Web

Alexander Evgin

13 марта 2020 г.

Outline

Recap

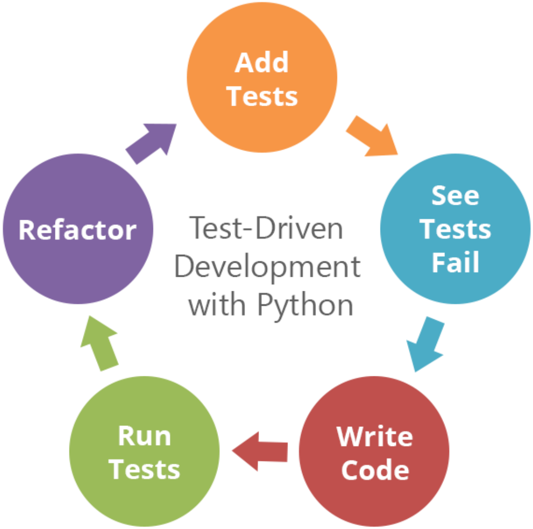
Tier 1: docstrings

Tier 2: type annotations

Tier 3: `assert`

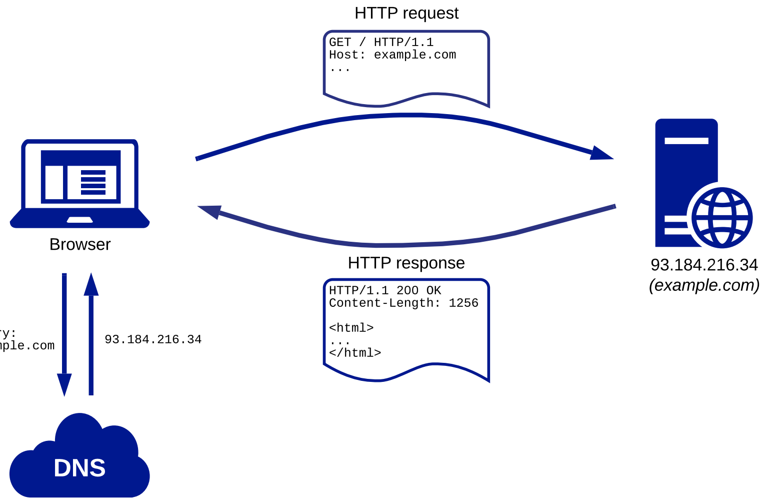
Tier 4: `pytest` (`unittest`, `doctest`)

Recap


Add Tests
See Tests Fail
Write Code
Run Tests
Refactor
Test-Driven Development with Python

Web

Browsing Algorithm



```
GET / HTTP/1.1
Host: example.com
...
```

```
HTTP/1.1 200 OK
Content-Length: 1256

<html>
...
</html>
```

Terminology

- HTTP
- HTTP request/response
- URL
- HTML
- Client
- Server
- Web Application: frontend, backend
- ...

URL

```
http[s]://example.com/foo/baz/bar?param=1&param2=0#frag1
```

schema *host* *path* *params* *fragment*

HTTP (Hypertext Transfer Protocol)

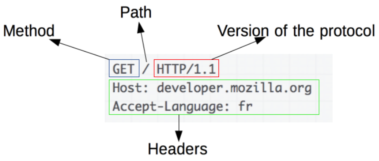
Request

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

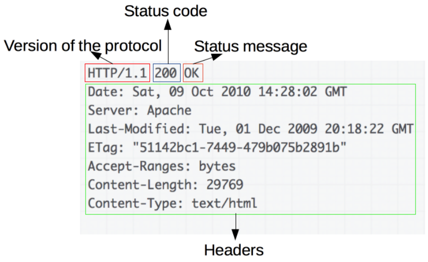
Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World.</p>
  </body>
</html>
```

HTTP (Hypertext Transfer Protocol)



```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

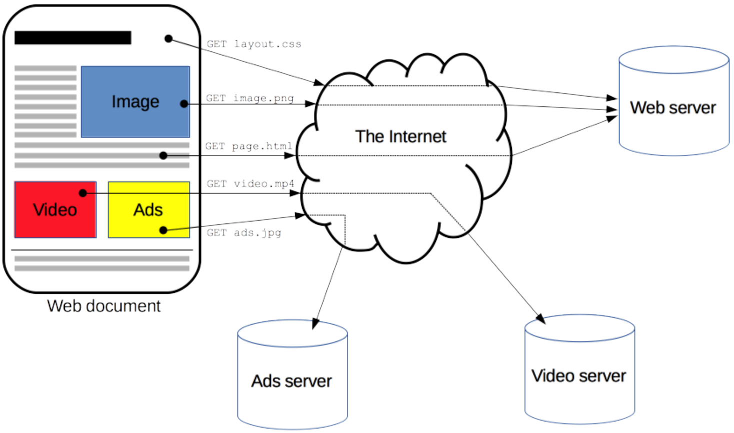


```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

HTML

