An Introduction to Web Development with Flask

Prudhvi Boyapalli

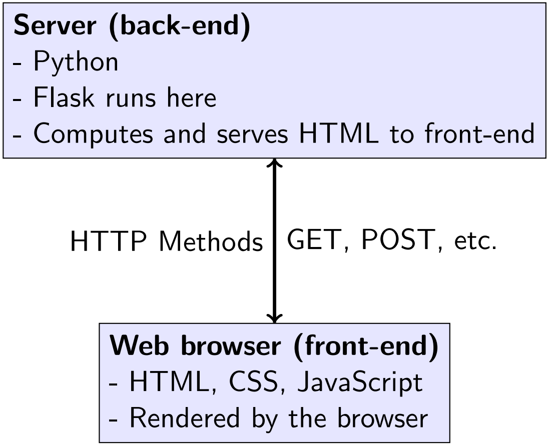
Rice Computer Science Club

Hack & Learn Workshop
January 13, 2017

Introduction

How does a web app work? What does Flask do?

Flask is a Python microframework for building web applications.



Necessary prior knowledge

- Basic familiarity with Python
  - If you've taken COMP140 you're in good shape
- Basic familiarity with HTML and JavaScript, CSS
  - Be able to read HTML and JavaScript syntax
  - CSS is optional

Install Python & Flask

- Install Python if you haven't already: `python.org`
  - I'm using 2.7, Flask also supports 3.3+
- Install Flask using PIP
  - **P**ip **I**nstalls **P**ackages from the **Py**thon **P**ackage **I**ndex a.k.a. PyPI

Flask Installation

```
> sudo pip install Flask
```

Editing

An IDE is overkill for this workshop. Use a programmer's editor that you're comfortable with:

- Vim (All platforms)
- Atom (All platforms)
- Visual Studio Code (All platforms)
- Sublime Text (All platforms)
- Notepad++ (Windows)
- TextMate (OSX)
- Whatever floats your boat

Debugging

You'll be testing and debugging your code from the browser, so use one with good developer tools:

- Chrome
- Firefox

Other browsers can be used, but Chrome or Firefox will minimize the chance of running into browser specific issues.

Hello World

Super basic hello world example

This is the bare minimum code needed on the back-end to serve a web page

./code/hello_world_example/hello_world.py

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

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Running the web app

Once we've written the flask code, we can run the server locally and view the results in our web browser

Execute these commands to run the code:

```
> cd /directory/with/project
> export FLASK_APP=hello_world.py
> flask run
```

Then you'll see something like:

```
* Serving Flask app "hello_world"
* Running on http://127.0.0.1:5000/
(Press CTRL+C to quit)
```

Running the web app

Once we've written the flask code, we can run the server locally and view the results in our web browser

Execute these commands to run the code:

```
> cd /directory/with/project
> export FLASK_APP=hello_world.py
> flask run
```

Then you'll see something like:

```
* Serving Flask app "hello_world"
* Running on http://127.0.0.1:5000/
(Press CTRL+C to quit)
```

Running the web app

Once we've written the flask code, we can run the server locally and view the results in our web browser

Execute these commands to run the code:

```
> cd /directory/with/project
> export FLASK_APP=hello_world.py
> flask run
```

Then you'll see something like:

```
* Serving Flask app "hello_world"
* Running on http://127.0.0.1:5000/
(Press CTRL+C to quit)
```

How did that work?

1. We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
2. We visited this page and the browser sent a request to our flask server
3. Because we visited `127.0.0.1:5000/` (emphasis on the `"/"`), the method bound to `"/"` was invoked
4. The `hello()` method in `hello_world.py` was executed and the return value is sent to the browser as HTML
5. The browser received the HTML and rendered the webpage

How did that work?

1. We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
2. We visited this page and the browser sent a request to our flask server
3. Because we visited `127.0.0.1:5000/` (emphasis on the `"/"`), the method bound to `"/"` was invoked
4. The `hello()` method in `hello_world.py` was executed and the return value is sent to the browser as HTML
5. The browser received the HTML and rendered the webpage

How did that work?

1. We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
2. We visited this page and the browser sent a request to our flask server
3. Because we visited `127.0.0.1:5000/` (emphasis on the `"/"`), the method bound to `"/"` was invoked
4. The `hello()` method in `hello_world.py` was executed and the return value is sent to the browser as HTML
5. The browser received the HTML and rendered the webpage

How did that work?

1. We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
2. We visited this page and the browser sent a request to our flask server
3. Because we visited `127.0.0.1:5000/` (emphasis on the `"/"`), the method bound to `"/"` was invoked
4. The `hello()` method in `hello_world.py` was executed and the return value is sent to the browser as HTML
5. The browser received the HTML and rendered the webpage

How did that work?

