67. What is page object pattern & Importance of its implementation

for every page We will create one JavaScript file and all the locators belongs to that page.

We'll go to that special file.

now for login page. We will create one login page object, JS file and in login you have this a userId, email, password, logging button, all these page objects. We will fill in this login js file

And now once you login then you will reach this dashboard page. no matter what operations you perform in the dashboard page, all those locators we will wrap in that dashboard page.

So like this, for every page, we will create the respective JS file and all the locators of that page

and all the logic methods will go into that JS files and at runtime. In our test case we will call those JS files and implement the code.

Create newfolder(Pageobjects) inside this create new file(LoginPage.js)

let's start by creating one class with the same name of the file name. And this is our class and inside this we will have one constructor.

Go to clientapp.spec.js copy all the login code ( url,email,password,click)

Note :  if you want to declare any locators, it's better to do that in the constructor itself.

when you create one variable inside a constructor and when you attach, start with “this”

So “this” belongs to current class. So this variable is nothing but class variable.

I am declaring all the initialization of this locators in the constructor itself so that when somebody creates an object for this class, all these variables will be initialized automatically.

So when you create a login page, you'll send this(page) argument to constructor.

So that way you can catch your page here.

I'm creating object for that login page class and I am sending this page value as an argument.

so this page value will come to the constructor which is present here.

Const loginpage = new LoginPage(page);

I’m just storing the locators in to one variable

so finally to make this class exposed to public and make available whole framework, you have

to export it

If you go outside of the folder( dot dot slash )

So we are getting this page from the constructor, which is okay, but the scope of this page variable will be inside only to make it available to the public. You have to create one local class variable called this dot page. And for this, you assign the actual page what you are getting.

Code for this video

New folder(Pageobjects) new file(LoginPage.js)

class LoginPage {

    constructor(page)

    {

        this.page = page;

        this.userName = page.locator("#userEmail");

        this.password = page.locator("#userPassword");

        this.signInbutton = page.locator("[value='Login']");

    }

    async goto()

    {

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

    }

    async validLogin(username,password)

    {

        await this.userName.type(username);

        await this.password.type(password);

        await this.signInbutton.click();

await page.waitForLoadState("networkidle");

    }

}

module.exports = {LoginPage};

ClientApp.PO.spec.js

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

const {LoginPage} = require('../Pageobjects/LoginPage');

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

    const username = "anshika@gmail.com";

    const password = "Iamking@000";

    const productName = 'ZARA COAT 3';

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

    const loginPage = new LoginPage(page);

    loginPage.goto();

    loginPage.validLogin(username,password);

}

68. Creating Page objects and action methods for end to end Script - Part 1

New file(DashboardPage)

class DashboardPage {

    constructor(page)

    {

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

        this.productsText = page.locator(".card-body b");

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

    }

    async searchProductAddCart(productName)

    {

        const graballTitles = await this.productsText.allTextContents();

// console.log(graballTitles);

 const count = await this.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;

}

 }

    }

   async navigatetoCart()

    {

       await this.cart.click();

    }

}

module.exports = {DashboardPage};

ClientApp.PO.spec.js

const {DashboardPage} = require('../Pageobjects/DashboardPage');

const dashboardPage = new DashboardPage(page);

    await dashboardPage.searchProductAddCart(productName);

    await dashboardPage.navigatetoCart();

69. Creating Page objects and action methods for end to end Script - Part 2

For each page we are creating new objects Can we create all these objects in one single file and simply call that file.

Under the Pageobjects folder, create another file (POManager.js)

So duty of this manager is to consolidate and keep the copy of all the objects.

Created an new file(POManager.js) in the Pageobjects folder

const {LoginPage} = require('./LoginPage');

const {DashboardPage} = require('./DashboardPage');

class POManager{

    constructor(page)

    {

        this.page = page;

        this.loginPage = new LoginPage(this.page);

        this.dashboardPage = new DashboardPage(this.page);

    }

    getLoginPage()

    {

        return this.loginPage;

    }

    getDashboardPage()

    {

        return this.dashboardPage;

    }

}

module.exports = {POManager};

ClientAppPO.spec.js

const {POManager} = require('../Pageobjects/POManager');

const poManager = new POManager(page);

    const loginPage = poManager.getLoginPage();

    const dashboardPage = poManager.getDashboardPage();

to run the file - npx playwright test tests/ClientApp.PO.spec.js

70. Assignment Solution - Converting Page object Model Project

In page objects folder ( each page has one file we created(cartpage,orderHistorypage,OrderReviewpage) based on the page of codes we placed in the respective file and we call all that in the main test(clientApp.po.spec.js)

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

 const {POManager} = require('../Pageobjects/POManager');

 test('Client App login', async ({page})=>

 {

   const poManager = new POManager(page);

