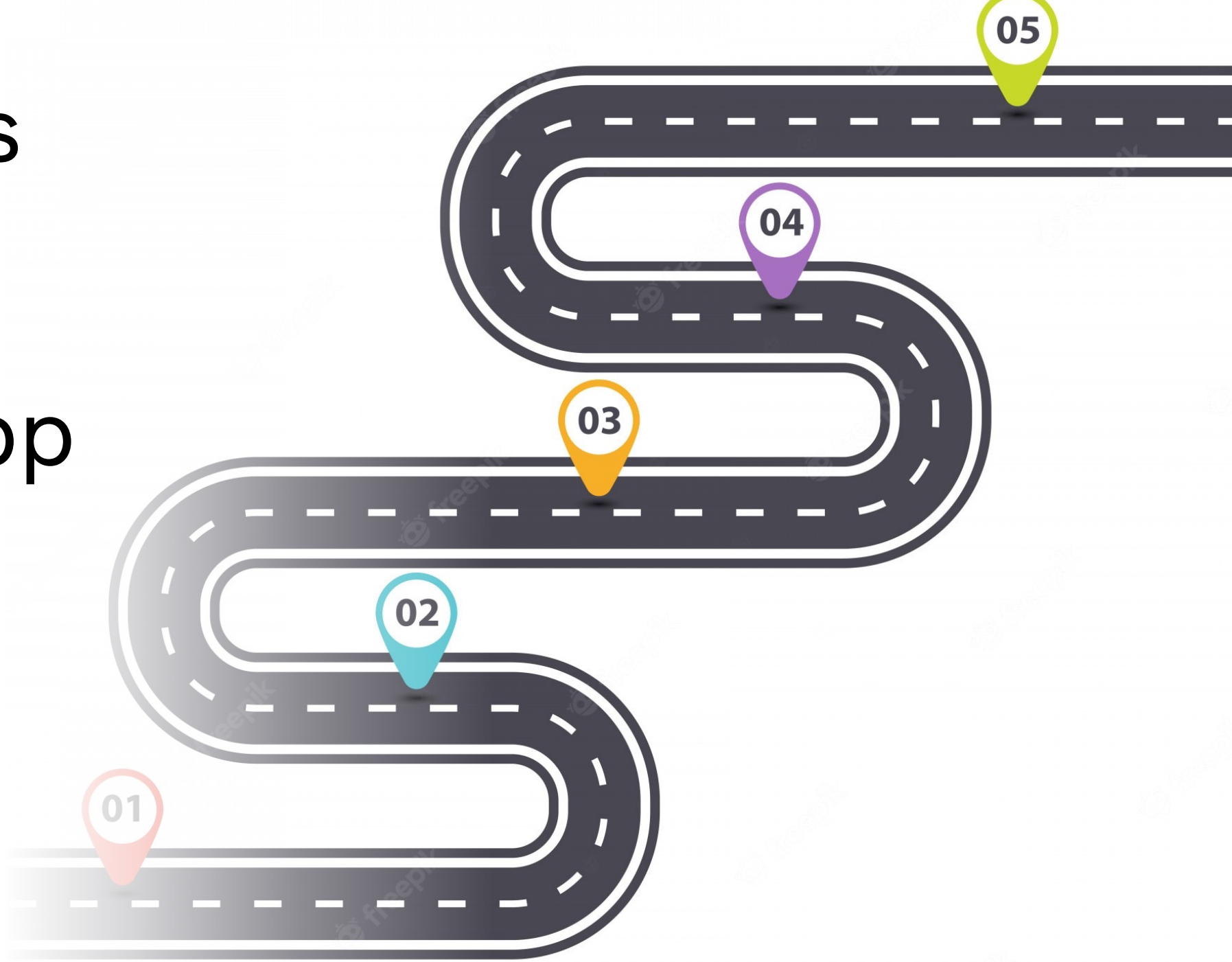
Baby Steps Towards Your First FastAPI App

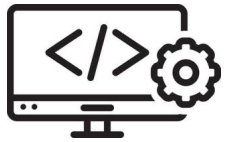
Shreya Khurana

Data Scientist @ GoDaddy



PyOhio

What is FastAPI?



Web framework

Python 3.6+

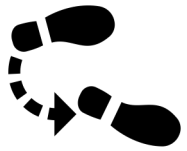
Fast (duh!)

- Comparable to Node, Go
- Starlette + Uvicorn → async request capability

OpenAPI compatible

- Generate Swagger documentation in a jiffy!

Why (not?) Flask?



Why I moved from Flask to FastAPI

- Quick Swagger generation!
- DO NOT give me more libraries to install!

Flask

FastAPI

- More number of years in development
- More plugins, extensions

- Async ops
- Code autocomplete possible with Pydantic

Let’s start simple

```python
from fastapi import FastAPI

app = FastAPI()


@app.get('/persons/{name}')
async def vecna_say_hi(name):
    return {'Vecna says, Hi {}'.format(name)}


@app.get('/persons/{name}')
async def vecna_say_hi(name: str):
    return {'Vecna says, Hi {}'.format(name)}


@app.get('/eggos/')
async def number_of_eggos(num: int = 1, max_num: int = 10):
    return {'El likes {} eggos, but feed her {} max!'.format(num, max_num)}
```

