Restful API And Microservices with Python

Day 1

Day 1 - Overview

- Virtualenvs and setting up Flask-RESTful
- Your first Flask-RESTful app
- Creating our TODO Item Resource
- The TODO Item List and creating TODO Items
- PUT to update TODO Items
- DELETE to delete TODO Items
- Test-first API design—what is that?
- Improving code and error control

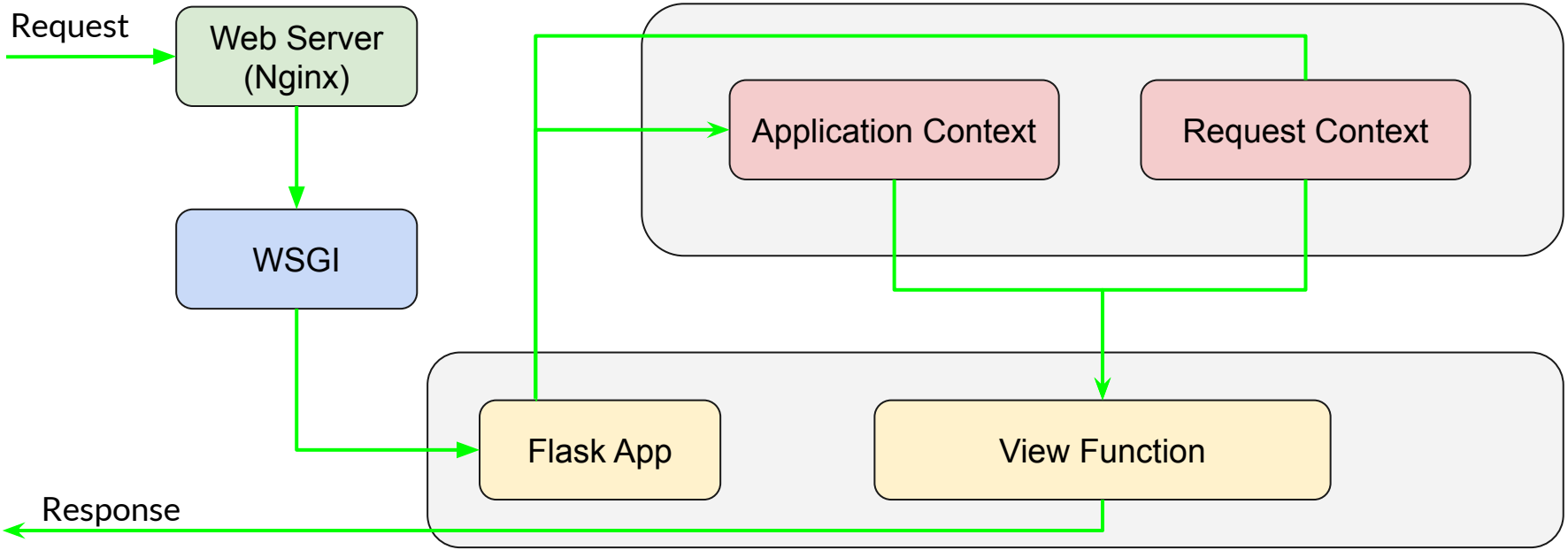
Prerequisite

- VM with windows OS
- Python 3.8 or >
- Visual Studio Code - Code Editor
- Postman
- Docker - Not Mandatory for current training

Why Flask?

- Lightweight micro framework
- Easy to scale
- Flexibility to alter because of simplicity.
- Large community support.
- Easy to integrate with AI and ML models.

Architecture of Flask Framework



Setting up the application and environment

- Create a folder called **workspace** where we will keep all the code repositories.
- Create a folder called **todo-flask-restful** inside **workspace** folder.
- Open Visual Studio Code editor and open **todo-flask-restful** folder.
- Create a file called **requirements.txt** to hold all dependencies and version. Add below content to

```
Flask==2.2.0
Flask-RESTful==0.3.9
```

- Create a virtual environment to isolate dependency creation w.r.t this application.

```
python -m venv todoenv
todoenv\Scripts\activate
```

- Install dependencies

```
pip install -r requirements.txt
```

Your first Flask Restful application

```
from flask import Flask, request
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

todoData = [
    {"id": 1, "name": "File ITR", "status": "STARTED"},
    {"id": 2, "name": "Complete Flask microservices", "status": "NEW"},
]

class ToDo(Resource):
    def __init__(self):
        Pass

    def get(self):
        return todoData

api.add_resource(ToDo, '/api/todos')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8001, debug=True)
```

Running and Debugging a Flask Restful Application

- Open terminal in Visual Studio code
- Execute below command

```
flask run                          #to start in default setting

flask -debug run -port 5001        #to enable debugging and specify server port

python app.py                      #specify details in source code
```

- Check the difference between debug and non debug mode.
- Hit the get endpoint from browser or postman

Creating a ToDO Resource using Flask-Restful

- Create a ToDo Resource - Completed in previous section
- Register the resource with api - Completed in previous section
- Create a GET call to list all available ToDos - Completed in previous section
- Create a POST call to create a new ToDo and add to the existing list.
- Create a PUT call to update an existing ToDo
- Create a DELETE call to remove an existing ToDo - **Task**

Test-First API design

Each new API/Resource created should have an corresponding unittest class that can validate the basic contract and functionality of the view function of the API.

- Create a unit test case for ToDo API - **test_todos.py**
- Create a test case for GET call to validate the HTTP status code and size of list response.
- Create a test case for POST call to validate the HTTP status code and size of list response.
- Create test case for PUT and DELETE - **Take home assignment**

Command: python -m unittest test_todos.py

Error Handling

At the moment the REST interfaces are ill equipped to handle error scenarios like below

- Duplicate entry
- Non existent Item
- Constraint violation

Create a custom exception class **ToDoAlreadyExists** that extends HTTPException in **exceptions.py**

Create a custom Api class to handle custom exception.

Update the POST call to reject a call with **ToDoAlreadyExists** exception is the new ToDo item name is same as that of an existing item.

Q and A