```
<html>
  <head>
    <title>Example Domain</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css"
    href="/static/style.css">
  </head>
  <body>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in
    documents.</p>
    <p><a href="https://www.iana.org/domains/example">More
    information...</a></p>
     <img src="https://via.placeholder.com/150"
     alt="Smiley face">
  </body>
</html>
```

Fetching a page



Cookies

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry

[page content]
```

```
GET / HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

Cookies

Session management
Logins, shopping carts, game scores, or anything else the server should remember

Personalization
User preferences, themes, and other settings

Tracking
Recording and analyzing user behavior

urllib

Make HTTP requests from Python:

```
>>> from urllib import request
>>> response = request.urlopen('http://example.com/')
>>> response.code
200
>>> response.read().decode('utf-8')
<!doctype html>...
```

urllib

```
from urllib import request

req = request.Request('http://www.example.com/')

req.add_header('Referer', 'http://www.me.ru/')
req.add_header('User-Agent', 'urllib-example/0.1')

r = request.urlopen(req)
```

urllib

```
>>> from urllib.parse import urlparse
>>> url = 'http://example.com/users/myself?key=1000'
>>> o = urlparse(url)
>>> o
ParseResult(scheme='http', netloc='example.com',
path='/users/myself', params='', query='key=1000',
fragment='')
```

requests

Much better!

```
>>> import requests
>>> response = requests.get('http://example.com')
>>> response.status_code
200
>>> response.headers['content-length'] # case insensitive
'648'
>>> print(response.text)
<!doctype html>
<html>
...
```

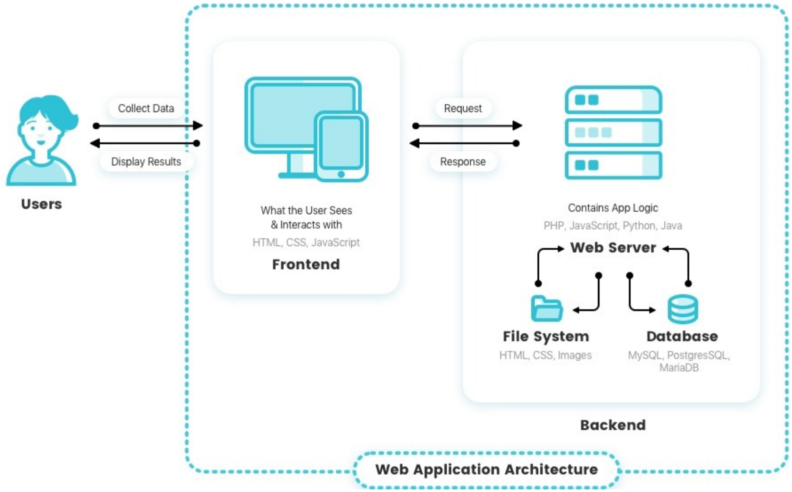
requests

Really user-friendly API:

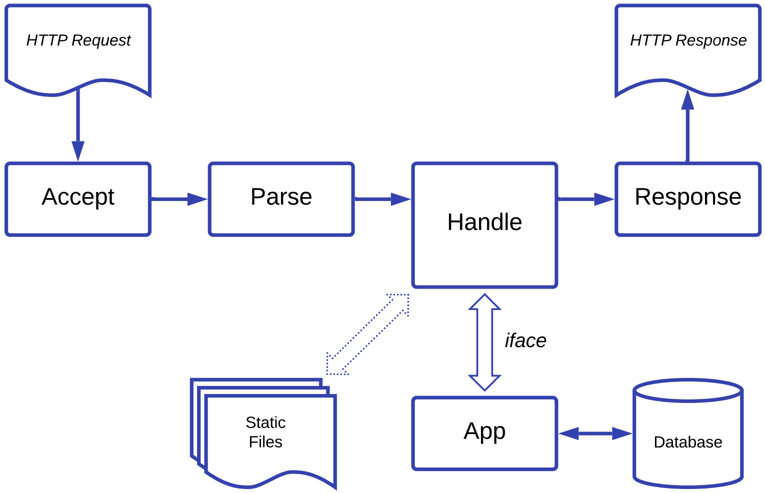
```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get('http://example.com', params=payload)
>>> print(r.url)
http://example.com/?key2=value2&key1=value1
```

