2. Approaches to create Request Body and Parameters - 1

Post request

1. Request **Payload/Body** (JSON format)
2. Send Request (Post URL)
3. Response Validations

Different ways of creating Request Payload / Request Body

➔ We can Request Payload / Request Body by using

- **Dictionary**
- **Json Module**
- **Using a Python Class (like POJO(Plain Old Java Object) class)**
- **@dataclass decorator in dataclass Python**
- **External json file**

Pytest Fixtures

➔ **fixtures** are functions that manage the setup and teardown process for test environments. Fixtures allow you to define code that needs to **run before a test (setup)** and **after a test (teardown).**

test_FixtureFunction.py

```
import pytest
@pytest.fixture()   # decorator
def setup():
  print("Launching browser...") #Executes once before every test method
  yield
  print("Closing browser..") #Executes Once after every test method
class TestClass:
 def test_Login(self, setup):
    print("This is login test")
 def test_Search(self, setup):
    print("This is search test")
```

Workout in Pycharm

➔ Create New Package or Directory and place **students.json file** in the same package.

➔ Start students API from Pycharm Terminal using below command

- **json-server --watch students.json**
- Open http://localhost:3000/students in any browser

test_PostRequestBodyExamples.py

```
import json, pytest, requests
from dataclasses import dataclass, asdict
BASE_URL = "http://localhost:3000/students"   #Global variable
student_id = None   #Global variable
request_headers = {"Content-Type": "application/json"}
```

Dictionary

➔ Used when data is simple and already available in key-value pairs (e.g., login credentials, form submissions)

Test to create Student using Dictionary

```
def test_createStudentUsingDictionary():
  global student_id
  request_body = {
     "name": "Scott",
     "location": "France",
     "phone": "123456",
     "courses": ["C", "C++"]
  }
  response = requests.post(BASE_URL,json=request_body)
                                        or
  response = requests.post(BASE_URL, data=json.dumps(request_body), headers=request_headers)
  assert response.status_code == 201, "Status code is not 201"
  response_body = response.json()
  assert response_body["name"] == "Scott", "Name is not correct"
  assert response_body["location"] == "France", "Location is not correct"
  assert response_body["phone"] == "123456", "Phone is not correct"
  assert response_body["courses"][0] == "C", "Course 1 should be C"
  assert response_body["courses"][1] == "C++", "Course 2 should be C++"
  student_id = response_body["id"]
  print(response.json())
```

Note

- ➔ By default, data= sends form data (key=value&key=value) — but your server (json-server) expects raw JSON, not form data.
- ➔ Use **data= with json.dumps()**, and make sure the **Content-Type is "application/json"**.
- ➔ Just don’t mix formats