Hybrid Framework explaination

Page class

insertData(String jsonData)

Purpose: This method is responsible for inputting JSON data into a text area and triggering the refresh of a table that displays that data.

Key Steps:

- **Click the Button**: It first clicks the tableDataButton,
- **Clear and Input Data**:
  - It clears any existing text in the inputTextBox .
  - The provided jsonData is then sent to the inputTextBox using the sendKeys method.
- **Refresh the Table**:
  - JavaScript to scroll the refreshButton into view, ensuring that it can be clicked even if it's not visible on the screen.

getTableData()

Purpose: This method extracts the data from the dynamic table and formats it into a two-dimensional array of strings.

Key Steps:

- **Calculate Number of Rows**: It calculates the number of rows in the table by dividing the total number of cells (tableCells.size()) by 3, assuming each row has three columns (Name, Age, Gender).
- **Initialize 2D Array**: A 2D string array tableData is created to hold the data, with dimensions based on the calculated number of rows and a fixed three columns.

The for-loop iterates over the number of rows. Within each iteration:

The method retrieves text values from the tableCells list using the calculated indices:

- tableCells.get(i * 3 + 1).getText(): Gets the Name (2nd column).
- tableCells.get(i * 3 + 2).getText(): Gets the Age (3rd column).
- tableCells.get(i * 3).getText(): Gets the Gender (1st column).

- Each retrieved value is stored in the corresponding position of the tableData array.

Return Data: Finally, the method returns the populated tableData, which can be used for further processing or assertions in tests.

Test Class

1st Loop Through Input JSON Data:

- This loop iterates through the jsonDataList, which contains JSON objects.
- For each JSON object, a new JSONObject (jsonObject) is created.
- The name, age, and gender from the input JSON object are extracted and added to the jsonObject.
- The newly created jsonObject is then added to the jsonArray.

2nd loop Verify Table Data:

This loop iterates through the original jsonDataList to verify that the data displayed in the table matches the input data.
For each row:

- It retrieves the current JSON object from jsonDataList.
- It asserts that the values in tableData match the expected values from the input JSON object:
  - **Name**: Asserts that the name in the table matches the expected name.
  - **Age**: Asserts that the age in the table matches the expected age.
  - **Gender**: Asserts that the gender in the table matches the expected gender.

If any assertion fails, an informative message is included, indicating the row number where the mismatch occurred.

Overview of the Program Structure

1. **Imports**: The necessary libraries from Rest Assured and TestNG are imported to handle HTTP requests and assertions.
2. **Base URI Setup**: The base URL for the API is defined in the @BeforeClass method.
3. **Test Methods**: Each HTTP method is implemented in its respective test method, with assertions to validate the responses.

Setup

@BeforeClass: This annotation indicates that the setup() method should run once before any of the test methods in this class.
RestAssured.baseURI: This sets the base URL for all subsequent requests, making it easier to write the endpoint paths without repeating the full URL.

GET Request Test

@Test: Marks this method as a test method for TestNG.
given().when().get("/posts/1"): Constructs the request to fetch a post with ID 1.
then(): Indicates that assertions follow.
.statusCode(200): Asserts that the HTTP response status is 200 OK.
.body("title", equalTo(...)): Asserts that the response body contains the expected title.
.log().status() and **.log().body()**: Logs the response status and body for debugging.

POST Request Test

Creating POST Data: This HashMap is used to structure the JSON data sent in the POST request.
contentType(ContentType.JSON): Specifies that the request body is in JSON format.
Assertions: Checks that the response has the expected status code 201 Created and that the returned values match the input data.

PUT Request Test

PUT Data Structure: Similar to POST, but includes the id to specify which resource to update.
Assertions: Ensures the updated values match the expected results.

PATCH Request Test

PATCH Data Structure: Only the title is sent for updating, partial updates.
Assertions: Verifies that the title is updated correctly.

DELETE Request Test

DELETE Request: Simply performs a delete operation on the specified post.
Assertions: Confirms that the status code indicates success and that the response body is empty, indicating the resource was successfully deleted.

Summary

- This program provides a complete suite of tests for the RESTful API endpoints.
- Each test method is designed to validate the functionality of a specific HTTP method, ensuring that the API behaves as expected for various types of requests.
- The assertions help verify that the responses match the expected outcomes, log() provides visibility into the interactions for easier debugging.