



Restful API And Microservices with Python

Day 3 & 4



Day 3 & 4 - Overview

- Pending items from Day 2 (sqlite CRUD operation)
- Improving the project structure and maintainability
- Setting up this section's project
- Advanced request parsing with Flask-RESTful
- Optimizing our final code and request parsing



Prerequisite

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



CRUD operation on ToDo items in sqlite database

- Import the below postman collection

<https://www.getpostman.com/collections/a682a18465106586dc51>

- Code explanation
- Execute the GET API to fetch list of items from the database
- Execute the POST API to create new item in the todo database
- Execute the PUT API to update an existing item in the database
- Execute the DELETE API to remove an existing item in the database.



Improving the project structure

- Fork the below repository

`git clone https://github.com/saurav-samantray/flask-microservices-training/`

- Clone the forked repository **flask-microservices-training**

`git clone https://github.com/<replace-with-your-git-username>/flask-microservices-training/`

- Open the new flask-microservices-training inside visual studio code.
- Open a new terminal and go to below location

`C:\workspace\flask-microservices-training\day3\user-management-service`

- Create and activate virtual environment

`python -m venv venv`

`./venv/Scripts/activate`

- Install dependencies

`pip install -r requirements.txt`

```
| init_db.py
| requirements.txt
| schema.sql
| server.py
| user-management.db
|
+---app
| | config.py
| | exceptions.py
| | __init__.py
| |
+---api
| | addresses_api.py
| | users_api.py
| | user_api.py
| |
+---database
| | user_db.py
| | __init__.py
| |
+---models
| | address.py
| | user.py
| |
+---test
| | test_users.py
```



Advanced request parsing with Flask-RESTful

- Request payload validation
- Schema handling using marshmallow
- Mandatory parameter validation
- Length and Range validation



Optimizing our final code

- Delegating logic to individual resource related files
- Creating models
- Model serialization
- Model Deserialization
- Setting environment specific configuration

`$Env:FLASK_ENV = 'development'`

- Initialize the database using the below command

`python init_db.py`

- Start the server using below command

`python server.py`



Q and A