Stage 1 – API Testing using postman

Stage 2 – API Automation (Robot Framework + Request Library)

1. API – Application Programming interface
   1. A Mechanism that enables two software components to communicate with each other.
2. Easily understand the API architecture with client and server

* Server – the database that holds the stock details and its price
* Client – the mobile app

1. Ways of API works
   1. SOAP – Simple Object Access Protocol
      1. Flexible widely in the past. Client and server exchange the messages using XML
   2. REST API – Representation State Transfer
      1. Client send the request to the server as data
      2. Server uses client input to start internal function and return the data
      3. REST defines set of functions such as GET, POST, PUT, DELETE etc
      4. Client and server exchanges the data using http.
      5. Main feature of REST Is statelessness. Server do not store client data between the requests.
   3. Web API/ Web Services
      1. An application processing the interface between web server and web browser.
      2. All Web Services are API but not all API are web service.
   4. GraphQL
      1. GraphQL is a query language that was developed specifically for APIs
      2. It is designed to give a specific data requested by the client and not more.
      3. GraphQL gives front end developers the ability to query multiple databases, microservices and API with the single graphQL endpoints.
2. Installation
   1. Install SOAP UI (Not required now, just listen)
   2. Install Postman (RestAPI – development, testing, management)
3. SOAP API Testing – WSDL (Web Service Description Language)
   1. Create SOAP project
   2. Provide the wsdl details
   3. It shows all the available methods. Try running the methods by providing the required data in the xml (request body)
   4. Validate the response.
4. Rest API – Web Archive Description Language (HTTP)
   1. Methods – Select/Update/Delete/Create – (GET, POST, PUT, DELETE, PATCH, HEAD, OPTION)
   2. URL – Base URL/resource
   3. Request Header & Body – depends on the request
   4. Response (Server Status Code)
      1. 200 🡪 Success
      2. 201 🡪 Created
      3. 404 🡪 resource not found
      4. 401 🡪 Auth error
      5. 403 🡪 Forbidden auth error

HTTP Methods

Get 🡪 Select (Path parameter & query string)

Post 🡪 Create

Delete 🡪 Delete

Put/Patch 🡪 update

Testcase for Login UI in facebook

1. Valid username and valid password
2. Valid username and invalid password
3. Invalid username and valid password
4. Valid mobile number and password (all combinations)
5. Empty username and password (all combinations)

API – assume checking product in amazon using product id

1. Without authorization
2. With restricted authorization
3. With full authorization
4. ….
5. Structure of postman to create request

Workspace

Collection 🡪 folder

Create All Request 🡪 HTTP Request

Sub collection 🡪 folder

Create any request🡪 HTTP Request

1. Work on Petstore API – Get pet by id (Get method with Path parameter)

Base URL - [https://petstore.swagger.io/v2](https://petstore.swagger.io/v2/swagger.json)

Resource - /pet/{petid}

1. Find pet by status (Get method with query parameter)
2. Add pet to store (Post)
3. Update a pet (Put)
4. Path parameter using variables
5. Can create variable at different scopes
   1. Global
   2. Environment
   3. Collection
   4. Data
   5. Local
6. Calendar API
7. Calendar list - <https://developers.google.com/calendar/api/v3/reference/calendarList/list>
8. Oauth 2.0

* Auth url
* Access token url
* Client id
* Client\_Secret
* Scope
* Redirect url – app owner/developer

Level 1: To get authorization code

Post 🡪 auth url?client\_id=&scope=&redirect\_url=

Level 2: Use authorization code and get **access token**

Post 🡪 access token url?auth\_code=&client\_secret=

1. Github API
2. Newman
3. Data Driven using postman

Automation:

1. Robot Framework – https://robotframework.org/

* Basic python is pre-requisite
* Keyword driven framework extensively designed using python.

1. Installation
   1. Install Python
   2. Install Pycharm
      1. Install plugin – robot framework language server
   3. Install robot framework

**pip install robotframework**

1. Robot framework Learning path
   1. Standard Libraries
      1. Builtin
      2. String
      3. Datetime
      4. Collections
      5. Operating System
   2. External libraries
      1. Selenium Library – Web Automation
      2. Appium Library – Android, ios, windows
      3. Request Library – API Automation
2. Create suite folder and suite file (.robot)
3. Robot Framework Section
   1. Settings
   2. Test Cases
   3. Keywords
   4. Variables
      1. Scalar ($)
      2. List (@)
      3. Dictionary (&}

Selenium – Web Automation

1. Install Selenium library

**pip install --upgrade robotframework-seleniumlibrary**

1. Keyword doc - <https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>
2. Architecture of Selenium WebDriver

Source code (Robot framework+selenium library) 🡪 Browser

1. Launch browser
2. Click, type, select
3. To inspect 🡪 tagname, attribute, text or not
4. Basic locators
   1. Id
   2. Name
   3. Classname
   4. Tagname
   5. Linktext
   6. Partial linktext

When there are duplicate locators the find element will pick the first element.

