Pre-requisite - Java Knowledge

Stage 1 - API testing and api automation using testng/junit

Stage 2 - Framework - Cucumber

Stage 3 - OpenAPI Generator and advanced techniques (will be covered between 1 & 2)

1. API - Application Programming Interface

A mechanism to enable two software components to communicate with each other.

Easily understand it - architecture of server and client

Server - database that holds the stock details and its price

Client - the mobile app

1. Ways of API
   1. SOAP
      1. Less flexible than that was widely used in the past. Client and Server exchange messages using XML
   2. REST API - Representation state transfer
      1. The clients send the request to the server as data.
      2. The server uses the client's input to start the internal function and return the data
      3. Set of functions/methods → GET, POST, PUT, PATCH, DELETE, etc
      4. Statelessness. Server does not store client data between requests.
   3. Wep API/Web Services (persistent connection)
      1. Application processing the interface between web server and web browser
      2. All web service are API but not all API are webservice
   4. GraphQL
      1. A query language that helps developers with multiple query abilities.
2. REST API
3. Methods - Select/Update/Delete/Create - (GET, POST, PUT, DELETE, PATCH, HEAD, OPTION)
4. URL - BASEURL/RESOURCE
5. Request headers & body - depends on the request
6. Response
   1. Response code (1xx - informational, 2xx - success, 3xx - redirectional, 4xx - client error , 5xx-server error)
      1. 200 - success
      2. 201 - created
      3. 404 - Not found
      4. 401 - Auth error
      5. 403 - Forbidden auth error
   2. Response content

4. HTTP Methods

* Get (Path parameter and query parameters)
* Post - create
* Put/Patch - update
* Delete - create

1. Petstore API - Find Pet By Id – Path parameter
   1. Base URL - [https://petstore.swagger.io/v2](https://petstore.swagger.io/v2/swagger.json)
   2. Resource - /pet/10 (URI)
   3. Endpoint → base url+resource
2. Petstore API - Find Pet By Id – Query parameter

API Automation - Simple maven project

Installation

1. Java installation
   1. JDK 17 and above
2. IDE
   1. IntellJ
   2. Eclipse

UpperCamelCase - MyProject

lowerCameCase - myProject

Naming convention

Project - UpperCamelCase

Package - lowercase

Class - UpperCamelCase

Methods & variable - lowerCameCase

Steps to automate the rest api

Maven - Build management tools

1. pom.xml - Project Object Model
   1. Easily run using CLI - maven phase/goal/life cycle
   2. Goals - to generate jars, deploy it in pipeline
   3. Dependencies (jar) and their dependent jar will be configured automatically
   4. Java compiler - can also be easily configured here
2. Create a Maven project
   1. Group ID - com.expleo
   2. Artifact ID - Project name
3. Add dependencies
   1. Rest assured - <https://mvnrepository.com/artifact/io.rest-assured/rest-assured/5.5.6>
   2. Testng - <https://mvnrepository.com/artifact/org.testng/testng/7.11.0>
   3. Jackson-databind - <https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.20.0>
4. Create a test package and test class
5. JsonNode vs JsonPath vs Pojo class
   1. Jsonnode - Json hierarchy then it can used but not typed cast properly
   2. Jsonpath - json expression - faster but not typed cast properly
   3. Pojo class - Used widely for automation & properly type cast
6. Pojo class
   1. Class/Model - It is a blueprint or type, or template from which object are created
   2. Object
      1. Object is an instance of a class
      2. Every Object has its own state (non-static variable) and behavior (non-static method)
7. Why swagger/OpenAPI
   1. APIs are like a contract between clients and server
   2. swagger/OpenAPI - blueprint for that contract - it gives the information about endpoints, input required, output comes.
   3. Automation tester → use the blueprint - generate models, validate response, and design the automation framework
8. Open API specification - 3 sections

Can be prepared in JSON or YAML

* 1. Meta information
  2. Path items
     1. Parameters
     2. Request bodies
     3. Response
  3. Reusable components (if it is reusable then below items can be mapped here)
     1. Schemas (data models)
     2. Parameters
     3. Response

1. OpenAPI specification & its validation
   1. Openapi & info
   2. Servers
   3. Paths & operations
      1. Parameters
      2. Responses
   4. Reusable components
2. Every @Test should have minimum one assertion

Day 3

1. Actual Order used widely in the market (from testing perspective)
   1. Get the swagger document
   2. Build the OpenAPI Specification (yaml)
   3. Use OpenAPI Generator to generate the model for automation testing
   4. Build your maven project with all @Test methods for testing the API request using the model generated
2. OpenAPI Generator
   1. Open source - automatically generates the client libraries, server stubs, api documentation and models from an openapi yaml/json
   2. 50+ languages
   3. Integrates with maven, gradle or CLI

Key benefits:

