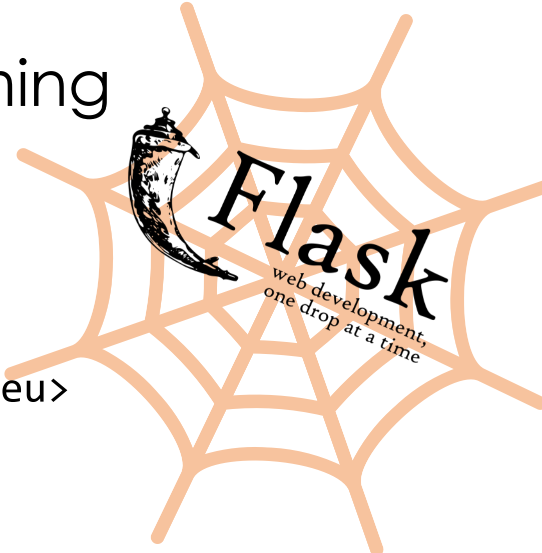
CS54

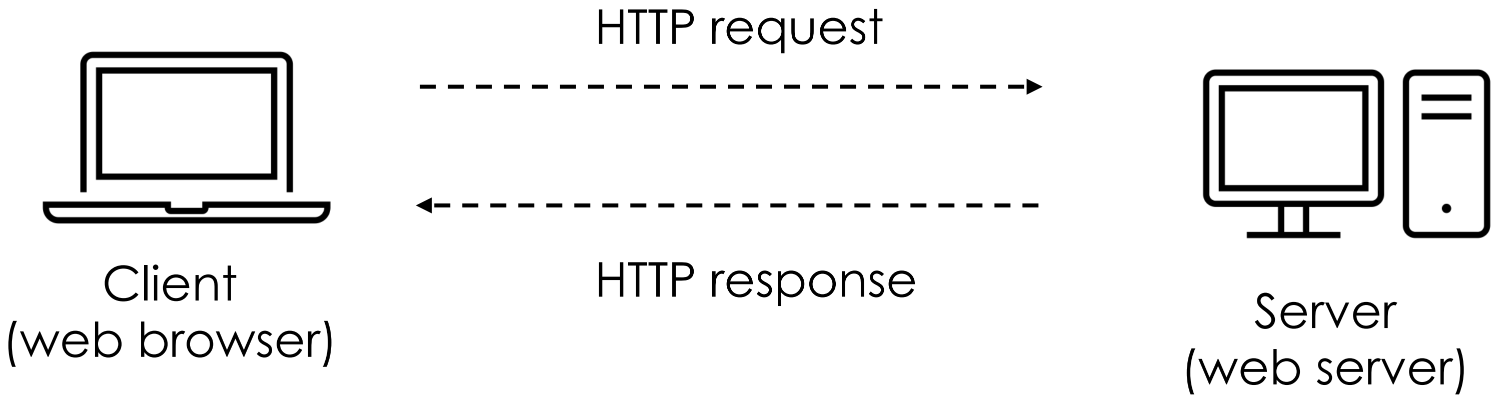
Introduction to Web Programming

2021-2022



Gérald Oster <`gerald.oster@telecomnancy.eu`>

Client / Server Model



HTML / CSS / (JavaScript)

Python / Flask

FLASK - Installation

- Micro framework: Flask

  https://flask.palletsprojects.com/en/2.0.x/

```
$ python3 –m venv venv
$ source venv/bin/activate
$ pip install Flask
```

FLASK – First web application

In a file named `app.py`

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello CS54'
```

In the terminal:

```
$ flask run
 * Environment: production
   WARNING: This is a development
   server. Do not use it in a
   production deployment.
   Use a production WSGI server
   instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/
  (Press CTRL+C to quit)
```

FLASK – How to run/configure the server

```
$ flask run
$ python -m flask run                          // alternatives
$ flask run -host=0.0.0.0 -port=8080           // passing options

