it

Tell the web application to actually start running

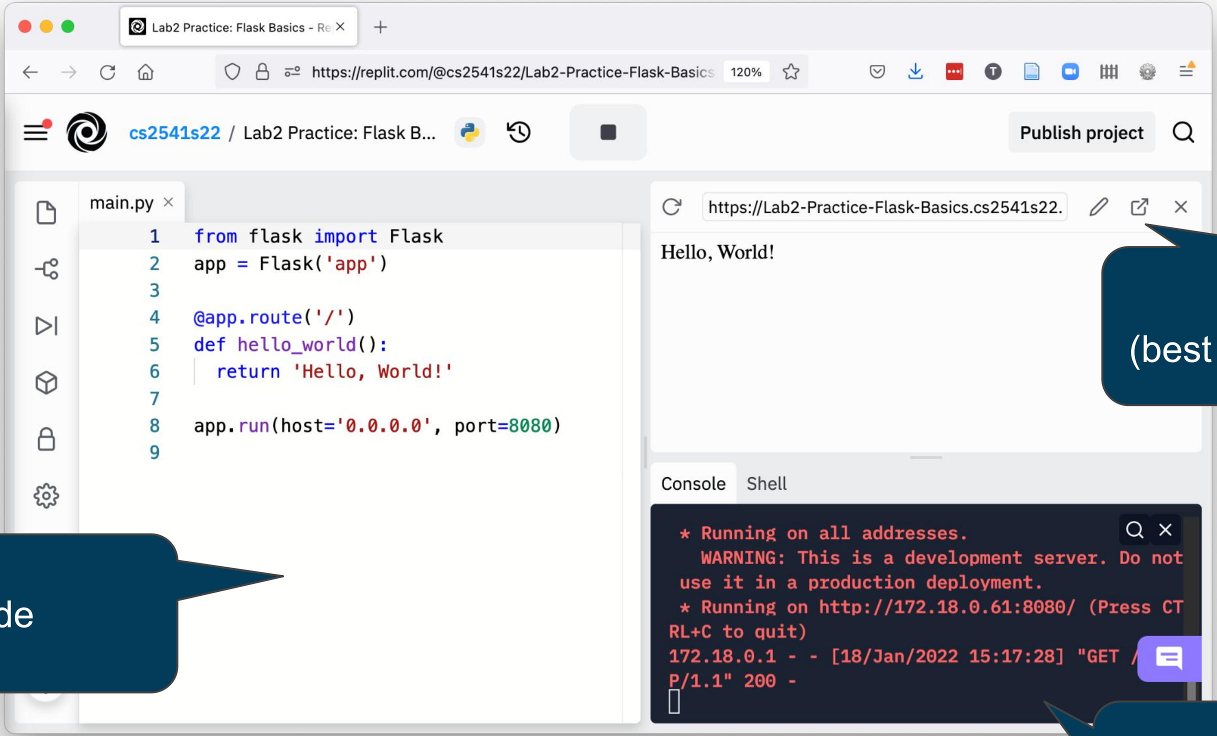
```
from flask import Flask
app = Flask('app')


@app.route('/')
def hello_world():
 return 'Hello, World!'


app.run(host='0.0.0.0', port=8080)
```

Be careful about copy/paste from slides!

Flask on Replit.com


Lab2 Practice: Flask Basics - Re
https://replit.com/@cs2541s22/Lab2-Practice-Flask-Basics
120%
cs2541s22 / Lab2 Practice: Flask B...
Publish project
main.py
https://Lab2-Practice-Flask-Basics.cs2541s22.
Console
Shell

```python
from flask import Flask
app = Flask('app')

@app.route('/')
def hello_world():
  return 'Hello, World!'

app.run(host='0.0.0.0', port=8080)
```

Hello, World!

```
 * Running on all addresses.
   WARNING: This is a development server. Do not
 use it in a production deployment.
 * Running on http://172.18.0.61:8080/ (Press CT
RL+C to quit)
172.18.0.1 - - [18/Jan/2022 15:17:28] "GET /
P/1.1" 200 -
```

App preview (best to open in new tab)

Write code

Log messages - print() calls will go here

Routes + Templates = Flask

Routes

- A backend service endpoint URL
- Function to be called when route is accessed

Templates

- Defines front end appearance of website
- Interacts with back end to allow data to be filled in

Both are good examples of our goal of Abstraction!

- Flask helps us separate design of different backend services and cleanly separates front end from back end

(Other stuff too, but this will get you most of the way to a good app!)

