

ML6

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?

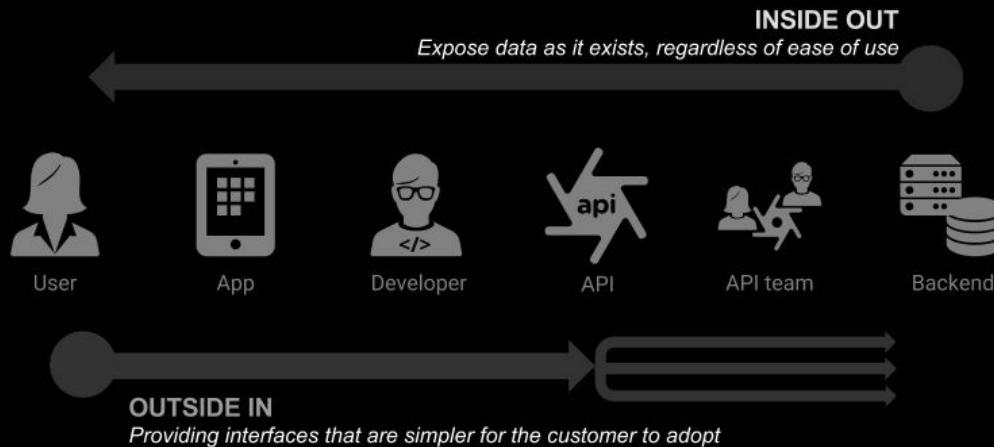
- A product-centric approach to developing APIs, where APIs are treated as “first-class citizens”. APIs are regarded as discrete products instead of integrations, and a set of APIs can form an API platform that fosters innovation.
- Defining and designing APIs and underlying schema before developing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



Sometimes distinguished as spec-first

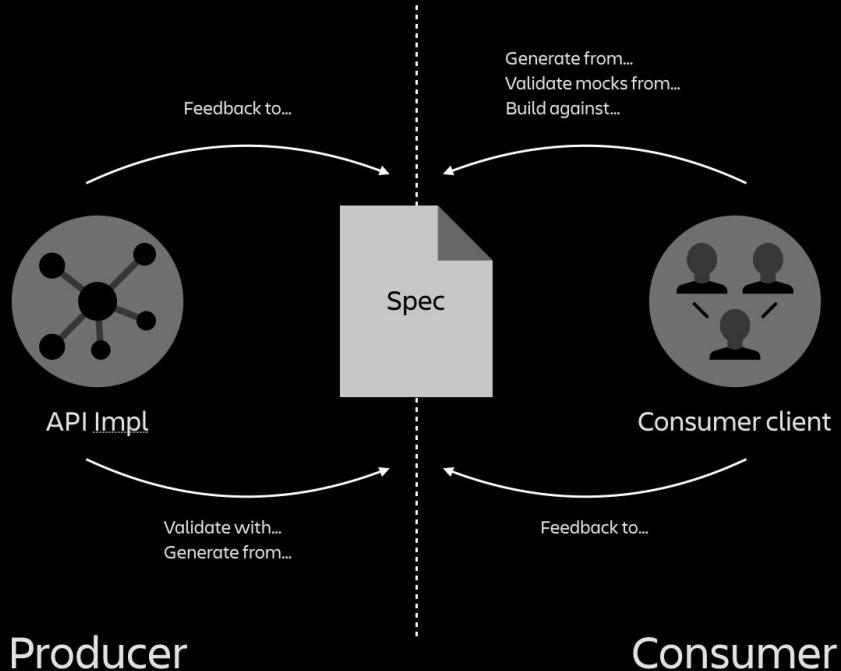
Why API-first?

- 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



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



How to spec-first?



