Connexion: open-source spec-first API framework

Robbe Sneyders

Principal Engineer @ ML6
Tech lead @ Connexion
Tech lead @ Fondant

RobbeSneyders
robbesneyders

Agenda

Topic
API-first & spec-first
Connexion
Concurrency, parallelism, and async
WSGI, ASGI & Middleware
Working in open source

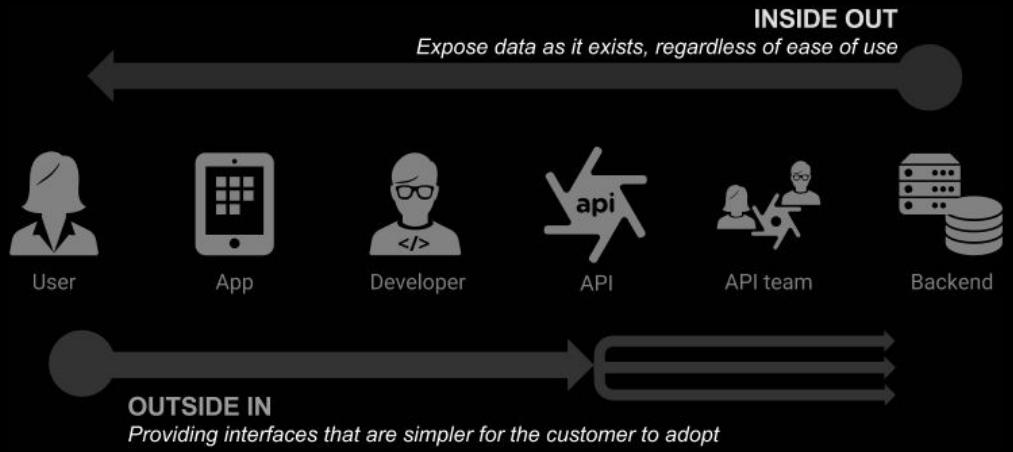
What is API-first?

- A product-centric approach to developing APIs, where APIs are treated as “first-class citizens”. APIs are regarded as discrete products instead of integrations, and a set of APIs can form an API platform that fosters innovation.

- Defining and designing APIs and underlying schema before developing dependent APIs, applications, or integrations.

Why API-first?

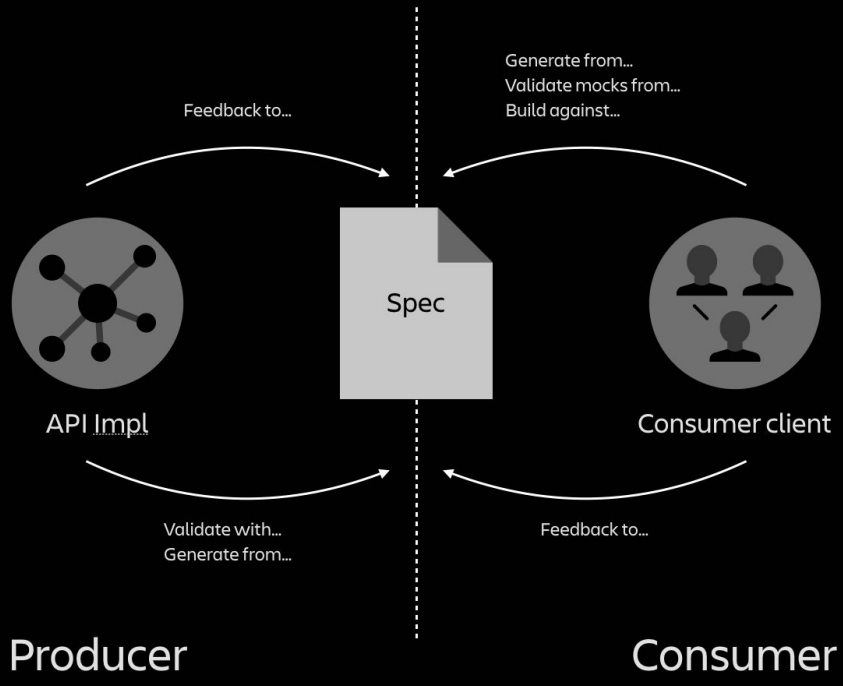
- Designing APIs around the consumers' use cases for the data rather than the structure of the data in your system ensures seamless integration
- APIs can be reused throughout the business, offering faster delivery and saving valuable developer resources



https://www.googlecloudcommunity.com/gc/Cloud-Product-Articles/Designing-APIs-from-the-Outside-In/ta-p/78985

