41. understanding the importance of API integration calls for Web tests

how API testing can help to make your web automation test easy.

So that means to reduce the execution time off your web automation or to make it more stable.

How you can use the help of APIs and integrate with your web automation test to build robust test cases.

all the new technology apps, whatever you are seeing are fully driven backend by “API’s” only.

So that means an API will give you a response and frontend guy will parse the json response  and render the data what they got from backend in to the frontend on the web pages.

So now if I want to log in, what exactly happens in the backend?

So in the backend, there will be one API call made.

So that details what you sent here from the frontend email and password that will be sent as a request and it will be sent to the API server.

So API server, that is one login call which is made and it will give us the response back with

the session token. So based upon that token, you will be authenticated in this web application.

let's try to hit this API from our playwright because playwright supports API testing also.

Using our playwright, we will do this API call.

We will grab the response, take the token, and we will inject that into our session storage that also playwright can do.

And then we will load this URL so that directly you will be logged in and you can skip that logging part.

So this whole web application is driving with the session cookie. That's the common scenario in most of the websites.So whenever you log in, that log in, it will create a one session cookie.

42. Playwright request method to make API calls and grab response – Example

New file created(WebAPIpart1.spec.js) - copy the entire clientApp.spec.js code in to webAPIPart1.spec.js file and remove the login code.

Dot beforeAll So that means before executing actual tests, let's say you have three tests in this file. So before everything, the block of code, what you write here will get executed.

Dot beforeeach -  let's say you have three test cases in this file.

So now when you execute before each, that means execute this code before each and every test.

In playwright To make any API testing. There is a one library called request.

Request - Exposes API that can be used for web api testing.

Const{test,expect,request} = require(‘@playwright/test’);

So this is what we need because right now we are going to perform web API only.

So call this. So using this request object, you can be able to call APIs and work with API responses.

So for API, just like have you started browser dot new context to open a context session. In API Also using this request dot context.

if you want to send default information, then you can send inside this context like loading the base URL or something like that or you can give headers, HTTP credentials, proxy things.

Let's say your API will only be accessible on giving proxy and timeout. Anything you can pass in this context.

From this context we are creating a new page and on that page we are acting up on right now from this context

Payload we are kept in one variable(loginpayload)

when you are storing it as a JavaScript object, the property need not have the quotes(“”) only value can have the quotes

 after giving this post URL any data, you have to grab them CURRLY BRACES

43. Parsing API response & passing token to browser local storage with playwright

To insert the token, we have to execute a JavaScript.

how playwright can execute any JavaScript expressions so for that there is a method called which comes with a page dot “addInitscript”

you can insert a lot of JavaScript inside of this. So that way it will work.

Now you have to write a JavaScript which can store that cookie right now.

if you want to insert a value in the local storage, first, you have to give the argument and that

argument you have to write it in function.

So basically, we are inserting a script function, which takes this an argument.

So inside the function. Just write a code which can insert the item in the local storage.

So for that there is one simple JavaScript window, browser window open the local storage and

Set the item

So in your browser window, open the storage and set the item what you want to set.

basically, this is a function which is taking this argument and setting it in the token

that function value that you have to send as a second argument comma and send your value(token).

This addinitscript take two arguments.

First one is a function.

Second one is the parameters.

So all parameters required for this function will be given as the second argument.

So it will simply take this parameters injected to this function and token will fall here and from here it will just come into this place.

now after setting this token and when you hit this URL, it will be directly taking you to dashboard page it will bypass your loginscreen.

const {test,expect,request} = require('@playwright/test');

const loginpayload = {userEmail: "anshika@gmail.com", userPassword: "Iamking@000"}

let token; // placed token outside it’s accessible for all the tests

test.beforeAll(  async() =>

{

   const apiContext = await request.newContext();

   const loginResponse = await  apiContext.post("https://rahulshettyacademy.com/api/ecom/auth/login",

   {

      data : loginpayload

   }  )

expect(loginResponse.ok()).toBeTruthy(); // assertion OK status code

const loginResponseJson = await loginResponse.json();  // return the JSON representation of response body

token = loginResponseJson.token;

console.log(token);

});

test('@Web Client App Login', async ({page})=>    {

    page.addInitScript(value => {

      window.localStorage.setItem('token',value);  // key : value pair

    }, token);

 //await page.goto("https://rahulshettyacademy.com/client");

 //await page.locator("#userEmail").fill(email);

 //await page.locator("#userPassword").fill("Iamking@000");

 //await page.locator("[value='Login']").click();