// configuration using environment variables

$ export FLASK_APP=app.py
$ export FLASK_ENV=development                 // debug + autorefresh
$ export FLASK_RUN_HOST=localhost
$ export FLASK_RUN_PORT=8080


$ pip install python-dotenv // then put all env variables
                             // in a file named .env

// or in the code of your main file

if __name__ == '__main__':
     app.run(host='0.0.0.0', port=8080)
```

Routing

```
@app.route("/halloween")

@app.route("/halloween/")

@app.route("/monsters/<int:monster_id>")

@app.route("/some_full_path/<converter:variable_name>")
```

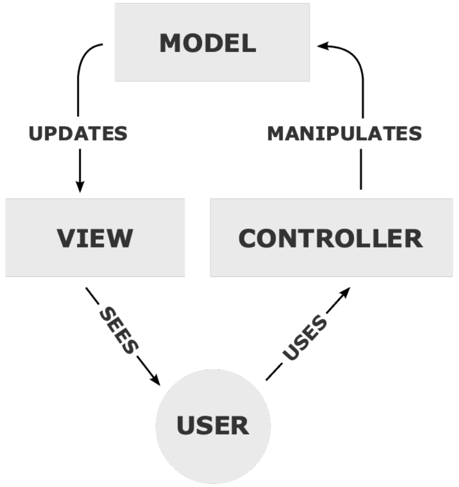
`string (default):`	accepts any text without a slash
`int:`	accepts positive integers
`float:`	accepts positive floating point values
`path:`	l ike string but also accepts slashes
`uuid:`	accepts UUID strings

Routing - Example

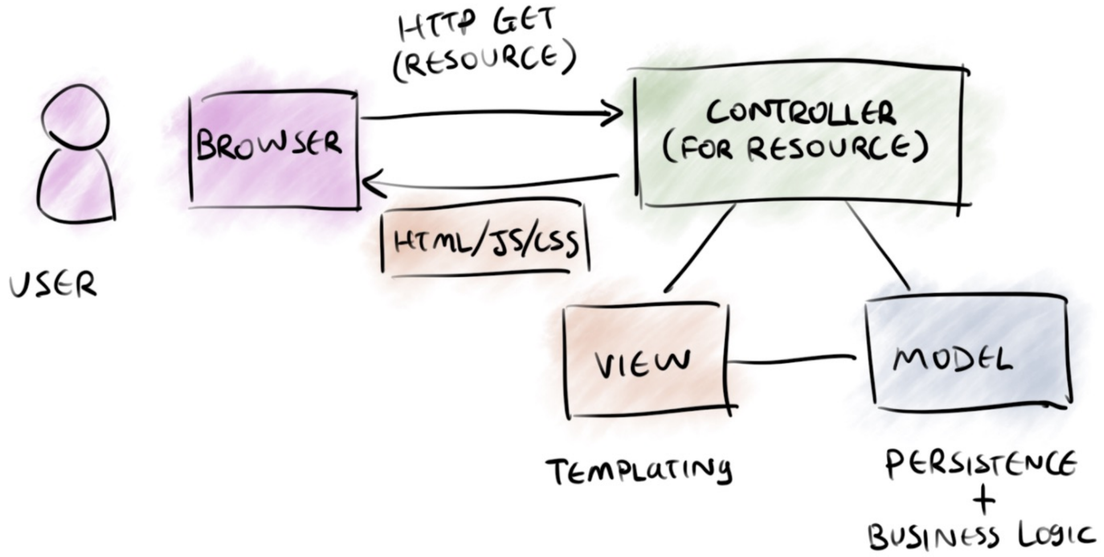
```python
@app.route("/monsters/<string:category>/<int:monster_id>")
def monster_by_id(category: str, monster_id: int):
   if category == "curcubitaceous" and monster_id == 666:
      return { "name": "Jack O'Lantern",
               "id":666, "category": "curcubitaceous" }
   else:
      return {}
```

