**Cypress introduction**

Frontend Web automation testing tool.

Specially design for modern web applications.

Automate any application which run on browser.

React JS, Angular JS…

It supports only JavaScript.

Doesn’t use any selenium library.

It is open source.

Testrunner-Free

Dashboard-paid

We can work on Node.js environment and come along with npm module.

**Who can use cypress🡺** Dev and QA team

Specially it was develop for developer to perform unit and integration testing and slowly QA team also starts using this tool. We can automate

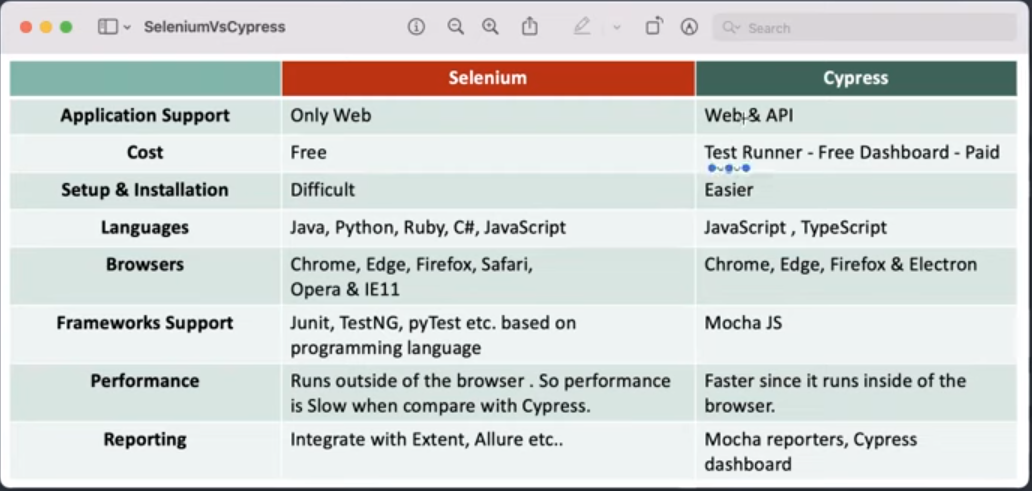
End to end test cases – QA people will prefer to use

Integration tests – Dev team to perform these

Unit test cases-- Dev team to perform these

API Testing-- QA people will prefer to use

**Difference between Selenium and Cypress**



**Cypress Ecosystem**

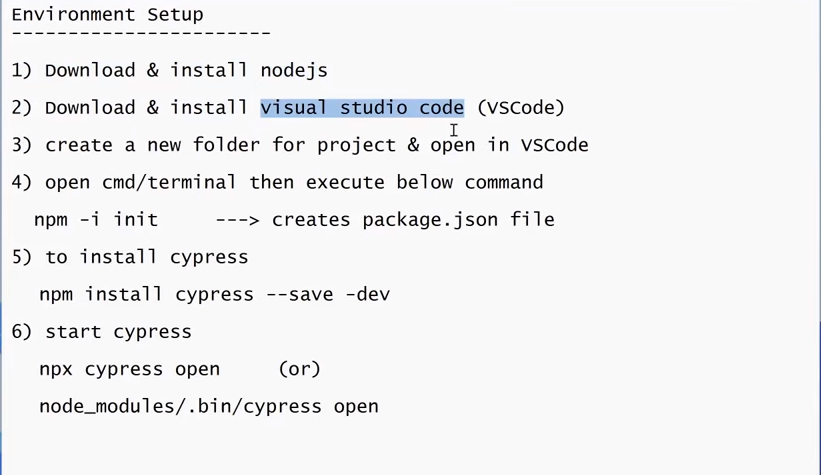
1. TestRunner – open source. Locally installed
2. Dashboard – record the test, history – paid

**Main features of Cypress**

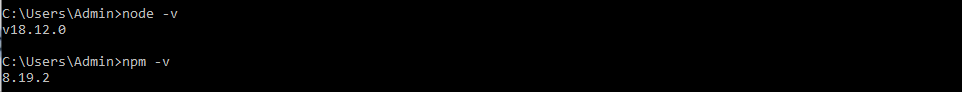
1. Time travel
2. Debuggability – it is very easy to Debugg the tests compare to other Automation tools.
3. Automatic waits (built in waits)—default wait for the element to perform the action.
4. Consistence result – we will get same result if we run the test n # of times.
5. Screen shots and videos are automatically capture form the test cases.
6. Cross browser testing🡪 run the test in multiple browsers.