 //await page.waitForLoadState("networkidle");   // loaded all the network but due to some flaky that why we use wait for method

 const email = "anshika@gmail.com";

 const productName = 'ZARA COAT 3';

 const products = page.locator(".card-body");

 await page.goto("https://rahulshettyacademy.com/client");

 await page.locator(".card-body b").first().waitFor();

 const graballTitles = await page.locator(".card-body b").allTextContents();

// console.log(graballTitles);

 const count = await products.count();

 //ZARA COAT 3

 for(let i=0; i<count ; ++i)

 {

   if(await products.nth(i).locator("b").textContent() === productName )

   {

    await products.nth(i).locator("text= Add To Cart").click();

    break;

}

 }

 //await page.pause();

 await page.locator("[routerlink\*='cart']").click();

 await page.locator("div li").first().waitFor();

 const bool = await page.locator("h3:has-text('ZARA COAT 3')").isVisible();

 expect(bool).toBeTruthy();

 await page.locator("text=Checkout").click();

 await page.locator("[placeholder\*='Country']").pressSequentially("ind",{delay:100});

 const dropdown = page.locator(".ta-results");

 await dropdown.waitFor();

 const optionsCount = await dropdown.locator("button").count();

 for(let i=0;i<optionsCount;++i)

 {

   const text = await dropdown.locator("button").nth(i).textContent();

    if(text === " India")

    {

        await dropdown.locator("button").nth(i).click();

        break;

    }

 }

 expect(page.locator(".user\_\_name [type='text']").first()).toHaveText(email);

    await page.locator(".action\_\_submit").click();

    await expect(page.locator(".hero-primary")).toHaveText(" Thankyou for the order. ");

    const orderId = await page.locator(".em-spacer-1 .ng-star-inserted").textContent();

    console.log(orderId);

    await page.locator("button[routerlink\*='myorders']").click();

   await page.locator("tbody").waitFor();

   const rows = await page.locator("tbody tr");

   for (let i = 0; i < await rows.count(); ++i) {

      const rowOrderId = await rows.nth(i).locator("th").textContent();

      if (orderId.includes(rowOrderId)) {

         await rows.nth(i).locator("button").first().click();

         break;

      }

   }

   const orderIdDetails = await page.locator(".col-text").textContent();

   expect(orderId.includes(orderIdDetails)).toBeTruthy();

});

44. Place order API to create order and bypass the flow in UI with mix of WEB/API

if you see the login application in your side in session storage, then this initialization

script is there right Then set it to the session storage, sessionstorage dot set item.

You need to figure it out by asking your developer like how they are storing our session information to log in.

So here, let's say your goal is to check whether the orders are coming are displaying in the order history page. And are you be able to view the order details?

// Test case - Verify if order creator is showing up in the history page.

Check if there is any API call to create order. If there is an API, then you create an order with this API Call that within a second you will get orderID so that orderID you can look directly in history page.

So you can directly jump to our orders page and look at for your order ID.

when you create an order, if internet is slow, our browser is not rendering properly or if there

is any browser issue. Your test will fail for many reasons.

You all know right that your automation is not that stable, so it is better to avoid as much as you can. And you can use help of API to create all this test data.

What is ready for your test and you can directly focus on your exact test case in UI and remaining all preconditions set up, post condition data deletion, everything. You can takecare by this API calls.

45. End to end validation with mix of API & Web concepts - Reduce test time

In test dot before all after token

In post method 1 st argument url, 2nd argument whatever u want to send put them in currlly braces

Const orderResponse = Apicontext.post(“createorderURL”,

{

Data : orderpayLoad,

headers:{

‘Authorization’ : token,

‘content-Type’ : ‘application.json’

},

})

Const orderresponsejson = Orderresponse.json(); - response is written for u as json

orderID = Orderresponsejson.orders[0];

when you are creating this payload as a JavaScript object, then you can remove this quotes(“ “) for all these keys. At runtime, the JavaScript object will be converted to Jason.

when you declare const, you have to declare value here itself.

If you are not initializing immediately, then it should go with let keyword.

See here we are using API because our purpose of this test case is only to make sure the created order is coming in to the history page or not. That is the requirement Then we can go through that API.

But if your requirement is to verify if you are creating an order successfully, then you should not do API. Bcze your test case itself is to verify if created order is success that you have to through webonly. if you do throught API, then that doesn't make sense.

In this case you are still doing UI testing only, but preconditions setup, you're relying on API,

46. Refactor API calls from utils folder and isolate from Web test logic

47. Part 2 - Refactor API calls from utils folder and isolate from Web test logic

let's create one utilities folder and we will move all this logging calls into that file.