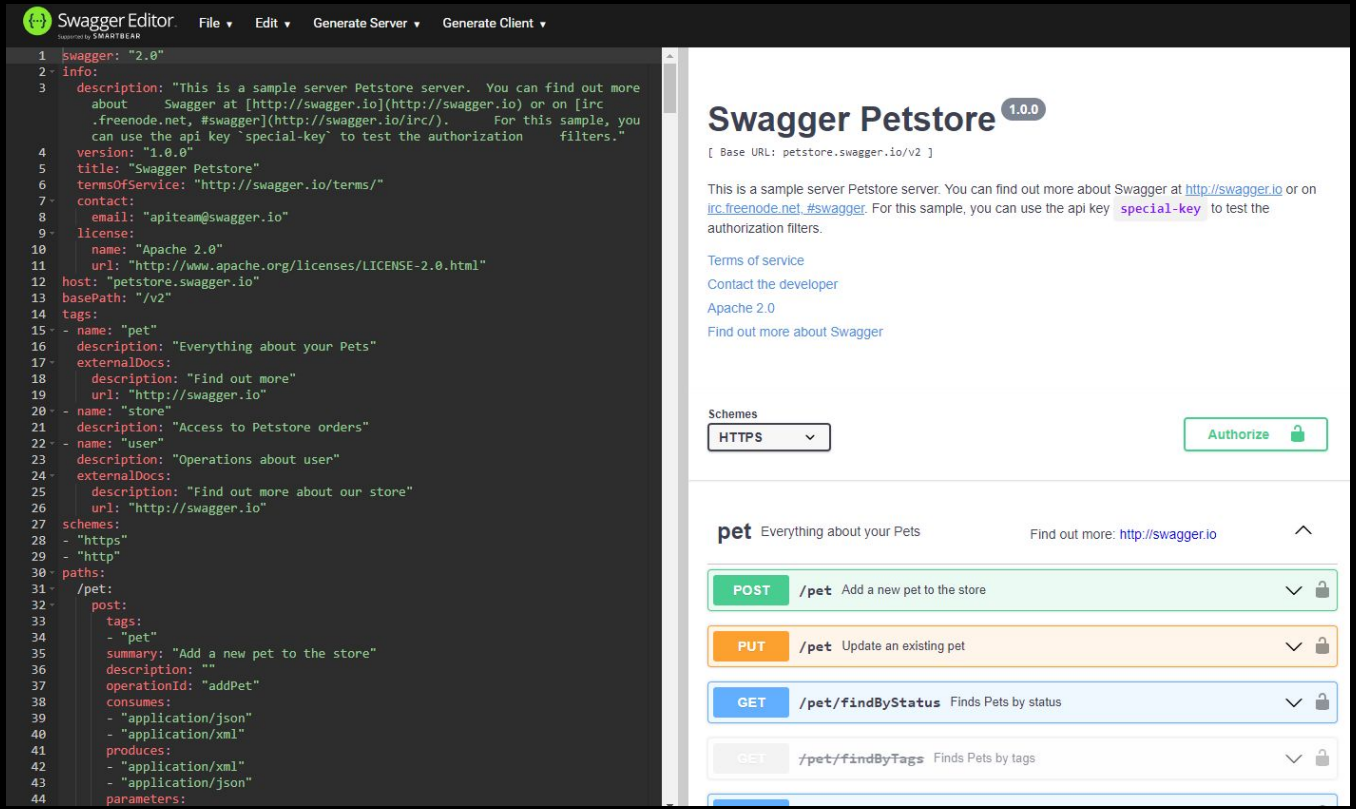
Why spec-first?

- Clear separation of WHAT vs. HOW: focus on the business problem we are trying to solve without mixing in implementation details
- Quick iteration on API definition based on feedback from different stakeholders
- Decoupled development of client applications and service provider
- Single source of truth for the API specification
  - Simplify compliance & governance
  - Basis for tooling & automation



https://www.atlassian.com/blog/technology/spec-first-api-development

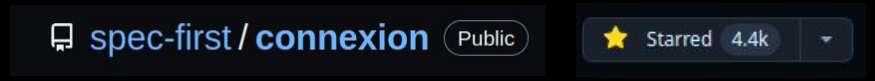
How to spec-first?



Agenda

Topic
API-first & spec-first
Connexion
Concurrency, parallelism, and async
WSGI, ASGI & Middleware
Working in open source

Connexion is an open source Python API framework that enables API-first development



Downloads last day: 233,261
Downloads last week: 1,457,469
Downloads last month: 6,289,652

Connexion is an open source Python API framework that enables API-first development

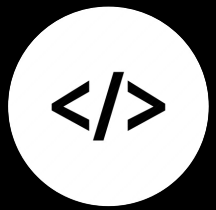
spec-first / connexion Public

Starred 4.4k

Connexion

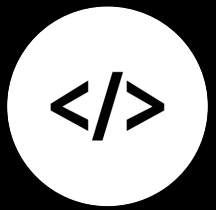
Contract

>



Code

Everyone else



Code

>

Contract

You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Instantiate an app and add your specification as an API:

```
import connexion

app = connexion.App(__name__, specification_dir='openapi/')
app.add_api('my_api.yaml')
app.run(port=8080)
```