Swagger Editor. File ▾ Edit ▾ Generate Server ▾ Generate Client ▾

```
1 swagger: "2.0"
2   info:
3     description: "This is a sample server Petstore server. You can find out more
4       about Swagger at [http://swagger.io](http://swagger.io) or on [irc
5         .freenode.net, #swagger](http://swagger.io/irc/). For this sample, you
6         can use the api key `special-key` to test the authorization
7         filters."
8     version: "1.0.0"
9     title: "Swagger Petstore"
10    termsOfService: "http://swagger.io/terms/"
11    contact:
12      email: "apiteam@swagger.io"
13    license:
14      name: "Apache 2.0"
15    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
16    host: "petstore.swagger.io"
17    basePath: "/v2"
18    tags:
19      - name: "pet"
20        description: "Everything about your Pets"
21        externalDocs:
22          description: "Find out more"
23          url: "http://swagger.io"
24      - name: "store"
25        description: "Access to Petstore orders"
26      - name: "user"
27        description: "Operations about user"
28        externalDocs:
29          description: "Find out more about our store"
30          url: "http://swagger.io"
31    schemes:
32      - "https"
33      - "http"
34    paths:
35      /pet:
36        post:
37          tags:
38            - "pet"
39          summary: "Add a new pet to the store"
40          description: ""
41          operationId: "addPet"
42          consumes:
43            - "application/json"
44            - "application/xml"
45          produces:
46            - "application/xml"
47            - "application/json"
48          parameters:
```

Swagger Petstore 1.0.0

[Base URL: petstore.swagger.io/v2]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net, #swagger](#). For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes:

pet Everything about your Pets Find out more: <http://swagger.io>

POST	/pet	Add a new pet to the store	<input type="button" value="▼"/>	<input type="button" value="🔒"/>
PUT	/pet	Update an existing pet	<input type="button" value="▼"/>	<input type="button" value="🔒"/>
GET	/pet/findByStatus	Finds Pets by status	<input type="button" value="▼"/>	<input type="button" value="🔒"/>
GET	/pet/findByTags	Finds Pets by tags	<input type="button" value="▼"/>	<input type="button" value="🔒"/>

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



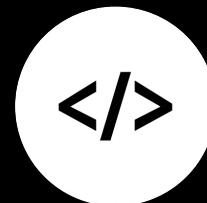
Starred 4.4k

Connexion



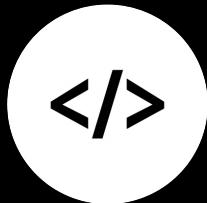
Contract

v



Code

Everyone else



Code

v



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

The screenshot shows the Swagger Petstore UI version 1.0.0. At the top, there's a dropdown for 'Schemes' set to 'HTTPS' and a green 'Authorize' button with a lock icon. Below this, a section titled 'pet' contains a list of endpoints:

- POST /pet** Add a new pet to the store
- PUT /pet** Update an existing pet
- GET /pet/findByStatus** Finds Pets by status
- GET /pet/findByTags** Finds Pets by tags
- GET /pet/{petId}** Find pet by ID
- POST /pet/{petId}** Updates a pet in the store with form data
- DELETE /pet/{petId}** Deletes a pet
- POST /pet/{petId}/uploadImage** uploads an image

Each endpoint row has a small lock icon at the end, indicating security information.

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

POST /pet/{petId} Updates a pet in the store with form data

Parameters

Name	Description
petId <small>* required</small> <small>integer(\$int64) (path)</small>	ID of pet that needs to be updated petid
name <small>string (formData)</small>	Updated name of the pet name
status <small>string (formData)</small>	Updated status of the pet status

Try it out

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

POST /pets Create a pet

Parameters

No parameters

Request body

Pet to add to the system

Example Value | Schema

Pet ▼ {
 name*: string
 example: fluffy
 tag: string
 example: red
}

Pet to add to the system

```
{ "tag": "red" }
```

Server response

Code	Details
400	Error: Bad Request

Response body

```
{
    "detail": "'name' is a required property",
    "status": 400,
    "title": "Bad Request",
    "type": "about:blank"
}
```

Pet to add to the system

```
{
    "name": "fluffy",
    "tag": 1
}
```

Server response

Code	Details
400	Error: Bad Request

Response body

```
{
    "detail": "1 is not of type 'string' - 'tag'",
    "status": 400,
    "title": "Bad Request",
    "type": "about:blank"
}
```

