API Automation Framework – Detailed Execution Documentation

This document provides a class-by-class and file-by-file explanation of the API automation framework built using Java, Rest Assured, and TestNG. It explains how each component participates in execution and how control flows from one file to another during runtime.

1. Entry Point of Execution

testng.xml

testng.xml is the entry point for the framework execution. When Maven runs `mvn test`, the Surefire plugin reads this file and hands over control to TestNG. Responsibilities: • Defines which test classes should run • Controls execution order • Supports grouping and parallelism Execution: testng.xml → TestNG Engine → Test Classes

2. Base Layer – Framework Initialization

BaseTest.java

BaseTest is the foundation of the framework and is extended by all test classes. Key responsibilities: • Initializes Rest Assured base URI • Stores shared runtime data such as accessToken and userId Execution: • @BeforeSuite method runs once before any test method • Values initialized here are accessible to all tests

3. Configuration Layer

EnvConfig.java

EnvConfig loads environment-specific configuration from the .env file. Responsibilities: • Reads BASE_URL and endpoint values • Decouples environment data from code Execution: • Called from BaseTest during suite initialization • Also used directly by test classes if required

4. Constants Layer

Endpoints.java

Endpoints class stores all API paths as constants. Responsibilities: • Centralizes API endpoints • Avoids hardcoding URLs in tests • Supports dynamic endpoints using placeholders Execution: • Referenced by test classes during API calls • Does not execute independently

5. Utility Layer

RequestSpecFactory.java

This utility class builds reusable Rest Assured request specifications. Responsibilities: • Default request setup (headers, content type, logging) • Authenticated request setup using token Execution: • Called from test classes before API invocation • Returns RequestSpecification to the calling test

6. Payload Layer

LoginRequestBody.java

This class generates request bodies for API calls. Responsibilities: • Builds login request payload • Keeps request body logic separate from tests Execution: • Called by LoginTest during request creation

7. Test Layer – Actual Execution

LoginTest.java

This is the first test executed in the framework. Responsibilities: • Calls Login API • Extracts and stores accessToken Execution Flow: TestNG → LoginTest → RequestSpecFactory → LoginRequestBody → Endpoint → Response

UserDetailsTest.java

This test depends on LoginTest. Responsibilities: • Uses accessToken to fetch user details • Extracts and stores userId Execution Flow: LoginTest completed → UserDetailsTest → Auth RequestSpec → Endpoint → Response

UserTodosTest.java

This test depends on UserDetailsTest. Responsibilities: • Uses userId to fetch todos list Execution Flow: UserDetailsTest completed → UserTodosTest → Dynamic Endpoint → Response

8. Complete Execution Flow Summary

1. Maven Surefire triggers TestNG 2. testng.xml defines test execution 3. BaseTest initializes framework 4. LoginTest generates token 5. UserDetailsTest fetches user data 6. UserTodosTest fetches todos 7. Suite execution completes

9. Key Design Principles

• Separation of concerns • Reusability • Readability • Maintainability • CI/CD readiness

Conclusion

This document serves as a reference guide for understanding how the framework operates internally. It is intended for learning, onboarding, and long-term maintenance purposes.