Routes Basics

Routes are functions:

- Specify the URL to access them
- Define the behavior to execute
- Return the content that should be displayed to the user

`@app.route` is a Python Decorator

- Special syntax to make a wrapper function. See [1] for details

```
from flask import Flask
app = Flask('app')


@app.route('/two')
def hello_world():
 Return '<html><body>This route has valid
HTML, but both will display!</body></html>'


@app.route('/')
def hello_world():
 return 'Hello, World from the root route!'


app.run(host='0.0.0.0', port=8080)
```

[1] https://realpython.com/primer-on-python-decorators/

Templating Basics: Passing Variables

/templates/index.html

Templates MUST be in a folder /templates

```
<html>
    <head>
        <title> {{ title }} </title>
    </head>
    <body>
        <h1> Hello {{ username }}</h1>
    <body>
</html>
```

Import module for rendering templates

```
from flask import Flask
from flask import render_template
app = Flask('app')

 @app.route('/index')
 def index():
     name = ‘Cat Meadows’
     return render_template(‘index.html’,
          title = ‘Welcome’,username=name)

 app.run(host='0.0.0.0', port=8080)
```

Demo: Hello X and Hello Y

Reuse:

- Template can be used in multiple routes
- Each route can fill different data into the template

```
@app.route('/')
def hello_world():
 name = 'Cat Meadows'
 return render_template('hello.html', title =
'Welcome', username=name)

@app.route('/helloTim')
def hello_world():
 name = 'Tim Wood'
 return render_template('hello.html', title = 'Welcome
2', username=name)
```

Demo:
https://replit.com/team/cs2541s22/Lab2-Practice-Template-Data

Routes with Parameters

We can extract data from the URL

- Parameters are available as python variables
- Flask lets you enforce types, have multiple parameters, etc

Modify the route definition and add the parameters as arguments to your function

What would we see if we visit /parameters/Ethan ?

```
from flask import Flask
app = Flask('app')

@app.route('/parameters/<name>')
def hello_name(name):
 return 'Hello, ' + name

app.run(host='0.0.0.0', port=8080)
```

Activity 1: Hello/XYZ

5 minutes!

Our earlier code was dumb!

- Repetitive routes that are just different based on the incoming data

Group Task:

- Make a single route which can say "hello **XYZ**" based on the URL data
- Must use template

https://replit.com/team/cs2541s22/Lab2-Practice-Template-Data

```
@app.route('/')
def hello_world():
 name = 'Cat Meadows'
 return render_template('hello.html', title =
'Welcome', username=name)

@app.route('/helloTim')
def hello_world():
 name = 'Tim Wood'
 return render_template('hello.html', title = 'Welcome
2', username=name)
```

/hello/Tim -> "Hello Tim!"
/hello/Ethan -> "Hello Ethan!"
etc...

Templating Basics: if

```
<html>
    <head>
        <title> {{ title }} </title>
    </head>
    <body>
        {% if username == "Cat Meadows": %}
        <h1> Hello, UTA!! </h1>
        {% else %}
        <h1> Hello, {{ username }} </h1>
        {% endif %}
    <body>
</html>
```

Syntax {% … %}

Always close conditionals with **{% endif %}**

Templating Basics: for loop

```
<html>
    <head>
        <title> {{ title }} </title>
    </head>
    <body>
        <ul>
        {% for user in users: %}
        <li> {{ user }} </li>
        {% endfor %}
        </ul>
    <body>
</html>
```

Always close for loops with **{% endfor %}**

```
from flask import Flask
from flask import render_template
app = Flask('app')


 @app.route('/index')
 def index():
     users = ['Tim', 'Ethan', 'Cat']
     return render_template('index.html',
          title = 'Welcome',users=users)


 app.run(host='0.0.0.0', port=8080)
```

More Template Syntax

Demo: "Lab-2-Practice-Template-Syntax" replit

Learn more:

- https://realpython.com/primer-on-jinja-templating/
- https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-ii-templates

Activity 2: Class Roster

1. Create a class roster **nested dictionary** in the format:

```
Tim = {
      “Name”: “Tim Wood”
      “SID”: “G12345678”
      “Engagement”: 2
 }
 #... define more students here …
roster = {
      “Tim” : Tim,
      “Ethan” : Ethan,
      “Cat” : Cat
 }
```

2. Use templates and routes with parameters to display:

- First route: list of students by name at the "/" index route
    - Each name should be a link to the second route
- Second route: display name, SID, and engagement points for the student

Note: You should only use **two** templates and **two** routes

Due TOMORROW 11:59PM

https://replit.com/team/cs2541s22/Lab2-Advanced-Routes-and-Templates

Formatting Examples

Emphasis box 1

Emphasis box 2

Emphasis box 3

Emphasis box 4

```
from flask import Flask
app = Flask('app')


@app.route('/')
def hello_world():
 return 'Hello, World!'

app.run(host='0.0.0.0', port=8080)
```