```
>>> GET /monsters/curcubitaceous/666 HTTP/1.1
```

Model-View-Controller Pattern (MVC)



[Source: Wikipedia]



[Source: Medium, Robert Zhu]

What Not To Do

```
@app.route("/halloween")
def halloween():
   return """
<html>
<head>
<title>Halloween</title>
<style>
body { background: #000000; text-align:center;}
</style>
</head>
<body>
<img alt="jackolantern" width="100%" src="./static/halloween.jpg"/>
</body>
</html>
"""
```

Templates (powered by Jinja2)

In a file `templates/hello.html`

```
<!doctype html>
<html>
<title>Hello from Flask</title>
<body>
{% if name %}
   <h1>Hello {{ name }}!</h1>
{% else %}
   <h1>Hello, World!</h1>
{% endif %}
</body>
</html>
```

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<login>')
def hello(login=None):
    return render_template('hello.html', name=login)
```

HTTP Methods

```python
from flask import request

@app.route('/whois', methods=['GET', 'POST'])
def whois():
    if request.method == 'GET':
        return request.args.get('name', 'John Doe')
    else:
        return '' # do something else
```

```
>>> GET /whois?name=TheBoss HTTP/1.1
```

Forms (basic way of managing)

```
@app.route("/search", methods=["GET", "POST"])
def search():
   if request.method == "GET":
      return """<form action="/search" method="post">
                  <input name="q" type="search"/>
                  <input type="submit" value="Search"/>
               </form>"""
   elif request.method == "POST":
      return do_the_job(request.form['q'])
   else:
     return {}
```

Static Resources / Redirect

- (by default) `'/static'` route is used for **static resources** (images, files, etc.) and associated with **`static/`** directory.

- The url of a resource can be obtained using the following code:

```
url_for('static', filename='style.css')
```

Redirection and Errors

```
from flask import abort, redirect, url_for

@app.route('/')
def index():
    return redirect(url_for('login'))

@app.route('/login')
def login():
    abort(401)
    this_is_never_executed()


from flask import render_template
@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'), 404
```

Cookies

- Reading cookies:

```
from flask import request
@app.route('/')
def index():
   username = request.cookies.get('username')
   # something do to with
```

- Sending cookies:

```
from flask import make_response
@app.route('/')
def index():
   resp = make_response(render_template(...))
   resp.set_cookie('username', 'TheBoss')
   return resp
```

Sessions

Session object to store information from one request to the next ones by the same « user » (cf. cookies).

```
from flask import session

app.secret_key = b"MY_SECRET_KEY" // need to define a secret key


session["username"] = "TheBoss" // add a value to the session

session.get("username", default_value) // get value from the session

session.pop("username", default_value) // remove value from the session
```

Data Persistance

- Put everything in a database!

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
S. Adams E-mail: SCOTTADAMS@AOL.COM
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1995 United Feature Syndicate, Inc.(NYC)
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.

SQLite3

- SQLite3 (https://www.sqlite.org/index.html):
  - « Public domain » Embedded database / SQL92 compliant (mostly)
- In the virtual machine provided by TN:

```
$ sudo apt-get install sqlite3
```

- Main commands:

  `.help` (your friends ;b)

  `.open` (open a database file – sqlite file format)

  `.quit` (exit SQLite)

  `.tables` (show all tables)

  `.schema` (show schema of all tables)

