6. Create npm Project and install Playwright dependencies for testing

File->Open folder->create one new folder(PLAYWRIGHTAUTOMATION)

one simple command which will help you to install playwright.And also it will create you one skeleton which is suitable to develop playwright project.

go to the terminal in your Visual Studio.

So that means we are creating one new Node project because playwright is ultimately writing on JavaScript, So whatever you develop on JavaScript, it's nothing but a node project.

npm init playwright - So playwright expect you to have one skeleton when you want to automate it. So that skeleton will automatically be create and also all the dependencies needed for automation, everything it will install also.

one single command initialize the playwright Project with npm init playwright

playwright.config.js – this is nothing but a test runner for your entire project.

Package.json - So this is the default file which gets created for every node project. ( Here we have the information of playwright dependencies )

Package-lock.json – this file related to package.json This is just an ignorable file for us.

Node\_modules – all the playwright related jars are installed in this node modules folder.

7. Importance of Playwright Test Annotation and async await understanding

create one new test under this test folder.

So right click, new file and let's call this as a (UIBasics.spec.js)

Import one annotation from playwright module all your playwright modules are installed in node\_modules they are nothing but playwright jars. so from these jars we have to import one annotation from there jars and we call it as test. (const {test} = require(‘@playwright/test’)

… These tests are executed in playwright environment that launches the browser and provides a fresh page to each test.

Test annotation takes (2 arguments) – 1st argument test case name, 2nd arguments as test function So the second argument is where you will write your actual code.

test(‘first playwright test’,function()

{

});

So this test annotation is coming from your playwright package.

 JavaScript is asynchronous. JavaScript, things will not get executed in this sequence.

3lines of code – emaiid – password – submit - There is no guarantee in JavaScript that these all three steps will execute in sequentially

whereas in JavaScript, all the three steps will try to execute at a time. That is what we call asynchronous.

If you write await before step one, it knows that I have to wait until the browser is completely open. And not only this one step, every step.

This await get activated only when you mark with function that async.

So in newer versions of JavaScript, if the function does not have any name.

So it just declared one test function and we declared a function. So we call this function as an anonymous function.

For anonymous function So from the newer versions of JavaScript, what you can do is you can skip writing this keyword called function. Instead, you can put one fat operator( => ) like this. so then it will be automatically treated as an anonymous function.

test(‘first playwright test’, async ()=>

{

await step1 : open borwser

await step2 : enter emailid

});

8. What is browser context and Page fixtures in Playwright ? – Example

browser - this is the fixture which comes by default from your playwight module.

this fixture is globally available to each and every test of your project.

So whenever you are creating a test annotation, there are some fixed set of fixtures(browser) which will be automatically available.

to use that browser inside of your function. We need to invoke browser, right for that we need this fixture.

general terminologies fixers are nothing but a global variables which are available across

your project.

So to represent and let this function know that this is specifically playwright fixers, you have to wrap that in curllybraces {browser}. If you don't give curly braces, then this is evaluated as like a normal browser string value.

Create a new context

.newContext - So this method will help you to open one fresh browser.I would call it as a fresh instance. This will be like incognito.

We passed browser as a parameter,

And we created one variable as ( context )

So if you simply pass “page” to another global variable then playwright it understood automatically that, this guy is in default mode. let me do these things for him. ( freshbrowser & newpage )

const {test} = require(‘@playwright/test’);

test(‘Browser context playwright test’, async ({browser}) =>

{

Chrome – plugins/cookies

Const context = await browser.newcontext(); // open fresh incognito browser

Const page = await context.newPage(); // [en new page/tab in browser

Await page.goto(“url”);

});

test(‘Browser context playwright test’, async ({page}) => // test word is annotation

{

Await page.goto(“url”);

});

Note : if you want to send any cookies/ plugins automaticallyin the incognito browser use browser fixer. If you don’t want anything to send we want fresh browser then use “page” fixer.

9. Importance of Playwright configuration file and its details to run the tests

Config.file

testDir : ‘./tests’ - this entire tests folder will get executed

timeout : 30 \* 1000 - timeout property up to how much time a test should wait before reporting failure.

expect property - is for assertions

reporter : ‘html’, - how do you want to report your test case results(html/json)

So this “use” property. Whatever you declare here your test cases will read all these properties here.

If you want to run in safari. There is a browser called the webkit.

This webkit It is default provided by playwright.

So when you install playwright, automatically this webkit will be installed. This is nothing but a safari engine. (browser = ‘webkit’)

So do remember that we are using that into a “use” property here so that will have all the information about metadata for your test like browser name and if you want to take screenshots, how many times it has to rerun your test if it fails retry mechanism and where logs should get stored. ( browserName : ‘chromium’ )

“npx” is the path which will point to the path of your playwright Module in the node modules.

Playwright - So this will directly point to the playwright model in this Node model package.

test – trigger the config file

( .only ) – if we use this if you have 50 testcases if you want to run only 1 test case in all the 50 tc’s. ( npm playwright test ) – tests will run in the backend and get the output

( npm playwright test –headed ) – browser will be open you can see the activities in the browser. ( double hifin before headed)

10. Running Playwright tests in multiple browsers - chrome, Firefox 7 Webkit

https://playwright.dev/docs/test-assertions

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

test('first playwright test', async ({browser})=>

{

   // chrome - plugins/cookie(parameter) browser : Browser

   const context = await browser.newContext();

   const page = await context.newPage();

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

   console.log(await page.title());

});

test.only('page playwright test', async ({page})=>

{

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

   //get title - assertion

   console.log(await page.title());

   await expect(page).toHaveTitle("Google");

});

// @ts-check

const { defineConfig, devices } = require('@playwright/test');

module.exports = defineConfig({

  testDir: './tests',

    timeout : 30\*1000,

  expect:  { // assertions

    timeout : 5000

  },

  reporter: 'html',

  use: {

    browserName : 'firefox',

       headless : true // browser not open test run backend

  },

});