Create new folder(utils). So all API utilities and Web utilities will push this into this folder.

Create new file in the folder(APIUtils.js)

Whenever you see all awaits in your steps, that means the whole method name Also you should Mark with async.

Now this apicontext, what you created in your actual test, you have to send this to

API utils file. To do that, what you can do here, you create one constructor.

when you create object of this class(APIUtils) from your test, then we will pass this apiContext

from our actual test.

const ApIUtils =new APIUtils(apiContext,loginpayload);

this above apiContext will come and fall in the constructor.

this constructor method will be automatically invoked when you create object for this class.

So that means you create an object(APIUtils) for that class in ur actual test by sending this parameter(apiConext,loginpayload)

So obviously this constructor method will be invoked and the parameter (apiconext,loginpayload), what you are sending there will be cast here in the constructor parameter.

this.apiConext = apiContext;

So from here you can assign that parameter to the local apiContext object.

So that means create one local API context object like this.

whatever apiContext you are sending from constructor, the scope will be inside this block only

now to get access to outside.

What you do is you create a class instance variable.

So this is your instance variable.

This will have access to entire class level because you are prefixed that with “this” this refers

to current class.

So this API context variable, I'm creating one more for this local class and I'm attaching to that

to this so that can be accessible anywhere.

One common thing is that you have to log in first so that log in information you should send mandatorily when you are calling any method in this class.

Finally, when everything is ready in your APIUTILS, you will simply say, export this file

If you don't call it export, then other files in this project do not know anything about this class.

module.exports = {ApiUtils};

So when you give like this, this class is now globally visible to all the files in your project.

if you don't give this then it is not visible and no other class can access this api utils class.

Note : Filename and classname should be same

API Uitils file

class APIUtils

{

    // this is just a precondition data setup

    constructor(apiContext,loginpayload)

    {

        this.apiContext  =  apiContext;

        this.loginpayload = loginpayload;

    }

    async gettoken()   // method name

    {

        const loginResponse = await  this.apiContext.post("https://rahulshettyacademy.com/api/ecom/auth/login",

        {

           data : this.loginpayload

        }  )

     //expect(loginResponse.ok()).toBeTruthy();

     const loginResponseJson = await loginResponse.json();  // return the JSON representation of response body

     const token = loginResponseJson.token;

     console.log(token);

     return token;

    }

    async CreateOrder(orderPayload)

    {

        let response = { };

        response.token = this.gettoken();

        const orderResponse = await this.apiContext.post("https://rahulshettyacademy.com/api/ecom/order/create-order",

 {

  data : orderPayload,

  headers:{

   'Authorization'   :  response.token,   //  this refers to current class.in this current class call the method of gettoken

   'Content-Type'   :  'application/json'

  },

 })

 const orderResponsejson = await orderResponse.json();

 console.log(orderResponsejson);

const orderId = orderResponsejson.orders[0];

 response.orderId=orderId;

 return response;  //it holds orderid & token

};

}

module.exports = {APIUtils};

WebAPIPart1.spec.js

const {test,expect,request} = require('@playwright/test');

const loginpayload = {userEmail: "anshika@gmail.com", userPassword: "Iamking@000"}

const orderPayload = {orders: [{country: "Cuba", productOrderedId: "6581ca399fd99c85e8ee7f45"}]}

const {APIUtils}=require('./utils/APIUtils');

let response;

test.beforeAll(  async() =>

{

   //login API

   const apiContext = await request.newContext();

   const ApIUtils = new APIUtils(apiContext,loginpayload);

   response = await ApIUtils.CreateOrder(orderPayload);

});

//Create order is success

test('place the order', async ({page})=>    {

    page.addInitScript(value => {

      window.localStorage.setItem('token',value);  // key : value pair

    }, response.token);

 await page.goto("https://rahulshettyacademy.com/client");

 await page.locator("button[routerlink\*='myorders']").click();

   await page.locator("tbody").waitFor();

   const rows = await page.locator("tbody tr");

   for (let i = 0; i < await rows.count(); ++i) {

      const rowOrderId = await rows.nth(i).locator("th").textContent();

      if (response.orderId.includes(rowOrderId)) {

         await rows.nth(i).locator("button").first().click();

         break;

      }

   }

   const orderIdDetails = await page.locator(".col-text").textContent();

  page.pause();

  expect(response.orderId.includes(orderIdDetails)).toBeTruthy();

});