Quart; an asyncio alternative to Flask

P G Jones - 2018-07-27
pgjones@stet.io

Quart; an ASGI alternative to Flask

P G Jones - 2018-07-27
pgjones@stet.io

Me

Background rejection for the neutrinoless doublebeta decay experiment SNO+.
Lincoln College, Oxford, UK.
2011 DPhil

VP Engineering Smarkets

@pgjones on github & gitlab

Flask

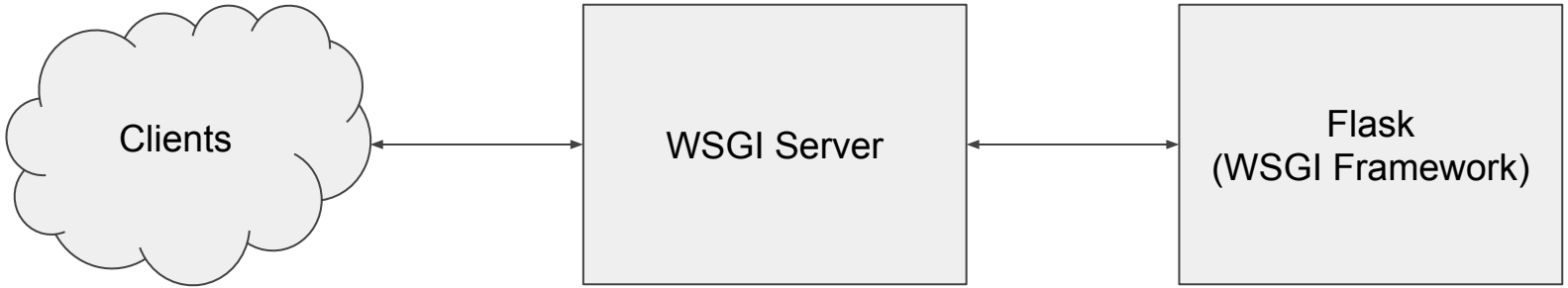
```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def hello_world():
    return render_template('index.html')

app.run()
```

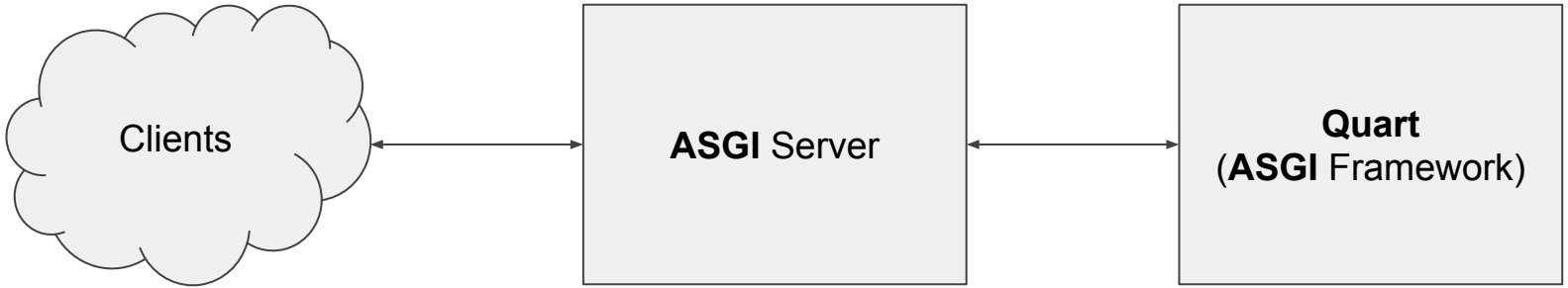
Production



WSGI, Web Server Gateway Interface

```
def application(environ, start_response):
    ...
    start_response(status, headers)
    return ...
```

Production



ASGI, Asynchronous Server Gateway Interface

```python
class Application:
    def __init__(self, scope):
        self.scope = scope

    async def __call__(self, receive, send):
        event = await receive()
        ...
        await send({
            "type": "http.response.start",
            "status": 200, ...
        })
```

ASGI Servers

Server name	HTTP/2	Server Push	Websocket Response
Hypercorn	✓	✓	✓
Uvicorn	✗	✗	✗
Daphne	✓	✗	✗

Explicit asynchronous code

```
async def coro():
    await other_coro()
    sync()

def sync():
    other_coro()  # Creates a coroutine, but doesn't run it
    await other_coro()  # SyntaxError
```

Introducing Quart

---

https://gitlab.com/pgjones/quart

0.6.4 Current release, released on the 2018-07-15

MIT Licensed

Python 3.6.1 or greater

Quart

```python
from quart import Quart, render_template

app = Quart(__name__)

@app.route('/')
async def hello_world():
    return await render_template('index.html')

app.run()
```

Quart & Flask

Quart aims to exactly match the Flask public API.

Quart tries to match the Flask private API.