You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

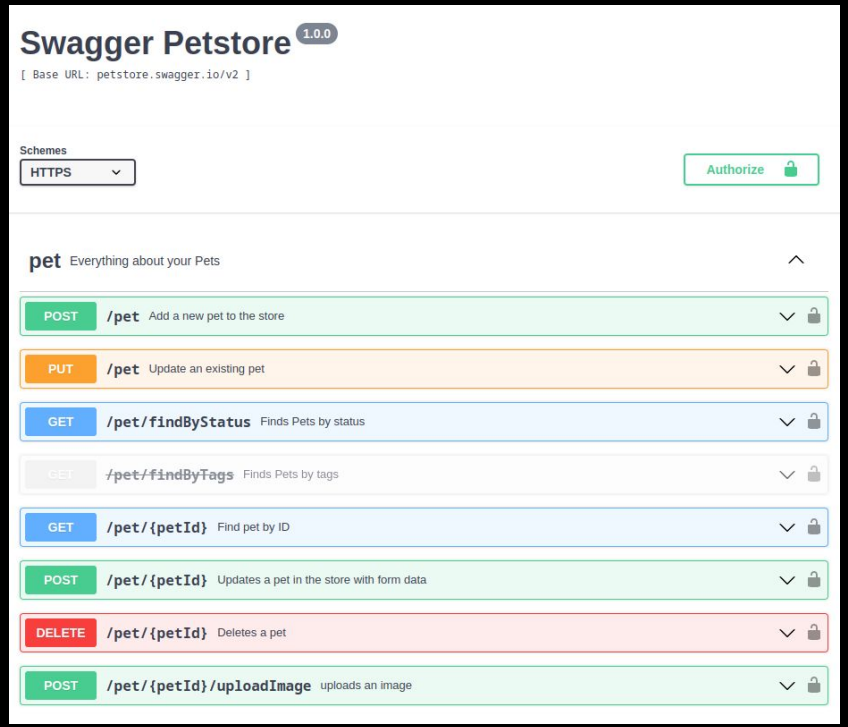
- Provides mock functionality to test and develop against the specification before implementation is ready or even started

```
connexion run specification.yaml --mock=all
```

You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

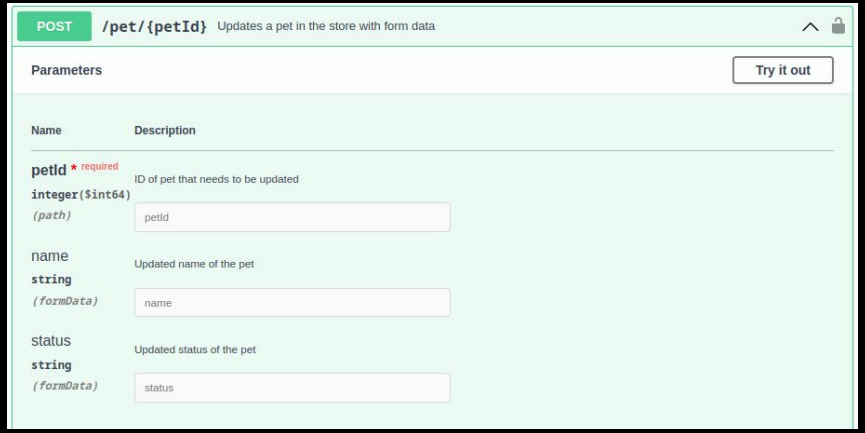
- Provides a Swagger UI for live documentation and testing



You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

- Routes your API endpoints to your python functions and provides request parameters as function arguments



Other frameworks:

```
@route(...)
def post_pet(request):
    ...
```

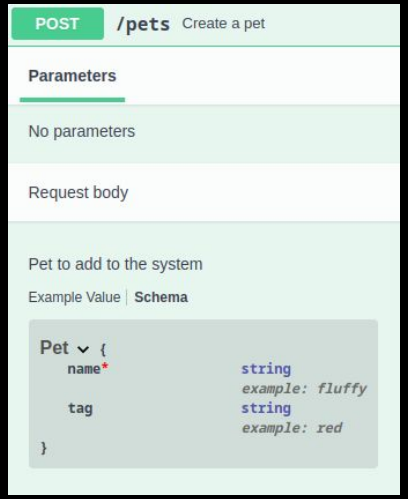
Connexion:

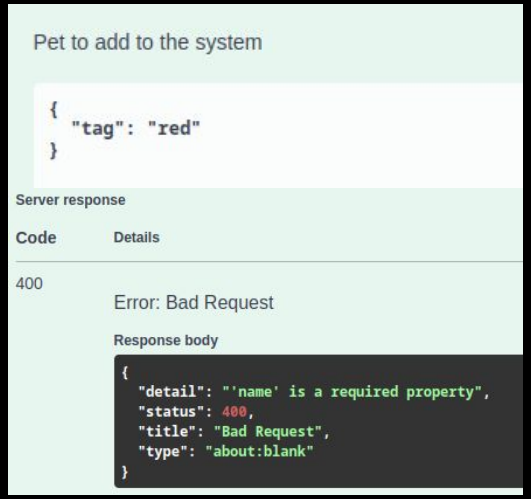
```
def post_pet(pet_id, name, status):
    ...
```

