**Hướng dẫn chi tiết API Test Automation với Postman**

**Bước 1: Khởi động Spring Boot để tạo API**

**Bước 2: Setup Postman Collection**

**2.1 Import Collection vào Postman**

**Cách 1: Import từ file JSON**

1. **Mở Postman Desktop hoặc Web**
2. **Click nút "Import" (góc trái trên)**
3. **Chọn tab "Raw text"**
4. **Paste JSON collection đã chuẩn bị:**

**{**

**"info": {**

**"name": "API Test Collection",**

**"description": "Collection for API automation testing",**

**"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"**

**},**

**"item": [**

**{**

**"name": "Get All Users",**

**"request": {**

**"method": "GET",**

**"header": [],**

**"url": {**

**"raw": "{{baseUrl}}/api/users",**

**"host": ["{{baseUrl}}"],**

**"path": ["api", "users"]**

**}**

**},**

**"event": [**

**{**

**"listen": "test",**

**"script": {**

**"exec": [**

**"pm.test('Status code is 200', function () {",**

**" pm.response.to.have.status(200);",**

**"});",**

**"",**

**"pm.test('Response is JSON', function () {",**

**" pm.response.to.be.json;",**

**"});",**

**"",**

**"pm.test('Response contains users array', function () {",**

**" const jsonData = pm.response.json();",**

**" pm.expect(jsonData).to.be.an('array');",**

**"});"**

**],**

**"type": "text/javascript"**

**}**

**}**

**]**

**}**

**],**

**"variable": [**

**{**

**"key": "baseUrl",**

**"value": "http://localhost:8080",**

**"type": "string"**

**}**

**]**

**}**

1. **Click "Import"**
2. **Verify collection xuất hiện trong sidebar**

**Cách 2: Import từ file**

1. **Lưu JSON vào file (ví dụ: api-collection.json)**
2. **Click "Import" → "Upload Files"**
3. **Chọn file đã lưu**
4. **Click "Import"**

**2.2 Cấu hình Environment**

**2.2.1 Tạo Environment mới**

1. **Click biểu tượng "Environments" (bên trái)**
2. **Click "Create Environment"**
3. **Đặt tên: API Test Environment**

**2.2.2 Thiết lập variables**

**Variable Name | Initial Value | Current Value**

**baseUrl | http://localhost:8080 | http://localhost:8080**

**apiKey | your-test-api-key | your-test-api-key**

**userId | | (để trống, sẽ set động)**

**authToken | | (để trống, sẽ set động)**

**2.2.3 Kích hoạt Environment**

1. **Chọn dropdown góc phải trên**
2. **Select "API Test Environment"**

**2.3 Test thủ công trước khi Automation**

**2.3.1 Kiểm tra từng Request**

**Request 1: Get All Users**

**// Pre-request Script**

**console.log('Testing Get All Users endpoint');**

**// Tests**

**pm.test('Status code is 200', function () {**

**pm.response.to.have.status(200);**

**});**

**pm.test('Response time is less than 1000ms', function () {**

**pm.expect(pm.response.responseTime).to.be.below(1000);**

**});**

**pm.test('Response contains users array', function () {**

**const jsonData = pm.response.json();**

**pm.expect(jsonData).to.be.an('array');**

**});**

**Request 2: Create User**

**// Pre-request Script**

**const userData = {**

**"name": "Test User " + Math.floor(Math.random() \* 1000),**

**"email": "test" + Math.floor(Math.random() \* 1000) + "@example.com",**

**"age": 25**

**};**

**pm.request.body.raw = JSON.stringify(userData);**

**// Tests**

**pm.test('Status code is 201', function () {**

**pm.response.to.have.status(201);**

**});**

**pm.test('User created successfully', function () {**

**const jsonData = pm.response.json();**

**pm.expect(jsonData).to.have.property('id');**

**pm.expect(jsonData.name).to.eql(userData.name);**

**// Lưu ID để sử dụng cho các test khác**

**pm.environment.set('userId', jsonData.id);**

**});**

**Request 3: Get User by ID**

**// Pre-request Script**

**const userId = pm.environment.get('userId');**

**if (!userId) {**

**throw new Error('userId not found in environment');**

**}**

**// Tests**

**pm.test('Status code is 200', function () {**

**pm.response.to.have.status(200);**

**});**

**pm.test('User data is correct', function () {**

**const jsonData = pm.response.json();**

**pm.expect(jsonData).to.have.property('id');**

**pm.expect(jsonData.id).to.eql(parseInt(userId));**

**});**

**2.3.2 Chạy test thủ công**

1. **Click từng request trong collection**
2. **Click "Send"**
3. **Kiểm tra Response:** 
   * **Status code**
   * **Response body**
   * **Response time**
4. **Xem kết quả Tests trong tab "Test Results"**
5. **Đảm bảo tất cả test PASS trước khi chuyển sang automation**