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

```
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
```

```
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

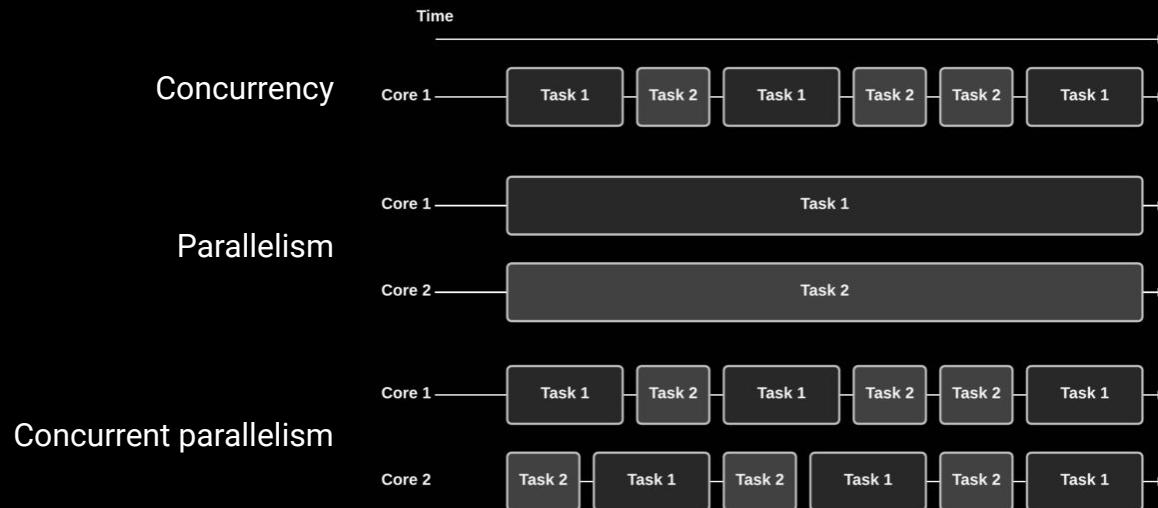
WSGI, ASGI & Middleware

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

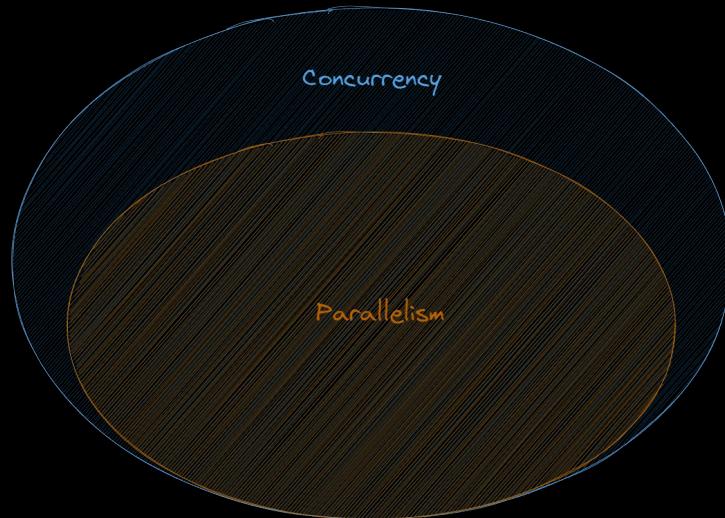


Concurrency vs parallelism

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

```
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

```
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”