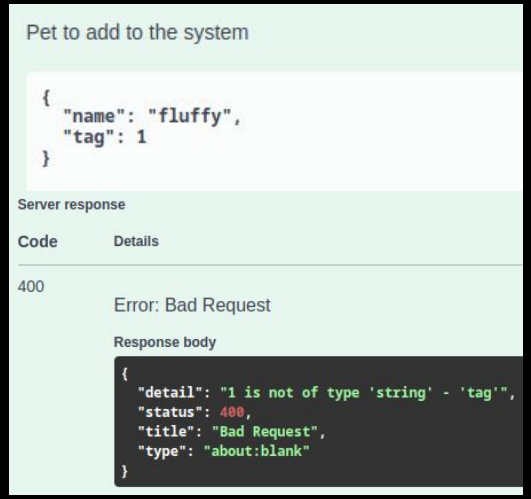
You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

- Validates incoming requests and responses, both body and parameters







You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

- Checks authentication: basic, bearer, API key, Oauth

```yaml
paths:
 /secret:
   get:
     summary: Return secret string
     operationId: app.get_secret
     responses:
       '200':
         description: secret response
     security:
       - api_key: []
components:
 securitySchemes:
   api_key:
     type: apiKey
     name: X-Auth
     in: header
     x-apikeyInfoFunc: app.apikey_auth
```

```python
TOKEN_DB = {"asdf1234567890": {"uid": 100}}


def get_secret(user) -> str:
    return f"You are {user} and the secret is 'wbevuec'"


def apikey_auth(token):
    info = TOKEN_DB.get(token, None)

    if not info:
        raise AuthorizationProblem("Invalid token")

    return info
```

Called first

You can describe your API in as much detail as you want, Connexion guarantees that it will work as you specified.

Based on your specification, connexion:

- Provides mock functionality to test and develop against the specification before implementation is ready or even started
- Routes your API endpoints to your python functions and provides request parameters as function arguments
- Validates incoming requests, both body and parameters
- Can validate responses
- Checks authentication: basic, bearer, API key, Oauth
- Provides a Swagger UI for live documentation and testing

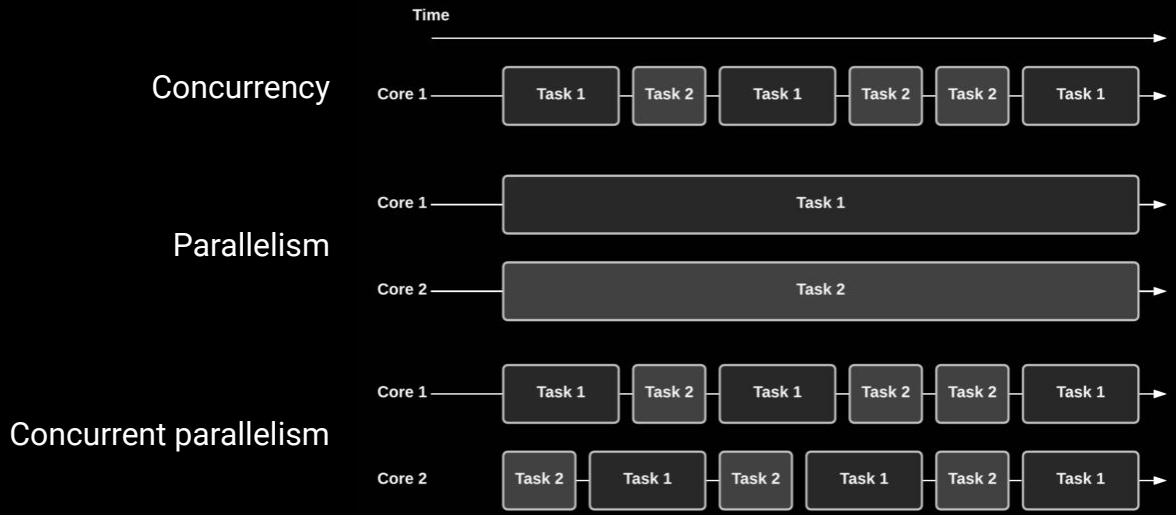
Agenda

Topic
API-first & spec-first
Connexion
Concurrency, parallelism, and async
WSGI, ASGI & Middleware
Working in open source

Concurrency vs parallelism

All about enabling us to handle multiple things at once

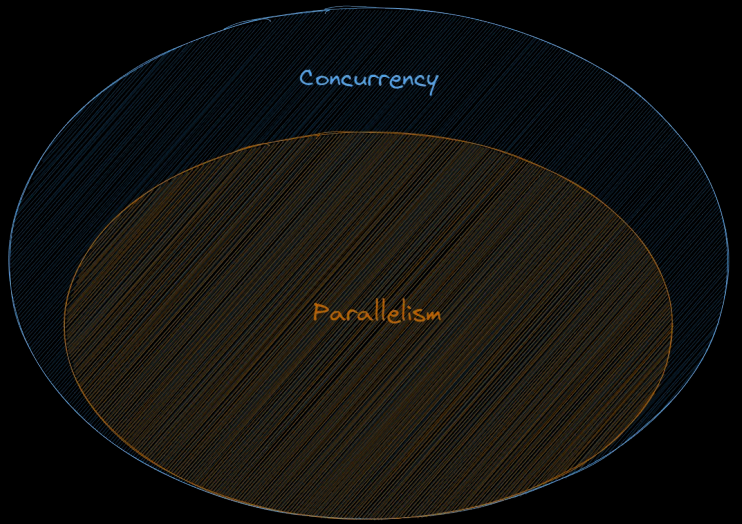
- Parallelism = doing multiple things at the same time.
- Concurrency = dealing with multiple things at the same time



Concurrency vs parallelism

Concurrency is **not** parallelism
However, it ***enables*** parallelism.