**2.4 Test Automation**

**2.4.1 Sử dụng Collection Runner**

**Cách 1: Collection Runner UI**

1. **Right-click collection "API Test Collection"**
2. **Chọn "Run collection"**
3. **Cấu hình Runner:** 
   * **Iterations: 1 (số lần chạy)**
   * **Delay: 100ms (thời gian chờ giữa requests)**
   * **Data file: (nếu có file CSV/JSON test data)**
   * **Environment: API Test Environment**
4. **Click "Run API Test Collection"**
5. **Xem kết quả realtime**

**Cách 2: Postman CLI (Newman)**

**# Cài đặt Newman**

**npm install -g newman**

**# Export collection và environment**

**# Postman → Collection → Export → Collection v2.1**

**# Postman → Environment → Export**

**# Chạy automation test**

**newman run api-collection.json \**

**--environment api-environment.json \**

**--reporters cli,html \**

**--reporter-html-export results.html**

**2.4.2 Cấu hình Advanced Automation**

**Pre-request Scripts cho toàn Collection**

**// Collection Pre-request Script**

**pm.globals.set('timestamp', Date.now());**

**pm.globals.set('randomEmail', 'test' + Math.floor(Math.random() \* 10000) + '@example.com');**

**console.log('Starting test run at:', new Date().toISOString());**

**Test Scripts cho toàn Collection**

**// Collection Tests**

**pm.test('Response time acceptable', function () {**

**pm.expect(pm.response.responseTime).to.be.below(2000);**

**});**

**pm.test('No server errors', function () {**

**pm.expect(pm.response.code).to.be.below(500);**

**});**

**2.4.3 Data-Driven Testing**

**Tạo CSV file (test-data.csv):**

**name,email,age**

**John Doe,john@example.com,30**

**Jane Smith,jane@example.com,25**

**Bob Johnson,bob@example.com,35**

**Sử dụng data trong request:**

**// Pre-request Script**

**const testData = {**

**"name": pm.iterationData.get('name'),**

**"email": pm.iterationData.get('email'),**

**"age": parseInt(pm.iterationData.get('age'))**

**};**

**pm.request.body.raw = JSON.stringify(testData);**

**Chạy với data file:**

**newman run collection.json \**

**--environment environment.json \**

**--data test-data.csv \**

**--iteration-count 3**

**2.4.4 Continuous Integration**

**GitHub Actions workflow (.github/workflows/api-test.yml):**

**name: API Test Automation**

**on:**

**push:**

**branches: [ main ]**

**pull\_request:**

**branches: [ main ]**

**jobs:**

**api-tests:**

**runs-on: ubuntu-latest**

**steps:**

**- uses: actions/checkout@v2**

**- name: Setup Node.js**

**uses: actions/setup-node@v2**

**with:**

**node-version: '16'**

**- name: Install Newman**

**run: npm install -g newman**

**- name: Start Spring Boot App**

**run: |**

**mvn spring-boot:run &**

**sleep 30**

**- name: Run API Tests**

**run: |**

**newman run postman/collection.json \**

**--environment postman/environment.json \**

**--reporters cli,junit \**

**--reporter-junit-export results.xml**

**- name: Publish Test Results**

**uses: dorny/test-reporter@v1**

**if: always()**

**with:**

**name: API Test Results**

**path: results.xml**

**reporter: java-junit**

**Bước 3: Monitoring và Reporting**

**3.1 Xem kết quả chi tiết**

* **Console logs: Kiểm tra Pre-request và Test scripts**
* **Test Results: Pass/Fail status cho từng test**
* **Response: Body, Headers, Status code**
* **Timeline: Thời gian thực hiện từng request**

**3.2 Export kết quả**

**# Export HTML report**

**newman run collection.json --reporters html --reporter-html-export report.html**

**# Export JSON report**

**newman run collection.json --reporters json --reporter-json-export report.json**

**3.3 Tích hợp với monitoring tools**

* **Postman Monitors: Chạy collection theo schedule**
* **Jenkins: Tích hợp vào CI/CD pipeline**
* **Slack/Teams: Gửi notification kết quả test**

**Best Practices**

1. **Tổ chức Collection: Nhóm requests theo chức năng**
2. **Sử dụng Variables: Tránh hardcode URLs và data**
3. **Error Handling: Kiểm tra cả success và error cases**
4. **Data Cleanup: Dọn dẹp test data sau khi chạy**
5. **Documentation: Thêm mô tả cho requests và tests**
6. **Version Control: Lưu collection và environment trong Git**