```
>>> r = requests.get('https://api.github.com/events')
>>> r.json()
[{u'repository': {u'open_issues': 0, u'url':
'https://github.com/...
```

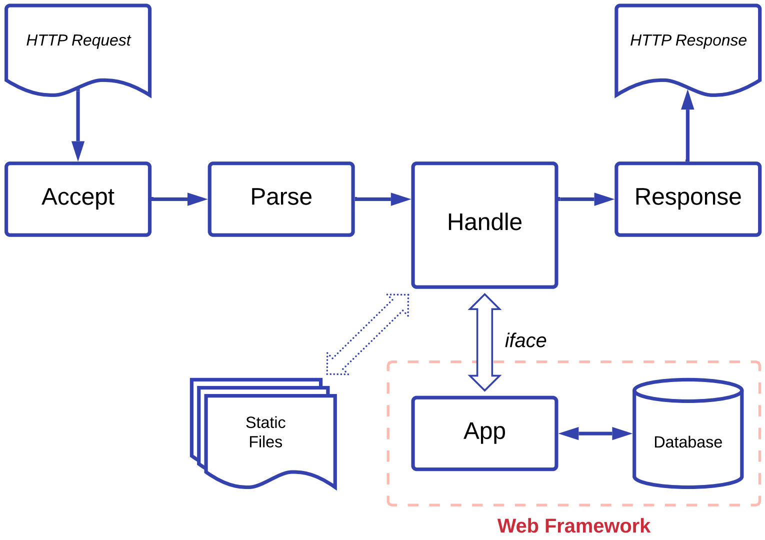
Web Application



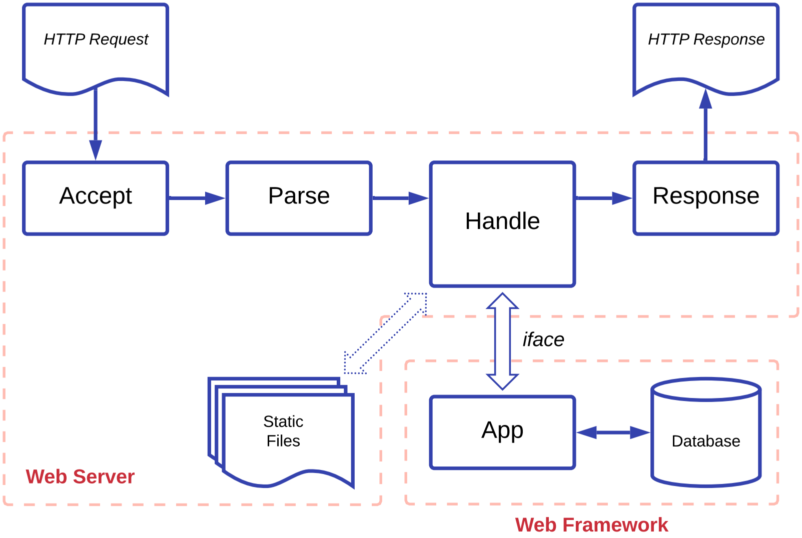
Backend


HTTP Request
Accept
Parse
Handle
Response
HTTP Response
iface
Static Files
App
Database

Backend

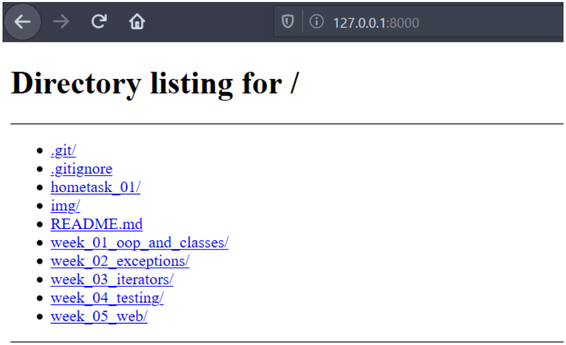

HTTP Request
HTTP Response
Accept
Parse
Handle
Response
iface
Static Files
App
Database
Web Framework

Backend


HTTP Request
HTTP Response
Accept
Parse
Handle
Response
iface
Static Files
App
Database
Web Server
Web Framework

Package `http`