If you have only one processor,
your program can still be concurrent
but it cannot be parallel.



Concurrency vs parallelism: an example

2 tasks:

Go to the passport office for a new passport

Prepare a presentation for a guest lecture

Concurrency vs parallelism: an example

- Sequential
  - Go to passport office, wait in line, get passport, go home, work on presentation
- Concurrent
  - Go to passport office, wait in line and start working on presentation, get passport, go home, finish presentation
- Parallel
  - Get your assistant start working on the presentation independently
  - Go to passport office, wait in line, get passport, go home, finish presentation

From concurrency to asyncio in Python

```python
import asyncio


async def do_first():
    print("Running do_first block 1")
    await asyncio.sleep(0)  # Release execution
    print("Running do_first block 2")


if __name__ == "__main__":
    asyncio.run(main())
```

From concurrency to asyncio in Python

```
import asyncio


async def do_first():
    print("Running do_first block 1")
    await asyncio.sleep(0)  # Release execution
    print("Running do_first block 2")


if __name__ == "__main__":
    asyncio.run(main())
```

Async

- I can be awaited
- I'll have to wait on something else to come back to me

From concurrency to asyncio in Python

```python
import asyncio


async def do_first():
    print("Running do_first block 1")
    await asyncio.sleep(0)  # Release execution
    print("Running do_first block 2")


if __name__ == "__main__":
    asyncio.run(main())
```

Await

- I can pause my execution here and give up control to let something else do something while I'm waiting (for the result)
- `await` creates a suspension point that indicates to the event loop that some I/O operation will take place thereby allowing it to switch to another task
- "co-operative multitasking"

```python
import asyncio


async def do_first():
    print("Running do_first block 1")
    await asyncio.sleep(0)  # Release execution
    print("Running do_first block 2")


if __name__ == "__main__":
    asyncio.run(main())
```

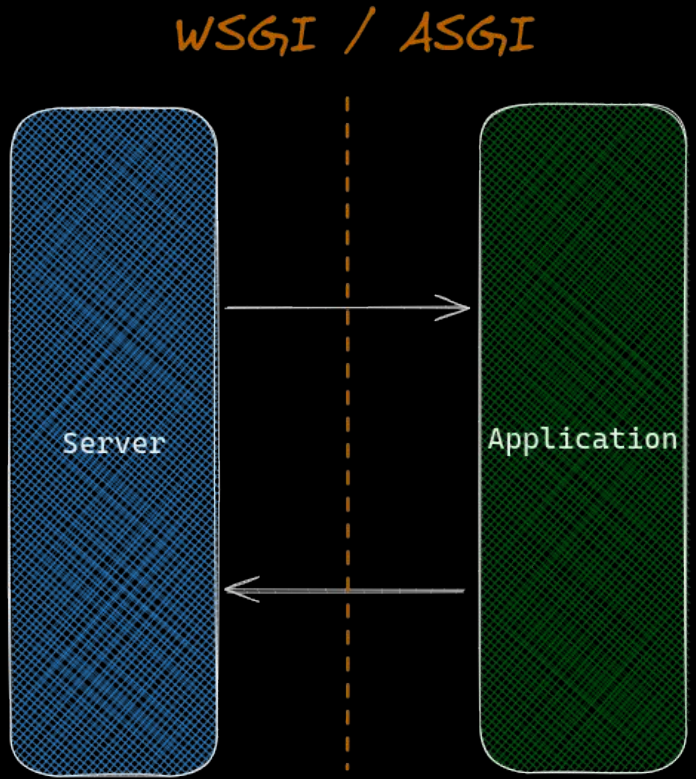
Event loop

- “coordinator”
- loop that monitors coroutines, taking feedback on what’s idle, and looking around for things that can be executed in the meantime

Agenda

Topic
API-first & spec-first
Connexion
Concurrency, parallelism, and async
WSGI, ASGI & Middleware
Working in open source

WSGI = Web Server Gateway Interface



ASGI = Asynchronous Server Gateway Interface

A standard interface between async-capable Python web servers, frameworks, and applications.