Python / SQLite3

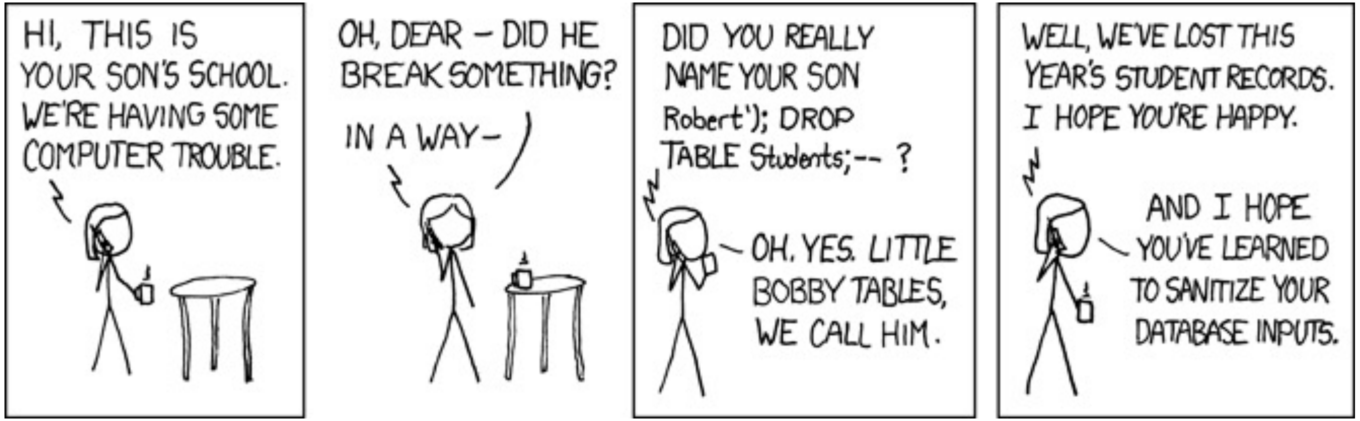
- https://docs.python.org/3/library/sqlite3.html

```python
import sqlite3

con = sqlite3.connect('halloween.db')

cur = con.cursor()

# print(cur.fetchone ())
for row in cur.execute('SELECT * FROM monsters ORDER BY height DESC'):
    print(row)
# print(cur.fetchall())

con.commit()
con.close()
```

Python / SQLite3 (cont.)

- What not to do (security issues: SQL injection):

```
monster_name = 'Cthulhu'
cur.execute("SELECT * FROM monsters WHERE name = '%s'" % monster_name)
# or:
cur.execute(f"SELECT * FROM monsters WHERE name = {monster_name} ")
```


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

[Source: https://xkcd.com/327/]

Python / SQLite3 (cont.)

- How to do it:

```
# either:
cur.execute("SELECT * FROM monsters WHERE (?)", ('Cthulhu'))
# or:
cur.execute("SELECT * FROM monsters WHERE name=:mname", {"mname": 'Cthulhu'})
```

SQLAlchemy

https://www.sqlalchemy.org/ - $ pip install SQLAlchemy

```
from sqlalchemy import create_engine

engine = create_engine('sqlite:///bookstore.db')
# engine = create_engine('postgresql://user:password@host/database')

con = engine.connect()

rs = con.execute('SELECT * FROM book’)

for row in rs:
   print(row)

con.close()
```

SQLAlchemy (cont.)

```
from sqlalchemy import create_engine
from sqlalchemy.sql import text

engine = create_engine('sqlite:///bookstore_tmp.db')
con = engine.connect()

rs = con.execute('DROP TABLE IF EXISTS book')
rs = con.execute('CREATE TABLE book (id INTEGER PRIMARY_KEY,
                        title VARCHAR, primary_author VARCHAR)')

statement = text('INSERT INTO book(id, title, primary_author) VALUES
                        (:id, :title, :primary_author)')
rs = con.execute(statement, {'id':1, 'title':'The Silmarillion’,
                                    'primary_author':'Tolkien' })
for row in rs:
   print(row)
con.close()
```

(very) Few Words About Testing

- By hand ;(
  - cURL / Requests module / …
  - Thunder Client extension in VSCode
- Unit tests using pytest
- End-to-end tests using Selenium

To go further…

- Views and Blueprints
- SQLAlchemy (Python ORM)
- REST API (Marshmallow)
- Forms (WTF)
- Security (Authentication)
- Deployment (WSGI)
- …