```
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

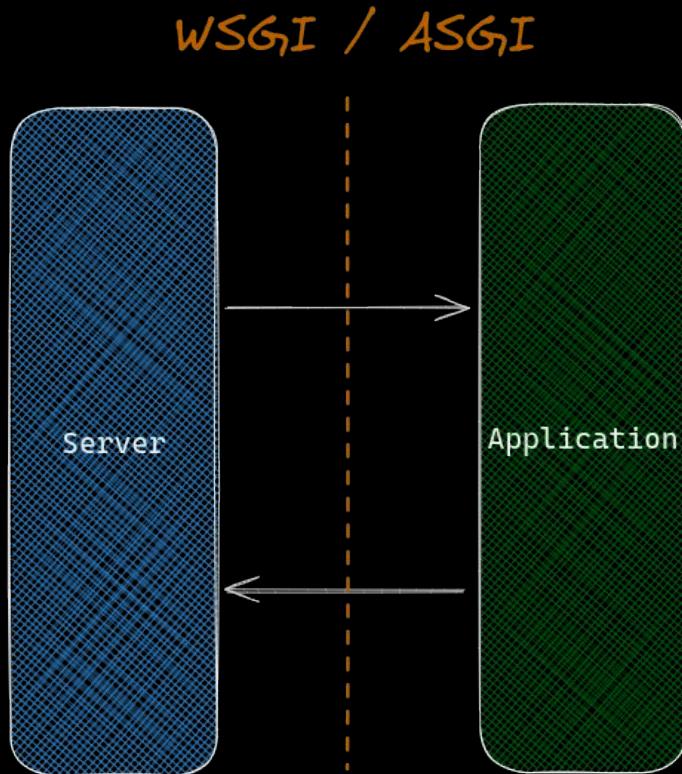
Connexion

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

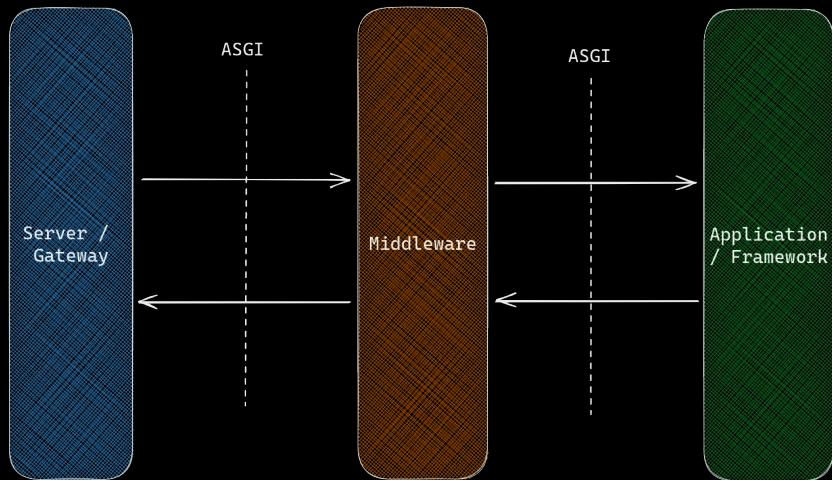
“call into this function with an HTTP request, and wait until an HTTP response is returned”