Spiritual successor to WSGI (Web Server Gateway Interface)

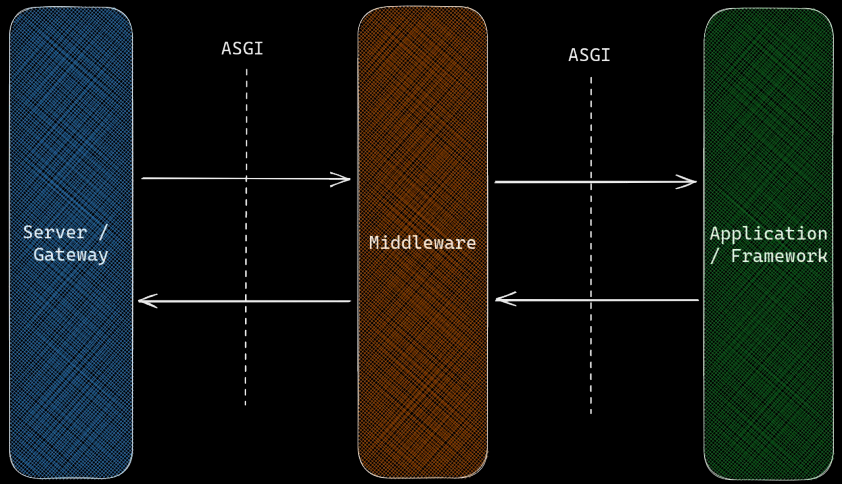
1) Adds support for async / await
2) Adds support for HTTP/2, WebSockets and other network protocols

WSGI = request-response cycle
"call into this function with an HTTP request, and wait until an HTTP response is returned"

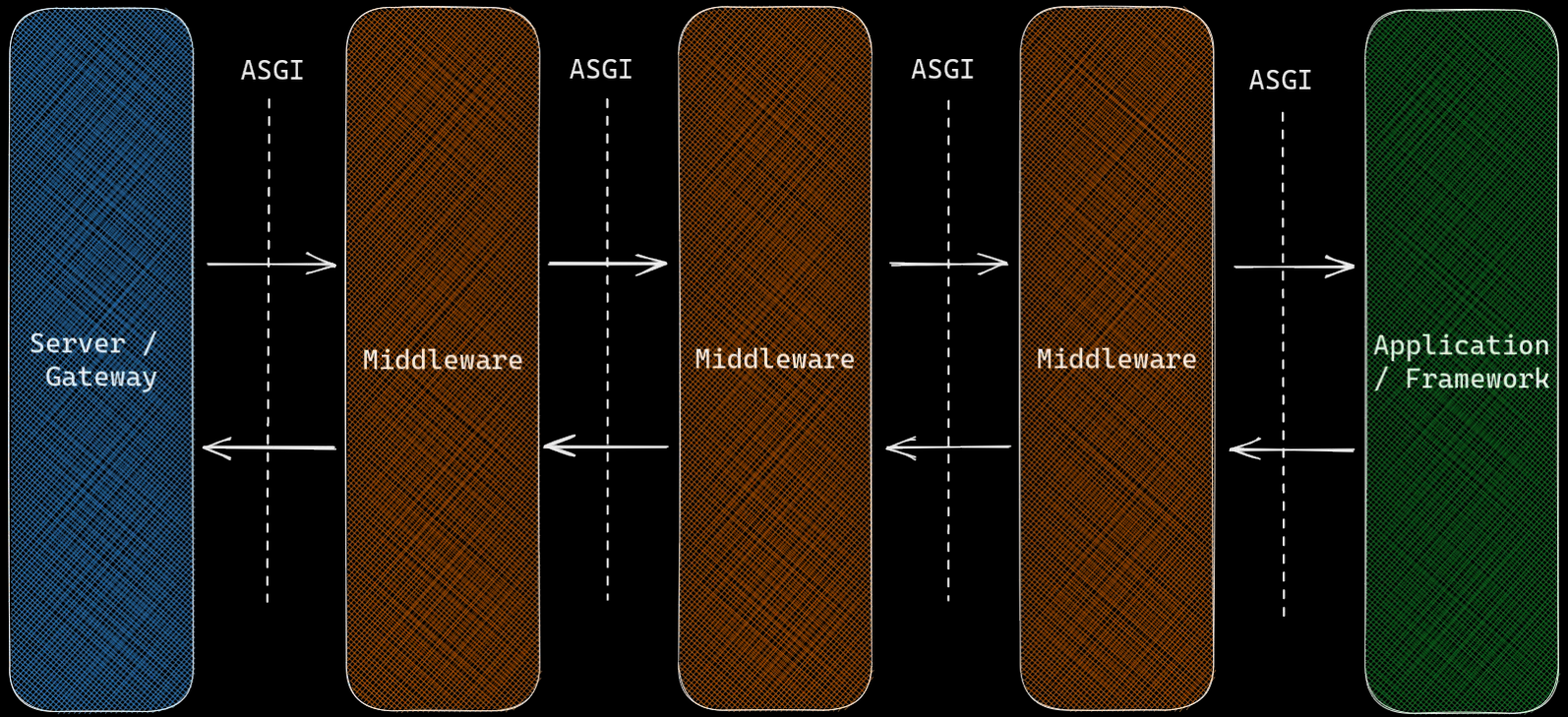
ASGI = receiving and sending events over long-lived connections
"here's a pair of channels between which the server and application can communicate"