**Limitation of Cypress**

1. We can automate window based /mobile applications.
2. Limited supports of browsers.
3. It supports only JavaScript/Typescript.
4. Reading and writing data into files is difficult.
5. 3rd party reporting tool integration is also limited.



After downloading node js, go to CMD promt and check it



Download VSCode

Create a folder for the project and open it in VS code.

Open terminal in VS Code and execute command: npm –i init

For the first time in VSCode, execute command: npm install cypress --save –dev

Once install cypress, the node module along with lib will be configure in the project.

**Writing and executing test**

To open cypress file. Do right click and new file and name the file with .cy.js

Syntax:

Describe(‘suite name’, ()=>

{

It(‘test1’,()=>

{

Expect(true).to.equal(true)

}

)

}

)

Each it block is test block.

Cy.visit : to open the url

Cy.title().should(‘eq’,’OrangeHRM’) : assertion

C:\Users\Admin\Desktop\cypressautomation> npx cypress run :command will execute the test in command with head less

C:\Users\Admin\Desktop\cypressautomation> npx cypress run - -headed :will execute the test in browser mode

C:\Users\Admin\Desktop\cypressautomation> npx cypress run - -spec <location of spec file> - -headed : will execute specific file of the location

C:\Users\Admin\Desktop\cypressautomation> npx cypress run - -browser chrome - - headed : test run in chrome

Test in Cypress

describe('MyFirst Test', ()=>{

    it('test1-verifing title-positive test', () =>{

       cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

       cy.title().should('eq','OrangeHRM')// Asserstion

    })

    it('test2-verifing title-negative test', () =>{

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

        cy.title().should('eq','OrangeHRM123')

     }

    )

})

**Cypress locators**

In cypress, it only support cssSelector and xpath.

Cy.get (Locator) 🡺 command to use to locate the element , it will use CSSSelector

* Tag#id
* Tag.Class
* Tag[attribute=’value’]
* Tag.Class[attribute=’value’]
* describe('CSSLocators',()=>{
* it('csslocator', ()=>{
* cy.visit("https://demo.nopcommerce.com/")
* cy.get("input[type='text']").type("Fahrenheit 451 by Ray Bradbury")// sending keys in the element
* cy.get("button[type='submit']").click()
* cy.get("body > div.master-wrapper-page").contains("Fahrenheit 451 by Ray Bradbury")//Assertion, verifying the text contains in the page
* })
* })

There is another way to locate the element in Cypress. That is Cypress xpath. For that we have to install the plug in the cypress where we can find it in cypress official website documents.

To install the cypress we have to run the command:

* npm install –D cypress –xpath <previous version>
* npm install -D @cypress/xpath <new versions>

To run Xpath in cypress, we will have to include “ require(‘@cypress/xpath’); in each project or in commands.js or e2e.js for all the project.

describe('XpathLocator',()=>{

    it('xpathlocator', ()=>{

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

        cy.xpath("//input[@name='username']").type('Admin');

        cy.xpath("//input[@name='password']").type('admin123');

        cy.xpath("//button").click()

        cy.url().should('eq','https://opensource-demo.orangehrmlive.com/web/index.php/dashboard/index')

       cy.xpath("//\*[@id='app']").contains('Paul Collings');

    })

})