ASGI = receiving and sending events over long-lived connections

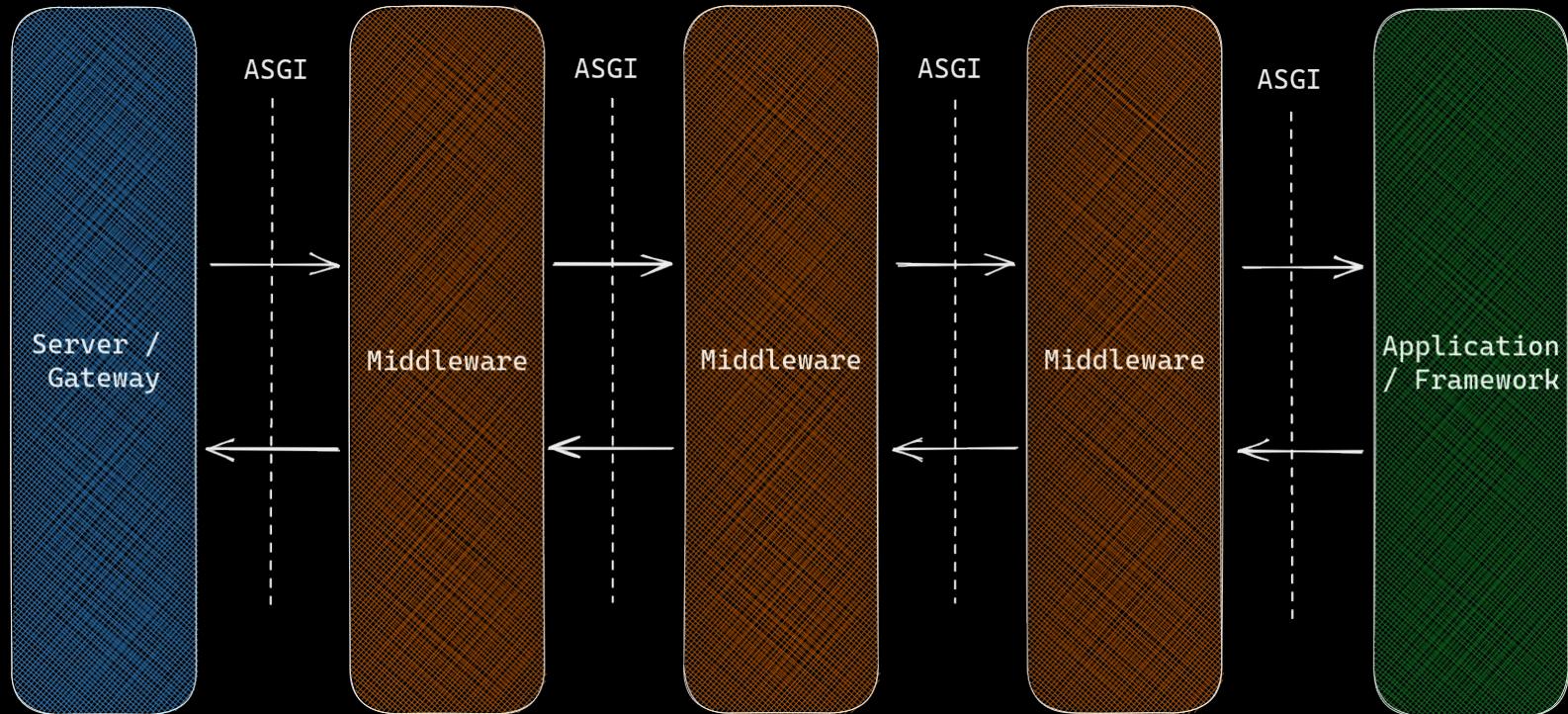
“here’s a pair of channels between which the server and application can communicate”