Middleware - code that plays the role of both server and application

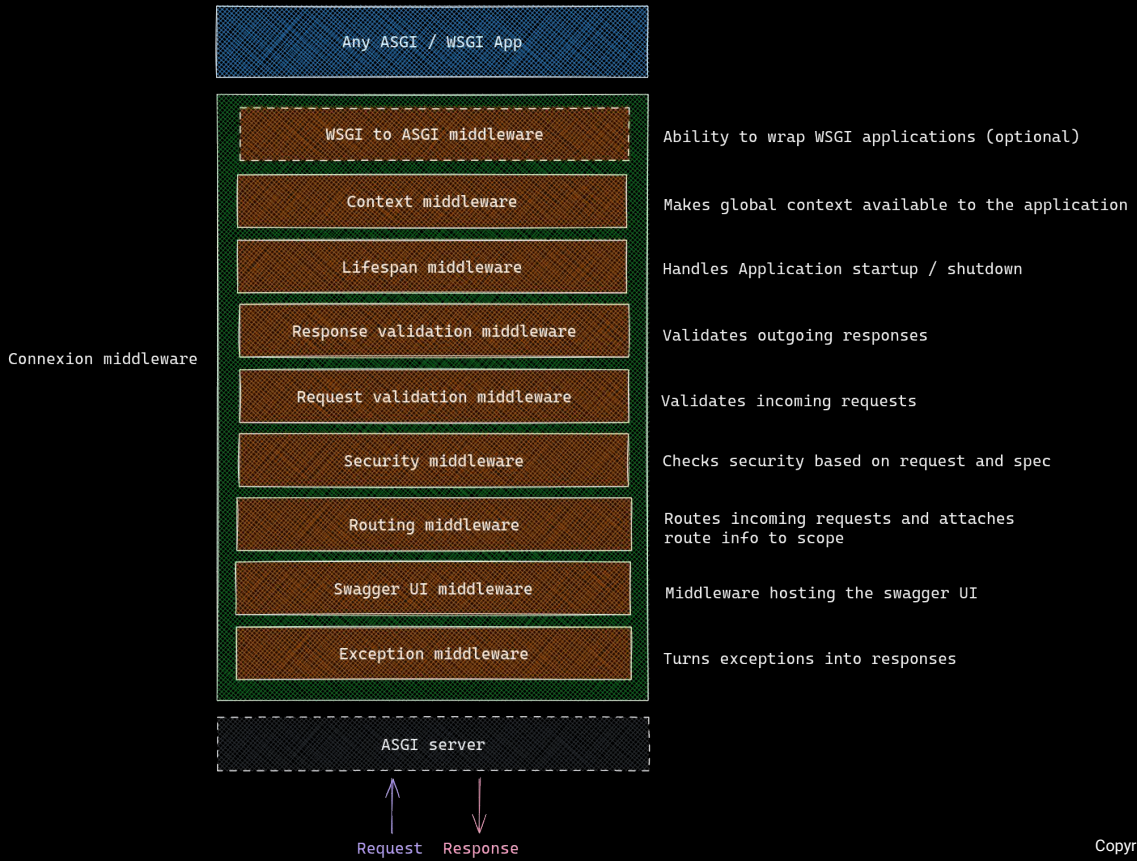
Middleware components play the role of a server with respect to an application, while also acting as an application with respect to a server.