1. Eliminates manual model and client side coding
2. Custom template
3. Generate the model using OpenAPI generator - CLI

Jar available → <https://mvnrepository.com/artifact/org.openapitools/openapi-generator-cli/7.16.0>

java -jar openapi-generator-cli-7.16.0.jar -i petstore.yaml -g java -o ./generated-model

* 1. -i → provide the openapi.yaml (openapi specification)
  2. - g → java
  3. -o → output directory
  4. - model-package → where the pojo (model) class should be kept
  5. –global-property- models → generates only the model
  6. modelDocs=false
  7. modelTests=false

1. Just use the model created to run with the existing project
   1. src/main/java/org/openapitools/client/model

Day 4

1. Git →

Git is a [free and open source](https://git-scm.com/about/free-and-open-source) distributed version control system

Architecture

Project (in local machine) → local repo (local machine)--> remote repo (github, aws code commit, bitbucket, gitlab)

Git concepts →

modified - staged - commit

Command to update code to github remote url

1. git init → initialize local repo
2. git add . → stage the files required to be committed
3. git commit -m “message” → update the local repo

To update remote

1. git remote add origin <https://github.com/balaji-githubstore/java-api-automation-expleo-oct-2025.git> - register the remote url with name origin
2. git push -u origin master
3. Git API
   1. List repositories for the authenticated user
   2. Create/Update/Delete
4. Verify List repo for auth user is working in postman/basic rest assured test
5. Create OpenAPI Spec for list repo for auth user - /user/repos
6. Required fields:
   1. Defined required - listed field are mandatory
   2. Not defined required - optional → if json object {} → still it gets passed
   3. Required inside the property → might throw error in latest version of open api - because it is considered as invalid
   4. Required (for path/query/header) → required: true → can be defined directly
7. OpenAPI Spec with various security

BDD Framework - Behaviour Driven Development

1. Actual BDD - Understanding the requirement
2. Modified form the BDD - ATDD - Acceptance test driven development - Writing the acceptance testing

BDD Framework

1. Python - Jbheave
2. C# - ReqnRoll
3. Java & javascript – Cucumber

Cucumber Architecture

Feature (.feature) → Step definition (.java)

Steps to create a BDD framework

1. Create a Maven project
   1. Add groupid and artifact id
2. Add dependencies
3. Feature file

Important concepts – (To improve Reusability, readability, maintenance)

1. Step parameterization
2. Scenario outline
3. Background
4. Datatable

Category obj1=new Category();

Category obj2=new Category();

obj1.setId(3000);

System.*out*.println(obj1.getId());

System.*out*.println(obj1.getName());

System.*out*.println(obj2.getId());

System.*out*.println(obj2.getName());

Day 3

1. Add pet
2. FindPetByStatus - binding to pojo model
3. OpenAPI Generator → model generator (POJO class)
4. We model generated using openAPI on the existing test method
5. Git API
6. Git or Setup for Cucumber framework

Day 4

1. Git API - list repo for users, create, update, delete
2. Add automation for the list repo for the user
3. Create an openAPI spec and run the automation with validation
4. Create a Model using the open api spec (YAML) —> do it as a task after the session
5. Cucumber framework - BDD Concepts

Day 5

1. Will concepts of BDD
2. Design of the actual framework
3. Mock API

Important notes:

1. Security types and usage in OpenAPI Spec

openapi: 3.0.3

info:

title: GitHub API (User Repos)

version: 1.0.0

servers:

- url: https://api.github.com

paths:

/user/bearer:

get:

security:

- tokenAuth: []

/user/basic:

get:

security:

- basicAuth: []

/user/api-key:

security:

- apiKeyAuth: [ ]

/user/oauth:

security:

- oauth2Auth:

- repo

- user

components:

securitySchemes:

tokenAuth:

type: http

scheme: bearer

bearerFormat: "token"

basicAuth:

type: http

scheme: basic

apiKeyAuth:

type: apiKey

in: header

name: X-API-Key

oauth2Auth:

type: oauth2

flows:

authorizationCode:

authorizationUrl: https://github.com/login/oauth/authorize

tokenUrl: https://github.com/login/oauth/access\_token

scopes:

repo: Access to repositories

user: Access to user profile

Assignments:

Task 1:

1. Use OpenAPI Spec git created in session and generate model using open api generator

Session Code - <https://github.com/balaji-githubstore/java-api-automation-expleo-oct-2025.git>

Github API for practice:

<https://docs.github.com/en/rest/repos?apiVersion=2022-11-28>

1. [List repositories for a user](https://docs.github.com/en/rest/repos/repos#list-repositories-for-a-user)
2. [List repositories for the authenticated user](https://docs.github.com/en/rest/repos/repos#list-repositories-for-the-authenticated-user)
3. [Create a repository for the authenticated user](https://docs.github.com/en/rest/repos/repos#create-a-repository-for-the-authenticated-user)
4. [Update a repository](https://docs.github.com/en/rest/repos/repos#update-a-repository)
5. [Delete a repository](https://docs.github.com/en/rest/repos/repos#delete-a-repository)