An introduction to ASGI, Asynchronous Server Gateway Interface.

P G Jones - 2019-06-15
pgjones.dev

Me

pgjones.dev
@pgjones github/gitlab
@pdgjones twitter

octopus wealth

2003 Frameworks

Zope, Quixote, Webware, SkunkWeb, PSO, and Twisted Web

QUIXOTE.CA

Motivation for WSGI

– – –

“Python currently boasts a wide variety of web application frameworks, such as Zope, Quixote, Webware, SkunkWeb, PSO, and Twisted Web -- to name just a few [1]. This wide variety of choices can be a problem for new Python users, because generally speaking, their choice of web framework will limit their choice of usable web servers, and vice versa.”

- PEP 333

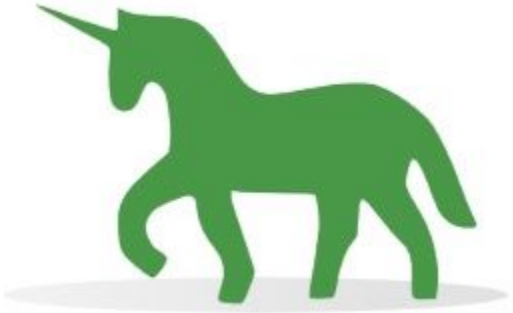
WSGI

Web
Server
Gateway
Interface

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

WSGI Frameworks

WSGI Servers



gunicorn

uWSGI

Apache

mod_wsgi

WSGI Limitations

- Websockets
  - wsgi.websocket as an unofficial workaround
- HTTP/2 (Concurrency)
- Can’t use async or await

Motivation for async Web

```python
from aiohttp import web

async def hello(request):
    await db.execute("SELECT * FROM tbl")
    return web.Response(text="Hello, world")

app = web.Application()
app.add_routes([web.get('/', hello)])
web.run_app(app)
```

2019 Frameworks

Japronto!

Motivation for ASGI

"Python currently boasts a wide variety of async web application frameworks, such as aiohttp, Sanic, Blacksheep, Japronto, Vibora to name just a few [1]. This wide variety of choices can be a problem for new Python users, because generally speaking, their choice of web framework will limit their choice of usable web servers, and vice versa."

ASGI

Asynchronous
Server
Gateway
Interface

```python
async def application(scope, receive, send):
    event = await receive()
    ...
    await send(
        {"type": "websocket.send", ...}
    )
```

https://asgi.readthedocs.io/en/latest/

Simple example

```
async def app(scope, receive, send):
    assert scope["type"] == "http"

    await send({
        "type": "http.response.start",
        "status": 200,
        "headers": [
            (b"content-type", "text/plain"),
        ],
    })

    await send({
        "type": "http.response.body",
        "body": b"Hello, world!",
    })
```

ASGI Development

- 2015 ASGI 1
  - Safely ignored
- 2017 ASGI 2
  - Very similar to ASGI 3
- 2019 ASGI 3
  - The version I am presenting

ASGI HTTP Scope

```
{
    "type": "http",
    "http_version": "2.0",
    "asgi": {
        "spec_version": "2.1",
        "version": "3.0",
    },
    "method": "POST",
    "scheme": "https",
    "path": "/",
    "query_string": b"a=b",
    "headers": [(b"content-length", b"5")],
    "client": ("10.0.0.0", 1234),
    "server": ("127.0.0.1", 443),
    "root_path": ".",
}
```

ASGI HTTP Messages

Server

```
{
    "type": "http.request",
    "body": b"hello",
    "more_body": True
}

{
    "type": "http.request",
    "body": b"world",
    "more_body": False
}

{"type": "http.disconnect"}
```

Application

```
{
    "type": "http.response.start",
    "status": 200,
    "headers": [(b"name", b"value")],
}

{
    "type": "http.response.body",
    "body": b"Hi",
    "more_body": False,
}
```

ASGI WebSocket Scope

```
{
    "type": "websocket",
    "http_version": "2.0",
    "asgi": {
        "spec_version": "2.1",
        "version": "3.0",
    },
    "scheme": "wss",
    "path": "/",
    "query_string": b"a=b",
    "headers": [(b"name", b"value")],
    "subprotocols": ["subprotocol"],
    "client": ("10.0.0.0", 1234),
    "server": ("127.0.0.1", 443),
    "root_path": ".",
}
```

ASGI WebSocket Messages

Server

```
{
    "type": "websocket.receive",
    "bytes": b"Hello",
}

{
    "type": "websocket.receive",
    "text": "Hello",
}

{"type": "websocket.disconnect"}
```

Application

```
{"type": "websocket.accept"}

{
    "type": "websocket.send",
    "text": "Hi",
}

{
    "type": "websocket.close",
    "code": 1000,
}
```

ASGI Lifespan

```
---
scope = {
    "type": "lifespan",
    "asgi": {"spec_version": "2.0", "version": "3.0"},
}
--- SERVER
{"type": "lifespan.startup"}
{"type": "lifespan.shutdown"}
--- APPLICATION
{"type": "lifespan.startup.complete"}
{"type": "lifespan.shutdown.complete"}
```

ASGI Extensions - Server Push

```
---

"scope": {
    ...,
    "extensions": {"http.response.push": {}},
}
---
{
    "type": "http.response.push",
    "path": "/path",
    "headers": [(b"name", b"value")],
}
```

