# RESTful APIs with PyFlask

Introduction and Demo

UULM 2024
HACKATHON X GAMEJAM

# RESTful APIs with PyFlask

Part 1 Part 2 Part 3

UULM 2024
HACKATHON X GAMEJAM

# REST

- acronym for **RE**presentational **S**tate **T**ransfer
- architectural style for **distributed hypermedia systems**
- **Six Guiding Principles**

| | |
|---|---|
| Uniform Interface | Cachable |
| Client-Server | Layered System |
| Stateless | COD |

# Principles

Client-Server Architecture: Operate independently; Client: UI/UX, Server: Backend

Statelessness: Each Request contains all infos, Server stores no session information, improves scalability

Cacheability: Responses explicitly marked as (non-)cachable

Uniform Interface: Standarized Way of interacting with resources
Resource Identification Resource
Representation
sELF-DESCRIPTIVE mESSAGES
Hypermedia as the Engine of Applicatoin State

Layered System: Hierarchical, Specific Function per layer

Code on Deman: Extend client functionality by transferring executable code

# Resources in REST

Key Abstraction of information
E.g. document, image, temporal service…

Resource Representation: State of the resource at any particular time
Consists of:
- data
- metadata describing the data
- hypermedia links to transition to next desired state

# Resources in REST

- Resource Identifiers: identify each resource involved in an interaction
- Media Type: data format, identifies specification on how to process data
- Self Descriptive: Client act solely based on the media type
- -> custom media type for each resource

# Resource Methods

Standard methods to for interacting with resources
Typically mapped to HTTP Methods

- GET (Read)
- POST(Create)
- PUT(Update/Replace)
- PATCH(Partial Update)
- DELETE
- HEAD (Metadata Retrieval)
- OPTIONS (Capabilities Discovery)

# APIs

Application Programming Interface

- Set of rules that allow one application to communicate with another
- Enables data exchange and application integration

# Types of APIs

- RESTful APIs
- SOAP APIs
- GraphQL APIs
- PRC APIs

# API Status Codes

Most Important

- 200 : OK
- 201 : Created
- 400 : Bad Request
- 404 : Not Found
- 500 : Internal Server Error

Collection of Status Codes -> https://http.cat/

# PyFlask

- Lightweight web Framework for Python
- Designed for simplicity and flexibility

- Pros:
  - Minimal Setup
  - Suitable for small to medium-sized projects
  - Extensive documentation
  - Easy setup via : pip install flask

# Core Concepts

- Routes
- Views
- Templates
- Handling Forms and Requests

# Defining Routes

Define a Route for each REST Resource Method for each Endpoint

E.G.: Define a endpoint for sentence Tokenization

```python
@app.route("/tokenize", methods=["POST"])
def tokenize_sentence():
    sentence = request.data.decode("UTF-8")
    tokens = _tokenize(sentence)
    return jsonify({"tokens":tokens}), 201

def _tokenize(sentence : str):
    return [i for i in sentence.split(" ")]
```

# PyFlask Extensions

Important Extensions:

- Flask-CORS : Manages Cross Origin Resource Sharing
- Flask-SQLAlchemy : Provides SQLAlchemy Integration
- Flask-Login : Handels user session and authentication
- Flask-Cache : Adds caching support

# Alternative: FastAPI

- Modern high-performance web framework for API creation

- Website: https://fastapi.tiangolo.com/

- Source Code: https://github.com/fastapi/fastapi


FastAPI FastAPI

UULM 2024 HACKATHON X GAMEJAM

# FastAPI - Features

- Speed : Supports Async/Await
- Type Safety : Supports Pydantic Models -> data parsing/validation
- Easy to Code : Syntax optimized for fast creation of APIs (200-300% Better!!)

FastAPI FastAPI

UULM 2024 HACKATHON X GAMEJAM

# FastAPI - Code

- Similar to pyflask
- Resource Method directly in the Decorator
- Automatic JSON Parsing via Pydantic Model

```python
## Pydantic Model for request data
class Item(BaseModel):
    sentence : str

@app.post("/tokenize")
def tokenize_item(item : Item):
    tokens = _tokenize(item.sentence)
    return tokens


def _tokenize(sentence : str):
    return sentence.split(" ")
```

FastAPI FastAPI

UULM 2024 HACKATHON X GAMEJAM

# Comparison PyFlask - FastAPI

## PyFlask


- Lightweight
- Synchronous
- jsonify()
- Microservices / simple APIs
- CORS extensions
- Large Community

## FastAPI


- High performance
- Asynchronous
- Automatic JSON serialization
- Microservices / Real-Time App
- Built-in CORS handling
- Rapidly Growing Community

# Thank you for your Attention