Automated API Testing with Rest Assured

4.2 Writing Automated Tests with Rest Assured

Learning objectives

- Send automated GET, POST, PATCH/PUT, DELETE requests
- Extract and inspect JSON responses
- Validate status codes and response bodies
- Write structured API tests using Rest Assured + JUnit
- Understand how automated API tests fit into backend workflows
- Apply these skills to test both Person and Task APIs

What You Will Learn

- How to structure a Rest Assured test
- How to send different HTTP requests
- How to assert response status codes
- How to validate JSON fields
- How to extract values for follow-up calls
- Preparing for advanced chaining and auth (later modules)

Testing GET Requests

- When testing GET endpoints, you typically validate:
  - HTTP status codes (200, 404, etc.)
  - Response structure (object or array)
  - Number of returned items
  - Specific field values
  - Presence or absence of fields
  - Data types and formats

Testing POST Requests

- The correct 201 Created status
- The returned object contains expected fields
- New resource has an auto-generated ID
- Input validation rules
- Proper JSON format in request bodies
- Location header (if provided)

Testing PUT/PATCH Requests

- Status codes (200 or 204)
- Modified fields are updated correctly
- Unchanged fields remain the same
- Behavior when updating a non-existing resource
- Business rules for updates
- JSON structure after update

Testing DELETE Requests

- Correct status code (204 or 200)
- Resource is no longer accessible afterwards
- Repeated deletes return 404
- Related data or constraints behave correctly
- No unintended side effects
- Cleanup operations for test isolation

Validating JSON Responses

- Understanding JSON objects and arrays
- Checking primitive values (strings, numbers, booleans)
- Validating nested fields
- Checking array sizes
- Matching values using conditions
- Ensuring correct data types

What Makes a Good API Test?

- Reliable, repeatable, and isolated
- Independent from other test methods
- Validating both success and failure scenarios
- Easy to understand and maintain
- Focused on behavior, not implementation
- Representative of real client interactions

Rest Assured Test Structure

- A typical test consists of:
  - Given: setup (headers, body, params)
  - When: sending the request
  - Then: validating the result
  - Uses JUnit for test execution
  - Uses JSON path for inspecting responses
  - Clean readable structure

Demo Introduction (Person API)

- In this demo, we will:
  - Update the controller to use ResponseBody for status codes
  - Write a set of Rest Assured tests for the Person API
  - Test GET, POST and DELETE endpoints
  - Validate JSON bodies and status codes
  - Show how automated tests replace manual Postman requests

Demo Goals

- The demo will help you understand:
  - How to structure Rest Assured test classes
  - How to test resource creation
  - How to validate lists and arrays
  - How to check nested or dynamic JSON
  - How to test multiple endpoints in sequence

Lab: Testing the Task API

- In this lab, you will:
  - Write automated Rest Assured tests for the Task API
  - Cover GET and POST
  - Validate JSON responses and field values
  - Build confidence writing real automated tests

Lab Instructions

- 1. Run the Task API from your previous labs, and first add the Rest Assured Dependency in pom.xml
- 2. Create a test class named TaskApiRestAssuredTest
- 3. Configure Rest Assured base URI and port
- 4. Write a test for:
  - GET /tasks (validate array size > 0)
- 5. Write a test for:
  - POST /tasks
  - Validate 201 Created
  - Validate returned id and correct fields

Lab Tips

- Keep test data simple and predictable
- Use multi-line strings for JSON bodies
- Validate both success and failure behaviors
- Keep tests readable and structured
- If you get unexpected failures, check the logs

Key Takeaways

- Rest Assured simplifies API testing dramatically
- You can automate GET, POST, PATCH/PUT, DELETE easily
- JSON validation is powerful with JSON path
- Tests should be isolated, repeatable, and meaningful
- Automated tests replace manual Postman calls