

### Assignment

#### Title: Chaining of Requests (Create, Update, and Delete) with Assertions in JIRA

#### Objective

The objective of this assignment is to create, update, and delete an issue in JIRA by chaining these requests using Rest Assured in Eclipse. You will also assert the status code, status line, and response time for each request.

#### Prerequisites

* Obtain your JIRA API key .

**Example Credentials:**

* URL: https://your-jira-instance.atlassian.net
* Username: your-email@example.com
* API Token: your-api-token

### Structure of the Project

Create a package jirachaining

Create separate classes for:

1. **Base Class**
2. **CreateIssue**
3. **GetIssue**
4. **UpdateIssue**
5. **DeleteIssue**

### Implementation Outline

#### 1. Base Class

**Purpose:** Set up the base URL and common configurations. Store the issueId.

**Pseudo Code:**

Class BaseTest {

// Define issueID as a global vaiable

public static String issueId;

Create a method –setUp(){

// Base URL for the JIRA instance

baseUrl = https://your-jira-instance.atlassian.net/rest/api/2/issue/ ";

// Authentication credentials ***preemptive().basic(username,******apiToken)***

}

}

#### 2. CreateIssue Class

**Purpose:** Handle the creation of an issue and store the issueId in the Base class.

**Pseudo Code:**

// Create Issue

Class CreateIssue extends BaseTest {

// Method to create an issue

public void createIssue() {

// Set up request with necessary headers and body

**{**

**"fields": {**

**"project": {**

**"key": "MNP"**

**},**

**"summary": "create issue in RA project",**

**"description": "Creating of an issue using project keys and issue type names using the REST API",**

**"issuetype": {**

**"name": "Story"**

**}**

**}**

**}**

// Send POST request to create an issue

.body(requestBody)

.post()

.then()

.assertThat()

.statusCode(201)

.statusLine("HTTP/1.1 201 Created")

.time(lessThan(2000L)) // Asserting response time less than 2000ms

.extract()

.response();

// Extract issueId from response and store it in BaseTest.issueId

BaseTest.issueId = response.jsonPath().getString("id");

Assert.assertNotNull(BaseTest.issueId, "issueId should not be null");

}

}

#### 3. GetIssue Class

**Purpose:** Handle retrieval of the issue details using the issueId from the Base class.

**Pseudo Code:**

// Get Issue

Class GetIssue extends BaseTest {

// Method to get an issue

public void getIssue() {

// Send GET request to retrieve the issue details

.get(issueId)

.then()

.assertThat()

.statusCode(200)

.statusLine("HTTP/1.1 200 OK")

.time(lessThan(2000L)); // Asserting response time less than 2000ms

}

}

#### 4. UpdateIssue Class

**Purpose:** Handle updating of the issue using the issueId from the Base class.

**Pseudo Code:**

// Update Issue

Class UpdateIssue extends BaseTest {

// Method to update an issue

public void updateIssue() {

// Set up request with necessary headers and updated body

String requestBody = **"{ \"fields\": { \"summary\": \"Updated Test Issue\" } }";**

// Send PUT request to update the issue

.body(requestBody)

.put(issueId)

.then()

.assertThat()

.statusCode(204)

.statusLine("HTTP/1.1 204 No Content")

.time(lessThan(2000L)); // Asserting response time less than 2000ms

}

}

#### 5. DeleteIssue Class

**Purpose:** Handle deletion of the issue using the issueId from the Base class.

**Pseudo Code:**

// Delete Issue

Class DeleteIssue extends BaseTest {

// Method to delete an issue

public void deleteIssue() {

// Send DELETE request to delete the issue

.delete(issueId)

.then()

.assertThat()

.statusCode(204)

.statusLine("HTTP/1.1 204 No Content")

.time(lessThan(2000L)); // Asserting response time less than 2000ms

}

}

### Implementing Chaining of Requests

**Pseudo Code:**

* Hint: Add @BeforeMethod in BaseClass and Depends on method for Update and Delete @Test(dependsOnMethod=”chaining.CreateIncident.create”)
* Run From Delete Instance

### Expected Outcome

Upon completion, you should be able to:

1. Understand how to use Rest Assured to interact with JIRA APIs.
2. Create, retrieve, update, and delete an issue in JIRA via API.
3. Chain these requests using separate classes for each operation with the issueId stored in the Base class.
4. Assert the status code, status line, and response time for each request to ensure successful operation.