Path parameters
- You see them in your path

Path Parameters

```python
from fastapi import FastAPI

app = FastAPI()


@app.get('/persons/{name}')
async def vecna_say_hi(name):
    return {'Vecna says, Hi {}'.format(name)}

@app.get('/persons/{name}')
async def vecna_say_hi(name: str):
    return {'Vecna says, Hi {}'.format(name)}

@app.get('/eggos/')
async def number_of_eggos(num: int = 1, max_num: int = 10):
    return {'El likes {} eggos, but feed her {} max!'.format(num, max_num)}
```

Declare data types

Query Parameters

```python
from fastapi import FastAPI

app = FastAPI()


@app.get('/persons/{name}')
async def vecna_say_hi(name):
    return {'Vecna says, Hi {}'.format(name)}


@app.get('/persons/{name}')
async def vecna_say_hi(name: str):
    return {'Vecna says, Hi {}'.format(name)}


@app.get('/eggos/')
async def number_of_eggos(num: int = 1, max_num: int = 10):
    return {'El likes {} eggos, but feed her {} max!'.format(num, max_num)}
```

Query parameters
- You don't see them in path
- But present in your url call
- key-value pairs

Optional Query Parameters

```python
from typing import Union

@app.get('/persons/{name}/eggos/')
async def number_of_eggos(
    name: str, num: int = 1, strange_character: Union[bool, None] = None,
):
    msg = '{} likes {} eggos!'.format(name, num)
    res = {'message': msg}
    if strange_character:
        res.update({'is_strange_character': strange_character})
    return res
```

Data Validation

```python
from enum import Enum

class StrangeCharacter(str, Enum):
    mike = 'mike'
    el = 'el'
    dustin = 'dustin'
    will = 'will'
    lucas = 'lucas'

@app.get('/persons/{name}')
async def vecna_says_hi(name: StrangeCharacter):
    if name in (StrangeCharacter.mike, StrangeCharacter.lucas, StrangeCharacter.dustin):
        return {'Vecna says, Hi {}'.format(name)}
    else:
        return {'Vecna says, Bye {}'.format(name)}
```

Validate data yourself!

- Possible values as Enum

Pydantic

```python
from pydantic import BaseModel
from typing import List


class EggoEater(BaseModel):
    name: str
    hobbies: Union[List[str], None] = None
    num_eggos: int


eggo_eaters = {'mike': {'name': 'Mike', 'num_eggos': 1, 'hobbies': ['D&D', 'writing to El']},
               'el': {'name': 'Jane', 'num_eggos': 2, 'hobbies': ['controlling things with mind',
                      'writing to Mike']},
               'will': {'name': 'Will', 'num_eggos': 0},
               'dustin': {'name': 'Dustin', 'num_eggos': 1, 'hobbies': ['D&D', 'hacking into school
                          computer systems']},
               'lucas': {'name': 'Lucas', 'num_eggos': 2}}


@app.get('/persons/{name}', response_model=EggoEater)
async def strange_character_eats_eggos(name: str):
    return eggo_eaters[name]
```

- Data validation using python type annotations.
- Enforces type hints at runtime
- Provides user friendly errors when data is invalid.

Pydantic

Response

Code | Details

200

Response body

```
{
  "name": "Mike",
  "hobbies": [
    "D&D",
    "writing to El"
  ],
  "num_eggos": 1
}
```

Response headers

Response Model

Schemas

EggoEater {

name*	string *title: Name*
hobbies	Hobbies [*title: Hobbies* string]
num_eggos*	integer *title: Num Eggos*

}

Pydantic

```python
@app.get('/persons/{name}', response_model=EggoEater, response_model_exclude={'hobbies'})
async def strange_character_eats_eggos(name: str):
    return eggo_eaters[name]


@app.get('/persons/{name}', response_model=EggoEater, response_model_include={'num_eggos'})
async def strange_character_eats_eggos(name: str):
    return eggo_eaters[name]
```

Exclude certain fields

Include only certain fields

Exception handling

Parameters

Name	Description
name * required string (path)	ksns

Code	Details
500 Undocumented	Error: Internal Server Error Response body Internal Server Error

```python
from fastapi import HTTPException


@app.get('/persons/{name}', response_model=EggoEater)
async def strange_character_eats_eggos(name: str):
    if name not in eggo_eaters:
        raise HTTPException(status_code=404, detail="Strange Character not found!")
    return eggo_eaters[name]
```

Code	Details

404
Undocumented
	Error: Not Found
	Response body
	{
	"detail": "Strange Character not found!"
	}
	Response headers

PyOhio

Logging

```python
from datetime import datetime
import logging
from logging.handlers import RotatingFileHandler
from fastapi.logger import logger as fastapi_logger
from fastapi import Request


formatter = logging.Formatter('%(asctime)s | %(levelname)s | %(name)s | %(message)s', '%Y-%m-%d %H:%M:%S')
logging.getLogger().setLevel(logging.INFO)
handler = RotatingFileHandler('fastapi_access.log', backupCount=0)
fastapi_logger.addHandler(handler)
handler.setFormatter(formatter)
fastapi_logger.info('****************** Starting Server *****************')
```

```python
@app.middleware('http')
async def log_middle(request: Request, call_next):
    response = await call_next(request)
    fastapi_logger.info(
        f"{datetime.now().strftime('%Y-%m-%d %H:%M:%S')} {request.method} {request.url} STATUS: {response.status_code}")
    return response
```

Middleware

- with every **request** before it is processed
- with every **response** before returning it

Helpful Materials

- https://fastapi.tiangolo.com/
- https://testdriven.io/blog/moving-from-flask-to-fastapi/
- https://swagger.io/specification/

Thank you!

Acknowledgements:
Team @ GoDaddy
Organizers @ PyOhio 2022