We can keep repeating this pattern

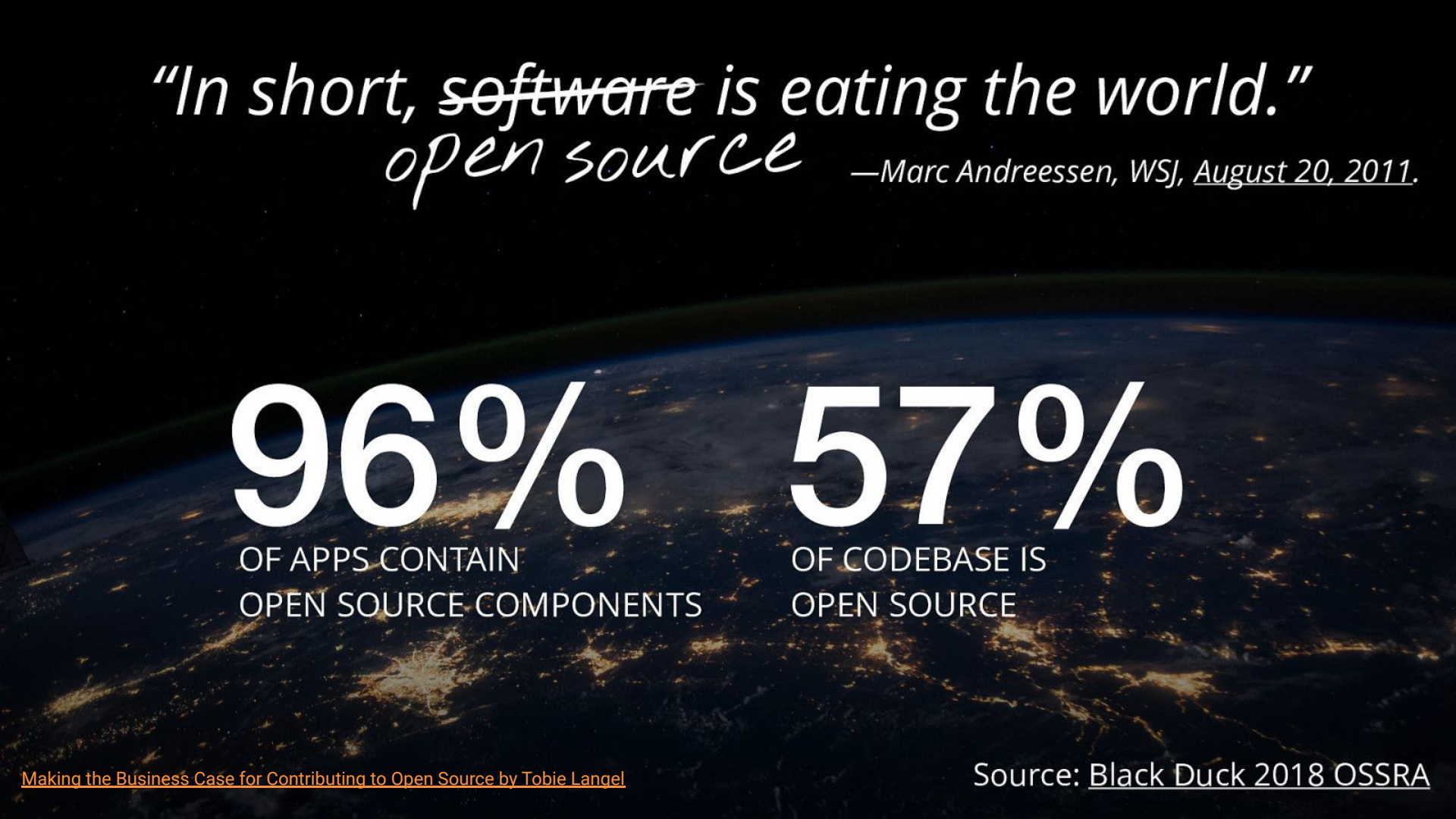


This is how connexion is built



Agenda

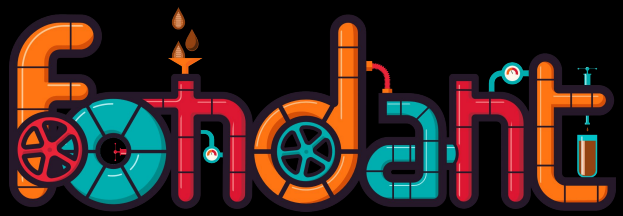
Topic
API-first & spec-first
Connexion
WSGI, ASGI & Middleware
Concurrency, parallelism, and async
Working in open source


"In short, ~~software~~ open source is eating the world."
—Marc Andreessen, WSJ, August 20, 2011.
96%
OF APPS CONTAIN
OPEN SOURCE COMPONENTS
57%
OF CODEBASE IS
OPEN SOURCE
Making the Business Case for Contributing to Open Source by Tobie Langel
Source: Black Duck 2018 OSSRA

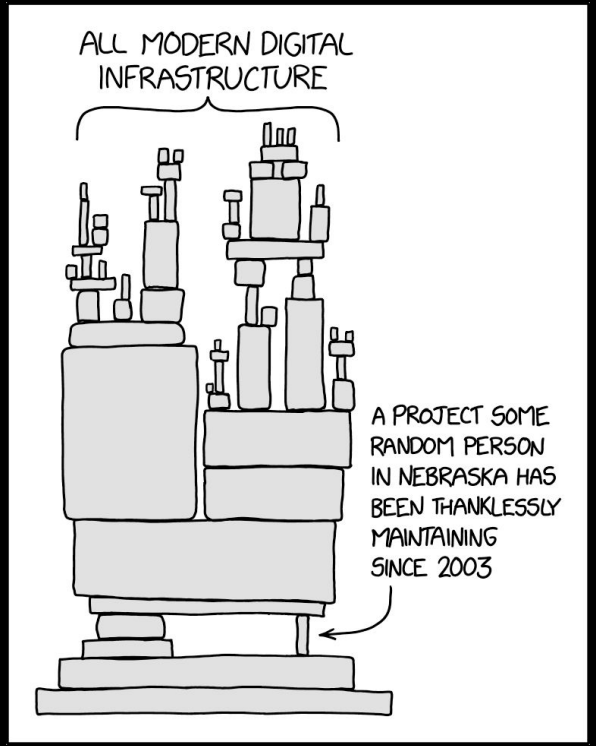
Not all open source is created equal

The “charity” model

The “business” model



The “charity” model has its issues



Learnings from maintaining both types of projects

The “charity” model	The “business” model
Main investment is personal time	Main investment is employee salary
Main priority is protecting limited maintainer time	Main priority is achieving adoption / revenue
Main motivation is satisfaction	Main motivation is monetary
Edge in marketing: trustworthiness	Edge in marketing: budget

Connexion is ~~the community's~~ ours now

Releasing Connexion to the Community

After 6 years and 3.9k GitHub stars, we are releasing Connexion, our API-first Python framework, to the Open Source community.

Henning Jacobs
Executive Principal Engineer

Posted on Feb 11, 2022

Tags: Python, Open Source


ML6