1. Advance locators
   1. XPath
   2. CSS
2. Click 🡪 Element should be present and visible
   1. ElementClickIntercepted -🡪 some popup hides the target element
   2. ElementNotInteractable 🡪 element is present not visible.
3. To inspect🡪 ctrl+shift+c or f12
4. Page load – no specific time limit – wait for page load to complete
5. findElement 🡪 takes only 0.5s/500ms to check the presence of element.
6. Synchronization
   1. Unconditional wait (not recommended)
      1. Sleep 8s
   2. Conditional wait (from selenium lib)
      1. Implicit wait
         1. Default implicit wait is 0s
         2. Applicable for all find\_element and find\_elements methods
         3. Example: Implicit wait – 30s
            1. If element is not present, it will check for 30s and then throw exception
            2. If element is present, it will do the operation immediately
            3. Polling time – 500ms (how frequently it checks for element)
      2. Explicit wait
7. Dropdown
   1. With Select tag
      1. Select From List By Index
      2. Select From List By Label
      3. Select From List By Value
   2. Without Select tag
8. Multiple tabs/windows, alert, frame
9. Mutliple tabs
   1. Using title or url
   2. Using NEW and MAIN locator
10. Close window vs Close Browser
    1. Close window- close the current tab/session (close())
    2. Close Browser – close the current browser/all session and also kill the process associated to it (quit())
11. Frame – embedding html into another html
    1. Even though the locator is correct we get element not found error.
    2. Check for tagname – iframe or frame
    3. Switch to frame
12. Setup and teardown
    1. Can be applied at setting level and also test case level.
13. Alert
14. Mouse/Keyboard
15. Upload the file/image
16. Javascript
    1. Click on hidden elements
    2. Type on readonly text box

Click and type using javascript

document.querySelector('#bill-date-long').click()

API Automation – Request Library

1. Install request library

pip install robotframework-requests

1. Keyword doc - <http://marketsquare.github.io/robotframework-requests/doc/RequestsLibrary.html>
2. Robot framework concept

Suite Setup - run only once before triggering the first test case   
Suite Teardown - run only once after the last test case  
Test Setup - runs before each test case  
Test Teardown - runs after each test case

1. API Automation – petstore and calendar api
2. Github API
   1. Rest API Doc - <https://docs.github.com/en/rest/repos?apiVersion=2022-11-28>
   2. Authentication
      1. Basic auth (Username & password)
      2. Bearer token
      3. Oauth 2.0
         1. <https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps>
         2. Scope - <https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/scopes-for-oauth-apps>
         3. Redirect url
            1. Postman desktop - <https://oauth.pstmn.io/v1/browser-callback>
            2. Postman browser - <https://oauth.pstmn.io/v1/browser-callback>
   3. API endpoints
      1. List public repos
      2. Used different auth
      3. List user repo
      4. Created repo
      5. Update repo – (change visibility to public)
      6. Delete repo
3. Postman monitor – scheduler
4. Newman
   1. Install nodejs

<https://nodejs.org/en>

* 1. Install newman using command line

npm install -g newman

* 1. Run collections

C:\Mine\Company\Swift Support 2 Jun 2024>newman run "PetStoreAPI.postman\_collection.json"

* 1. Run collections with environment

C:\Mine\Company\Swift Support 2 Jun 2024>newman run "PetStoreAPI.postman\_collection.json" -e "QA.postman\_environment.json"

Customize framework for API Automation

1. Create a project for each application
2. Create a test suites folder and suite file (robot)
3. Create a test case and then connect with test template.
4. Test Template – helps to run the test case with multiple set of test data
   1. Create a keyword with proper arguments.
   2. Register the template at setting level or test case level
5. Test Template with csv and excel file
   1. Use DataDriver Library
      1. Install the library

pip3 install --upgrade robotframework-datadriver

pip install --upgrade robotframework-datadriver[XLS]

* + 1. Now import the library and provide csv or excel details

1. Command line control of robot framework
   1. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#command-line-options-for-test-execution>

Assignments

1. Complete at least 4 api calls - <https://reqres.in/>
2. People contact api - <https://developers.google.com/people/api/rest/v1/contactGroups/list>

Reference

|  |  |  |
| --- | --- | --- |
| **Strategy** | **Match based on** | **Example** |
| id | Element id. | id:example |
| name | name attribute. | name:example |
| identifier | Either id or name. | identifier:example |
| class | Element class. | class:example |
| tag | Tag name. | tag:div |
| xpath | XPath expression. | xpath://div[@id="example"] |
| css | CSS selector. | css:div#example |
| dom | DOM expression. | dom:document.images[5] |
| link | Exact text a link has. | link:The example |
| partial link | Partial link text. | partial link:he ex |
| sizzle | Sizzle selector deprecated. | sizzle:div.example |
| data | Element data-\* attribute | data:id:my\_id |
| jquery | jQuery expression. | jquery:div.example |
| default | Keyword specific default behavior. | default:example |

When UI text is uppercase and html text is lower case

Linktext & partial linktext 🡪 use the font present on UI

Xpath – font based on html