1. We started up a local flask server which is listening at IP address 127.0.0.1, port 5000
2. We visited this page and the browser sent a request to our flask server
3. Because we visited `127.0.0.1:5000/` (emphasis on the `"/"`), the method bound to `"/"` was invoked
4. The `hello()` method in `hello_world.py` was executed and the return value is sent to the browser as HTML
5. The browser received the HTML and rendered the webpage

Interactive hello world with variable routing

Now that we have a basic idea of how flask works, we can do some cool stuff with routing:

./code/hello_world_example/hello_user.py

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

@app.route("/world")
def hello():
    return "Hello World!"

@app.route("/user/<name>")
def hello_user(name):
    return "Hello %s, how are you today?" % name

if __name__ == "__main__":
    app.run()
```

Running the new and improved web app

As before:

```
> export FLASK_APP=hello_user.py
> flask run
* Serving Flask app "hello_user"
* Running on http://127.0.0.1:5000/
(Press CTRL+C to quit)
```

Hello World

```
http://127.0.0.1:5000/world
```

Hello <user>

```
http://127.0.0.1:5000/user/Foo
http://127.0.0.1:5000/user/Bar
```

Running the new and improved web app

As before:

```
> export FLASK_APP=hello_user.py
> flask run
* Serving Flask app "hello_user"
* Running on http://127.0.0.1:5000/
(Press CTRL+C to quit)
```

Hello World

```
http://127.0.0.1:5000/world
```

Hello <user>

```
http://127.0.0.1:5000/user/Foo
http://127.0.0.1:5000/user/Bar
```

List Sort

Conclusion

Summary & Next Steps

We covered:

- Basic & and variable routing
- `GET` and `POST` requests
- Accessing request data
- Building and serving a “complex” web app with HTML, CSS and JavaScript

Next Steps

More Flask Features

- URL Building
- Additional HTTP Methods (`GET`, `POST`, `DELETE`, etc.)
- HTML templates with placeholders
- User file uploads
- Reading and writing cookies
- Redirects and error handling
- Managing sessions

Deploying your web app

Once you've developed and tested you app locally, you can deploy your flask code and make it publicly accessible. There lots of tutorials on how to do this for various cloud providers (AWS, Azure, DigitalOcean, etc).

Next Steps

More Flask Features

- URL Building
- Additional HTTP Methods (`GET`, `POST`, `DELETE`, etc.)
- HTML templates with placeholders
- User file uploads
- Reading and writing cookies
- Redirects and error handling
- Managing sessions

Deploying your web app

Once you've developed and tested you app locally, you can deploy your flask code and make it publicly accessible. There lots of tutorials on how to do this for various cloud providers (AWS, Azure, DigitalOcean, etc).

Resources

- **This presentation & and all of the code**
  `github.com/hack-rice/flask-tutorial`
- **Flask website & awesome documentation**
  `flask.pocoo.org`

Contact

Prudhvi Boyapalli

- prb2 AT rice.edu

HackRice

- Fall 2017
- Annual hackathon right here on campus
- Great way to hone your skills and make something cool

RiceApps

- Work on a team of students to build useful applications
- Great way to gain hands on development experience
- `http://riceapps.org`

Rice CS Facebook Group

- `https://facebook.com/groups/365985373415471`

CS Club Facebook Page

- `https://facebook.com/ricecsclub`

Contact

Prudhvi Boyapalli

- prb2 AT rice.edu

HackRice

- Fall 2017
- Annual hackathon right here on campus
- Great way to hone your skills and make something cool

RiceApps

- Work on a team of students to build useful applications
- Great way to gain hands on development experience
- `http://riceapps.org`

Rice CS Facebook Group

- `https://facebook.com/groups/365985373415471`

CS Club Facebook Page

- `https://facebook.com/ricecsclub`

Contact

Prudhvi Boyapalli

- prb2 AT rice.edu

HackRice

- Fall 2017
- Annual hackathon right here on campus
- Great way to hone your skills and make something cool

RiceApps

- Work on a team of students to build useful applications
- Great way to gain hands on development experience
- `http://riceapps.org`

Rice CS Facebook Group

- `https://facebook.com/groups/365985373415471`

CS Club Facebook Page

- `https://facebook.com/ricecsclub`

Contact

Prudhvi Boyapalli

- prb2 AT rice.edu

HackRice

- Fall 2017
- Annual hackathon right here on campus
- Great way to hone your skills and make something cool

RiceApps

- Work on a team of students to build useful applications
- Great way to gain hands on development experience
- `http://riceapps.org`

Rice CS Facebook Group

- `https://facebook.com/groups/365985373415471`

CS Club Facebook Page

- `https://facebook.com/ricecsclub`