CS 240
Computer Systems

#12: Web Services, Python, and flask

Oct. 5, 2021 · Wade Fagen-Ulmschneider

Web Services

We describe anything that provides data back from an HTTP endpoint as a "web service". There are several three main categories:

[1]:

```
REQ  1  GET /cs240/fa2021/ HTTP/1.1\r\n
     2  Host: courses.grainger.illinois.edu\r\n
     …  ...
```

Advantages:

Disadvantages:

---

[2]:

National Weather Service API:

`https://api.weather.gov/points/{latitude},{longitude}`

Location of 2215 CIF:

```
REQ  1  GET /points/                    HTTP/1.1\r\n
     2  Host: api.weather.gov\r\n
     …  ...
```

RESTful Requirements:

[3]:

```
REQ  1  POST /q/api/queues/788/staff/1 HTTP/1.1\r\n
     2  Host: queue.illinois.edu\r\n
     3  Content-Length: 477\r\n
     …  …
        { "id": 30540, "startTime": "2021-10-05T15:48:57.000Z", "endTime": null, "createdAt":
        "2021-10-05T15:48:57.000Z", "updatedAt": "2021-10-05T15:48:57.000Z", "userId": 1,
        "queueId": 788, "user": { "name": "Wade A Fagen-Ulmschneider", "id": 1, "netid": "waf",
        "universityName": "Wade A Fagen-Ulmschneider", "preferredName": null, "isAdmin": true,
        "createdAt": "2018-02-14T05:27:54.000Z", "updatedAt": "2018-02-21T05:34:33.000Z" } }
```

HTTP Verbs (Defined in RFC 7231 §4)

Every HTTP request has an "action verb" that describes the action requested of the web server:

- **`GET`**: Requests a representation of the specified resource. Requests using GET should only retrieve data.
- **`POST`**: Submits an entity to the specified resource, often causing a change in state or side effects on the server.

- **`HEAD`**: Asks for a response identical to a GET request, but without the response body.
- **`PUT`**: Replaces all current representations of the target resource with the request payload.
- **`DELETE`**: Deletes the specified resource.
- **`PATCH`**: Applies partial modifications to a resource.

- **`CONNECT`**: Establishes a tunnel to the server identified by the target resource.
- **`OPTIONS`**: Describes the communication options for the target resource.
- **`TRACE`**: Performs a message loop-back test along the path to the target resource.

Why the Web Works: Allowing Maximal Flexibility and Acceptance

Python Programming
All modern programming languages provide many libraries for quickly and easily working with web requests. In CS 240, we will focus on Python and use the **flask** library for web requests.

Python Overview:

- Python is an "interpreted" programming language:
    - **Note**: Python only allows one thread to access the CPU (others can be blocked or ready, but there is no parallel execution)! *(Simplifies the execution environment, but prevents optimizations that are possible in C/C++.)*
- Python is a "dynamically typed" programming language:
- Python's control-flow is whitespace delimited:
- Python places heavy emphasis on code readability:

12/hello.py

```
1  s1 = "Hello"
2  s2 = " World"
3
4  for i in range(10):
5    if i < 5:
6      print(s1)
7    elif i < 8:
8      print(s1 + s2)
9    else:
10     print(s2)
```

Flask Library:
The flask library focuses on providing a simple interface to handling web requests:

12/app.py

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  # Route for "/" for a web-based interface to this
   micro-service:
5  @app.route('/')
6  def index():
7    from flask import render_template
8    return render_template("index.html")
9
10 # Extract a hidden "uiuc" GIF from a PNG image:
11 @app.route('/extract', methods=["POST"])
12 def extract_hidden_gif():
13   # ...
```

Import Statements (Line 1, 7):

Python Comments (4, 10):

Python Function Definitions (Lines 6, 12):

Python Decorator (Lines 5, 11):

Running a Python program:

Flask is widely used, lots of great resources available! *(This is why we use widely used libraries!)*