```
python -m http.server 8000 --bind 127.0.0.1
```

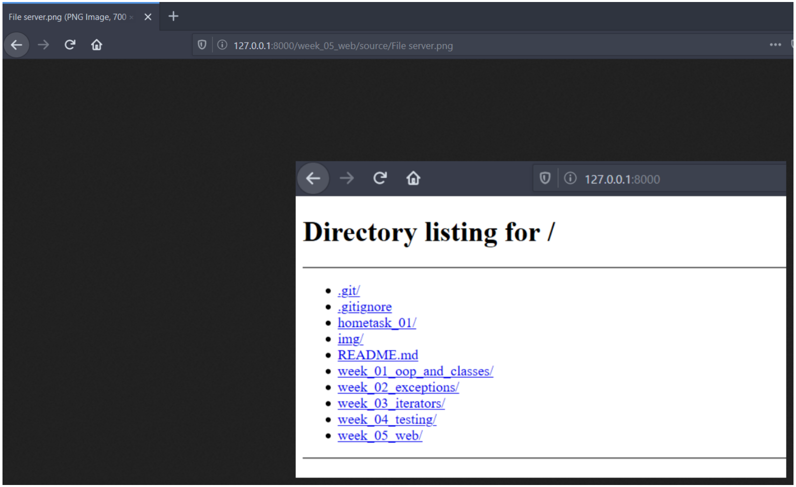


Directory listing for /

- .git/
- .gitignore
- hometask_01/
- img/
- README.md
- week_01_oop_and_classes/
- week_02_exceptions/
- week_03_iterators/
- week_04_testing/
- week_05_web/

Try this on right now!

Package `http`


File server.png (PNG Image, 700 ×

127.0.0.1:8000/week_05_web/source/File server.png

127.0.0.1:8000

Directory listing for /

- .git/
- .gitignore
- hometask_01/
- img/
- README.md
- week_01_oop_and_classes/
- week_02_exceptions/
- week_03_iterators/
- week_04_testing/
- week_05_web/

Package `http`

```
Serving HTTP on 127.0.0.1 port 8000 (http://127.0.0.1:8000/)
...
127.0.0.1 - - [11/Mar/2020 01:43:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2020 01:46:55] "GET /week_05_web/
HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2020 01:47:01] "GET /week_05_web/source/
HTTP/1.1" 200 -
127.0.0.1 - - [11/Mar/2020 01:47:03] "GET
/week_05_web/source/File%20server.png HTTP/1.1" 200 -
```

Package http

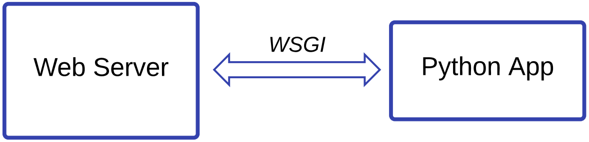
```
from http.server import HTTPServer, SimpleHTTPRequestHandler

class Handler(SimpleHTTPRequestHandler):
    def do_GET(self):
        print(
            'Request from {}:{}'.format(*self.client_address)
        )
        return super().do_GET()

PORT = 8080
HOST = '127.0.0.1'

with HTTPServer((HOST, PORT), Handler) as httpd:
    print(f'Serving at {HOST}:{PORT}')
    httpd.serve_forever()
```

Package `http`

```
python fileserver.py
```

```
Serving at 127.0.0.1:8080
Request from 127.0.0.1:55018
127.0.0.1 - - [13/Mar/2020 02:09:11] "GET / HTTP/1.1" 200
```

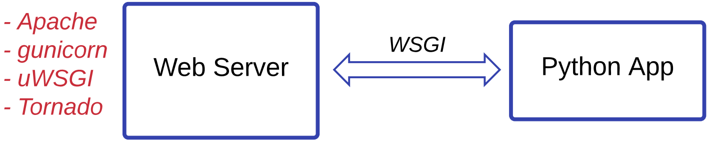
WSGI (Web Server Gateway Interface)



WSGI application

- callable
- `environ`, `start_response`
- call `start_response` with HTTP code and headers
- return iterable object: response body

WSGI (Web Server Gateway Interface)



WSGI application

- callable
- `environ`, `start_response`
- call `start_response` with HTTP code and headers
- return iterable object: response body

WSGI (Web Server Gateway Interface)

```python
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    yield b'Hello, World\n'
```

Flask

“Hello world”

```python
# hello.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, world!'
```

```
PS C:\path\to\app> $env:FLASK_APP="hello.py"
PS C:\path\to\app> flask run
```

(On Windows PowerShell)

Flask globals

```python
from flask import Flask, request

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def hello():
    if request.method == 'POST':
        save_data()
    return render_response(200)
```

Escaping

```python
from flask import Flask, escape

app = Flask(__name__)

@app.route('/user/<username>')
def hello(username):
    return f'Hello, {escape(username)}'
```

Jinja2

`templates/index.html`:

```
<html>
<body>Hello, {{ name }} !</body>
</html>
```

```
from flask import Flask, escape, render_template

app = Flask(__name__)

@app.route('/user/<username>')
def hello(username):
    return render_template('index.html',
                           name=escape(username))
```

`Flask` — ***micro***-framework, ***huge*** opportunities

Q & A