As Flask - Basic Authentication

```
def requires_auth(func):
    @wraps(func)
    async def decorated(*args, **kwargs):
        auth = request.authorization
        if not auth or not check_auth(auth.username,
                                      auth.password):
            abort(401)
        return await func(*args, **kwargs)
    return decorated
```

Flask Extensions

```python
def extension_code():
    data = request.get_json()
    data['key']  # Error


async def quart_code():
    data = await request.get_json()
    data['key']
```

Beyond Flask - Streaming requests

```
from async_timeout import timeout

@app.route('/', methods=['POST'])
async def index():
    async with timeout(app.config['BODY_TIMEOUT']):
        async for data in request.body:
            ...
```

Streaming responses

```
@app.route('/sse')
async def sse():
    async def send_events():
        ...
        event = ServerSentEvent(data)
        yield event.encode()

    return send_events(), {
        'Content-Type': 'text/event-stream',
        'Cache-Control': 'no-cache',
        'Transfer-Encoding': 'chunked',
    }
```

Beyond Flask - HTTP/2

```
---
...
---
async def index():
    result = await render_template('index.html')
    response = await make_response(result)
    response.push_promises.add(
        url_for('static', filename='css/base.css'),
    )
    return response
```

https://medium.com/python-pandemonium/how-to-serve-http-2-using-python-5e5bbd1e7ff1

Beyond Flask - Websockets

```python
from quart import Quart, websocket

app = Quart(__name__)

@app.websocket('/ws')
async def ws():
    while True:
        data = await websocket.receive()
        await websocket.send(data)
```

https://medium.com/@pgjones/websockets-in-quart-f2067788d1ee

Websockets @ EuroPython

```
connected = set()

async def broadcast(message):
    for websock in connected:
        await websock.send(message)

@app.websocket('/ws')
async def ws():
    connected.add(websocket._get_current_object())
    while True:
        fruit = await websocket.receive()
        if fruit in {🍒, 🐱}:
            await broadcast(fruit)
```

https://ep2018.europython.eu/conference/talks/python-and-web-sockets

Quart Extensions

- Quart-CORS Cross Origin Resource Sharing (access control)

- Quart-OpenApi RESTful API building.

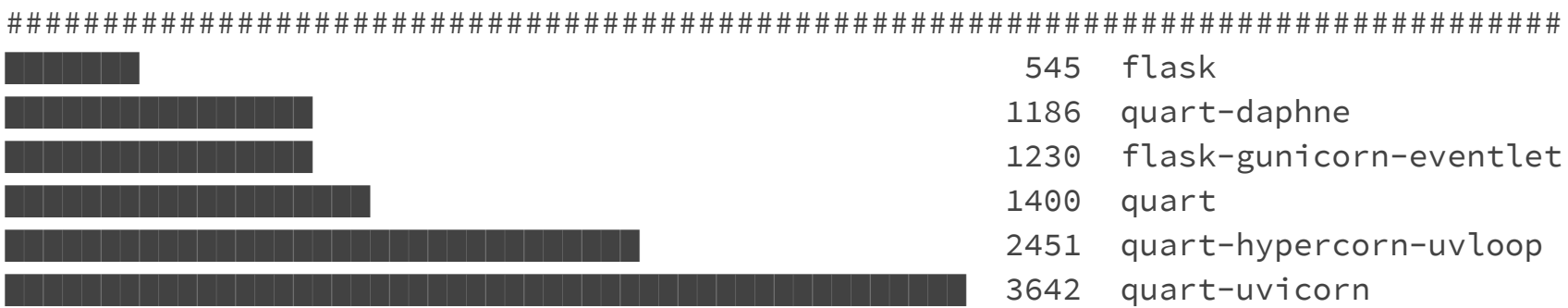
https://pgjones.gitlab.io/quart/flask_extensions.html

Quart type hinting

```
ResponseValue = Union[
    Response, str, AsyncGenerator[bytes, None],
    Generator[bytes, None, None],
]

ResponseReturnValue = Union[
    ResponseValue,
    Tuple[ResponseValue, dict],
    Tuple[ResponseValue, int],
    Tuple[ResponseValue, int, dict],
]
```

Simple Benchmarks



https://gitlab.com/pgjones/quart-benchmark

Beyond Flask - Faster

Route	Requests per second		Average Latency [ms]	
	Flask	Quart	Flask	Quart
GET /films/995/	330.22	1160.27	60.55	17.23
GET /films/	99.39	194.58	201.14	102.76
POST /reviews/	324.49	1113.81	61.60	18.22

https://hackernoon.com/3x-faster-than-flask-8e89bfbe8e4f

Quart future

Quart & Flask

Trio/Curio compatibility?

Something you need?

Conclusion

ASGI is the async equivalent of WSGI

Quart is quite powerful/useful

I’d like contributions, bug reports, production reports, ideas, etc...