ASGI Extensions - WebSocket HTTP Response

```
---

"scope": {..., "extensions": {"websocket.http.response": {}}}
---
{
    "type": "websocket.http.response.start",
    "status": 401,
    "headers": [(b"name", b"value")],
}
---
{
    "type": "websocket.http.response.body",
    "body": b"Hi",
    "more_body": False,
}
```

ASGI Frameworks

ASGI Frameworks Compared

Framework	HTTP	WebSocket	Server Push	WebSocket HTTP response
Quart	✓	✓	✓	✓
FastAPI	✓	✓	❌	❌
Responder	✓	✓	❌	❌
Starlette	✓	✓	❌	❌
Django (Channels)	✓	✓	❌	❌

ASGI Servers

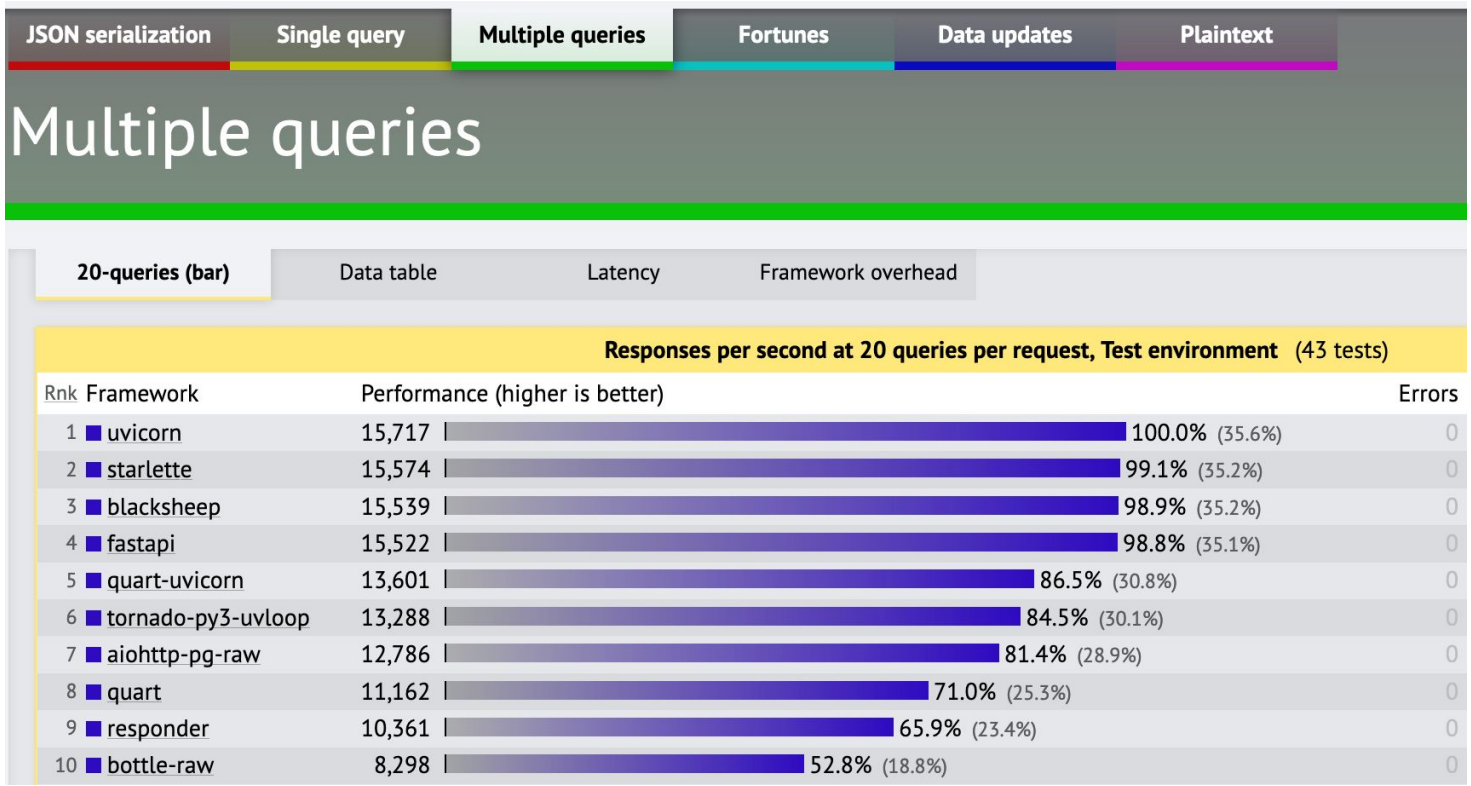
Daphne

Mangum

ASGI Servers Compared

Framework	HTTP	WebSockets	Server Push	WebSocket HTTP response	Serverless
Hypercorn	1 ✓ 2 ✓	1 ✓ 2 ✓	✓	✓	❌
Uvicorn	1 ✓ 2 ❌	1 ✓ 2 ❌	❌	❌	❌
Daphne	1 ✓ 2 ✓	1 ✓ 2 ❌	❌	❌	❌
Mangum	1 ✓ 2 ❌	1✓ 2 ❌	❌	❌	✓

TechEmpower Round 17 2018-10-30 Python only



Questions?