**⭐ API Automation Framework - Complete Documentation**

**📈 1. Framework Overview**

**🔹 What Does This Framework Do?**

* Automates API testing using **RestAssured**.
* Manages API requests and responses through **service classes**.
* Uses **configurable assertions** to validate API responses.
* **Modular, reusable, and scalable** for future enhancements.

**📈 2. High-Level Flow of Execution**

1. **Test case (BuyerTest.java)** calls a service method (BuyerService.addBuyerRequest()).
2. **Service layer (BuyerService.java)** constructs and sends the API request.
3. **API response is captured** and passed to FrameworkAssertions.assertThat(response).
4. **Assertion layer (ResponseAssertion.java)** validates the response (status, headers, JSON fields).
5. **Test execution report** is generated, logging failures and successes.

**📈 3. Key Components & Responsibilities**

| **Component** | **Purpose** |
| --- | --- |
| BuyerTest.java | Contains API test cases. |
| BuyerService.java | Handles API requests and responses. |
| FrameworkAssertions.java | Centralized assertion factory for validating responses. |
| ResponseAssertion.java | Contains reusable response validation methods. |
| RestAssuredClient.java | Configures API client with headers, logging, and authentication. |
| ConfigFactory.java | Loads environment-specific properties (URLs, auth tokens). |
| 1. Component Purpose  2. BuyerTest.java Contains API test cases.  3. BuyerService.java Handles API requests and responses.  4. FrameworkAssertions.java Centralized assertion factory for validating responses.  5. ResponseAssertion.java Contains reusable response validation methods.  6. RestAssuredClient.java Configures API client with headers, logging, and authentication.  7. ConfigFactory.java Loads environment-specific properties (URLs, auth tokens).  8. EndPoint.java Manages API endpoint constants.  9. |

**📈 4. Detailed Execution Flow**

**📂 Step 1: The Test Case Calls an API**

**File:** BuyerTest.java  
**Description:** A test case sends an API request and validates the response.

@Test(priority = 1, description = "Create a new buyer request and validate response")

1. 1. public void verifyAddingBuyerRequest() {

2. 2. BuyerVO buyerRequestVO = generateBuyerPayload();

3. 3. Response response = BuyerService.addBuyerRequest(buyerRequestVO);

4. 4.

5. 5. // Validate API Response

6. 6. FrameworkAssertions.assertThat(response)

7. 7. .hasSameStatus(200)

8. 8. .containsField("buyerId")

9. 9. .containsFieldWithValue("companyName", buyerRequestVO.getCompanyName())

10. 10. .containsResponseText("Buyer Request Sent Successfully")

11. 11. .hasHeader("Content-Type", "application/json; charset=utf-8");

12. 12. }

13. 13.

14.

**📂 Step 2: The Service Layer Sends the API Request**

**File:** BuyerService.java  
**Description:** Handles API interactions and sends HTTP requests.

1. public static Response addBuyerRequest(BuyerVO buyerRequestVO) {

2. return RestAssuredClient

3. .withDefaultConfigurations()

4. .basePath(EndPoint.BUYER\_SERVICE\_BASE\_PATH)

5. .header("X-Uuid", "30f15e6f-0a90-43b8-8c59-c43547e665c3")

6. .contentType(ContentType.JSON)

7. .body(buyerRequestVO)

8. .accept(ContentType.JSON)

9. .expectStatusCode(StatusCode.OK.getStatusCode())

10. .request(Method.POST, EndPoint.ADD\_BUYER\_REQUEST\_PATH);

11. }

12.

**📂 Step 3: Assertions Validate the API Response**

**File:** ResponseAssertion.java  
**Description:** Provides reusable assertion methods.

1. public class ResponseAssertion {

2. private final Response response;

3.

4. public ResponseAssertion(Response response) {

5. this.response = response;

6. }

7.

8. public static ResponseAssertion assertThat(Response response) {

9. return new ResponseAssertion(response);

10. }

11.

12.

1. public ResponseAssertion hasSameStatus(int expectedStatus) {

2. Assertions.assertThat(response.getStatusCode())

3. .as("Expected status code: " + expectedStatus + ", but got: " + response.getStatusCode())

4. .isEqualTo(expectedStatus);

5. return this;

6. }

7. }

8.

**📈 5. Configuration & Setup**

**Environment Configuration**

**File:** ConfigFactory.java

1. public static EnvironmentConfig getEnvironmentConfig() {

2. return ConfigCache.getOrCreate(EnvironmentConfig.class);

3. }

4.

**API Endpoints**

**File:** EndPoint.java

1. public static final String BUYER\_SERVICE\_BASE\_PATH = "/api";

2. public static final String ADD\_BUYER\_REQUEST\_PATH = "/addBuyerRequest";

3.

**📈 6. Best Practices for Future Enhancements**

* **Use FrameworkAssertions.java** to centralize assertions for all tests.
* **Add new assertion methods in ResponseAssertion.java** instead of duplicating in tests.
* **Use ConfigFactory.java** for environment management instead of hardcoding values.
* **Use RestAssuredClient.java** for common API client configurations.
* **Write modular service methods (BuyerService.java)** to make tests reusable.

**📈 7. Summary of Execution**

**🚀 End-to-End Flow**

1. **Test case (BuyerTest.java) calls an API using BuyerService.java**.
2. **BuyerService.java sends API requests using RestAssuredClient.java**.
3. **Response is passed to FrameworkAssertions.assertThat(response)**.
4. **ResponseAssertion.java validates status codes, headers, and response fields**.
5. **Failures are logged, and the test either passes or fails**.

**🚀 Now, You Have a Complete API Automation Framework!**

This documentation ensures that you **never forget the framework flow** and can **easily modify or expand it** in the future.