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

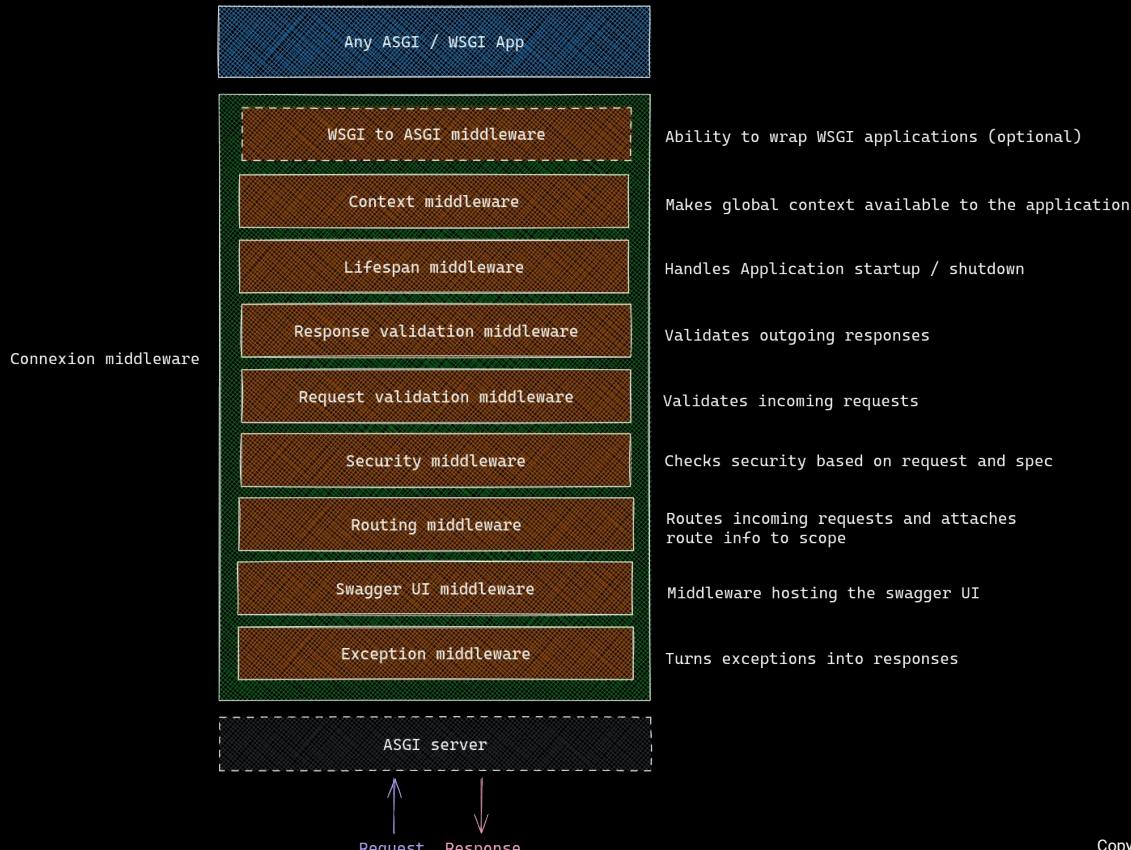
Middleware components play the role of a server with respect to an application, while also acting as an application with respect to a server.



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 is eating the world."
open source*

—Marc Andreessen, WSJ, August 20, 2011.

96%
OF APPS CONTAIN
OPEN SOURCE COMPONENTS

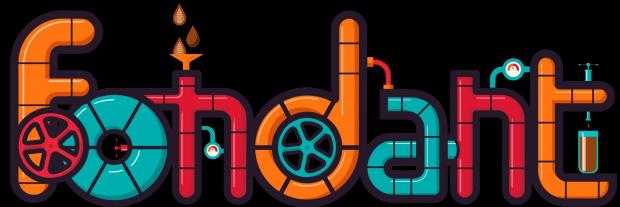
57%
OF CODEBASE IS
OPEN SOURCE

Not all open source is created equal

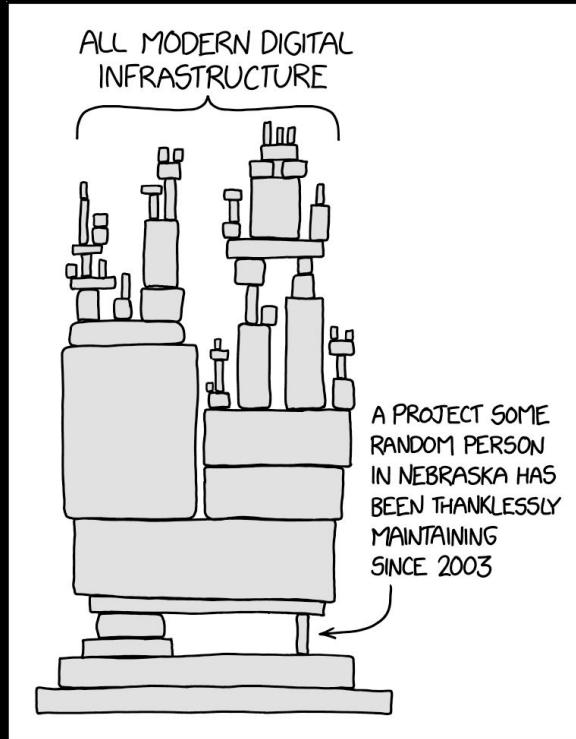
The “charity” model



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