    //js file- Login js, DashboardPage

     const username = "anshika@gmail.com";

     const password = "Iamking@000"

     const productName = 'Zara Coat 3';

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

     const loginPage = poManager.getLoginPage();

     await loginPage.goTo();

     await loginPage.validLogin(username,password);

     const dashboardPage = poManager.getDashboardPage();

     await dashboardPage.searchProductAddCart(productName);

     await dashboardPage.navigateToCart();

    const cartPage = poManager.getCartPage();

    await cartPage.VerifyProductIsDisplayed(productName);

    await cartPage.Checkout();

    const ordersReviewPage = poManager.getOrdersReviewPage();

    await ordersReviewPage.searchCountryAndSelect("ind","India");

    const orderId = await ordersReviewPage.SubmitAndGetOrderId();

   console.log(orderId);

   await dashboardPage.navigateToOrders();

   const ordersHistoryPage = poManager.getOrdersHistoryPage();

   await ordersHistoryPage.searchOrderAndSelect(orderId);

   expect(orderId.includes(await ordersHistoryPage.getOrderId())).toBeTruthy();

 });

71. How to drive the data from external json files to playwright tests

how to perform our data driven testing ?

So that means there is some data like user name, password, product name.

So is it good way to hard code in the test – no right

So you can do it sometimes it's better to drive the data from external file.

In the utils folder created one file(placeorderTestData.js)

{

    "username" : "anshika@gmail.com",

     "password" : "Iamking@000",

     "productName" : "Zara Coat 3"

}

Import this file in the main test file

There is method JSON.parse – use is convert JSON string in to an object means JavaScript object.

There is method JSON.stringify – use is convert JSON in to an string format

//JSON->String->js Object

 const dataset = JSON.parse(JSON.stringify(require("../utils/placeorderTestData.json")));

 test('Client App login', async ({page})=>

 {

   const poManager = new POManager(page);

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

     const loginPage = poManager.getLoginPage();

     await loginPage.goTo();

     await loginPage.validLogin(dataset.username,dataset.password);

     const dashboardPage = poManager.getDashboardPage();

     await dashboardPage.searchProductAddCart(dataset.productName);

     await dashboardPage.navigateToCart();

    const cartPage = poManager.getCartPage();

    await cartPage.VerifyProductIsDisplayed(dataset.productName);

    await cartPage.Checkout();

72. Implementing Parameterization in running tests with different data sets

In utils - placeorderTestData.js file

[

{

    "username" : "anshika@gmail.com",

     "password" : "Iamking@000",

     "productName" : "Zara Coat 3"

},

{

    "username" : "rahulshetty@gmail.com",

    "password" : "Iamking@000",

    "productName" : "Adidas Originals"

}

]

We need to write one for loop for the entire test

So I am wrapping up this whole test in one for loop

for(const data of dataset)

{

 test(`Client App login for ${data.productName}`, async ({page})=>

 {

   const poManager = new POManager(page);

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

     const loginPage = poManager.getLoginPage();

     await loginPage.goTo();

     await loginPage.validLogin(data.username,data.password);

     const dashboardPage = poManager.getDashboardPage();

     await dashboardPage.searchProductAddCart(data.productName);

     await dashboardPage.navigateToCart();

    const cartPage = poManager.getCartPage();

    await cartPage.VerifyProductIsDisplayed(data.productName);

    await cartPage.Checkout();

73. How to pass test data as fixture by extend test annotation behaviour

 this way of doing only supports one set of data validation.

You cannot apply parameterization when you send data as the fixture.

So these are default fixtures like “test” provided by playwright

So you can also provide custom fixtures for this test by extending its base properties.

I created one property for this test.

Now, this test know that there is one fixer, that knowledge we are giving.

And if anybody want to access that fixer, they will get all these data.

In utils folder creted one file(test-base.js)

const base = require('@playwright/test');

exports.customtest = base.test.extend(

    {

testDataForOrder :  {

     username : "anshika@gmail.com",

     password : "Iamking@000",

     productName : "Zara Coat 3"

        }

    }

)

In main test file

const {customtest} = require('../utils/test-base');

customtest(‘Client App login’, async ({page,testDataForOrder})=>

 {

   const poManager = new POManager(page);

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

     const loginPage = poManager.getLoginPage();

     await loginPage.goTo();

     await loginPage.validLogin(testDataForOrder.username,testDataForOrder.password);

     const dashboardPage = poManager.getDashboardPage();

     await dashboardPage.searchProductAddCart(testDataForOrder.productName);

     await dashboardPage.navigateToCart();

    const cartPage = poManager.getCartPage();

    await cartPage.VerifyProductIsDisplayed(testDataForOrder.productName);

    await cartPage.Checkout();

 });