**Cypress Assertions**

1. **Implicit assertions**

Should And🡺 eq, contain, include, not.contain, not.eq , not.include, be.visible, exist, have.lenght, have.value

describe("AssertionDemo",()=>{

    it("implicit assertion",()=>{

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

//should and

        // cy.url().should('include','orangehrmlive.com');//Asserstion

        // cy.url().should('eq',"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

        // cy.url().should('contain','orangehrm');

        // cy.url().should('include','orangehrmlive.com')

        // .should('eq',"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

        // .should('contain','orangehrm')

        cy.url().should('include','orangehrmlive.com')

        .and('eq',"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

        .and('contain','orangehrm')

        .and('not.contain','greenhrm')//negative assertion

        cy.title().should('include',"OrangeHRM")

        .and('not.include','orangeHRM')

        cy.get('.orangehrm-login-branding > img').should('be.visible') //Visibiltiy asserstion

        cy.get('.orangehrm-login-branding > img').should('exist') // existance asserstion

        cy.xpath('//a').should('have.length','5')//# of thelinks present or not

        cy.xpath("//input[@name='username']").type('Admin')

        cy.xpath("//input[@name='username']").should('have.value','Admin')

1. **Explicit assertions**: For this assertion, we will have to implement under the java script function.

Expect🡺 BDD –Behavior Driven Development

Asserts🡺TDD – Test Driven Development

describe("AssertionDemo", () => {

    it("Explicit assertion", () => {

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")

        cy.xpath("//input[@name='username']").type('Admin')

        cy.xpath("//input[@name='password']").type('admin123')

        cy.xpath("//button").click();

        let expName = "Paul Collings"

        cy.get(".oxd-userdropdown-name").then((x) => {

            let actName = x.text()

            //BDD sytle

            expect(actName).to.equal(expName)

            //expect(actName).to.not.equal(expName)

            //TDD style

            assert.equal(actName, expName)

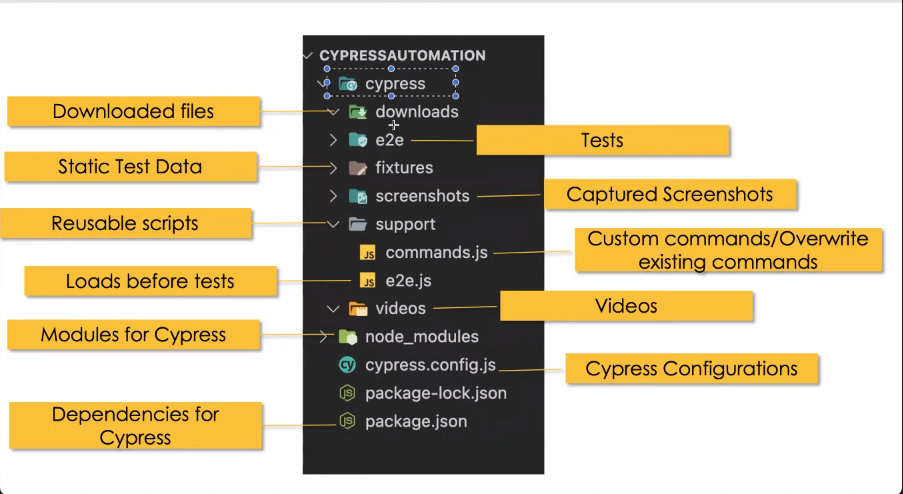
            //assert.not.equal(actName, expName)

        })

    })

})

**Folder structure**



Checkbox and radio button

describe("CheckUIElement",()=>{

// it("Checking Radio Butons",()=>{

//     cy.visit("https://itera-qa.azurewebsites.net/home/automation")

//     cy.get("#female").should('be.visible')//visibility of female radio button

//     cy.get("#male").should('be.visible')//visibility of male radio button

//     cy.get('#female').click().should('be.checked')

//     cy.get('#male').click().should('be.checked')

//     cy.get('#female').should('not.be.checked')

// })

it("Checking Radio Butons",()=>{

    cy.visit("https://itera-qa.azurewebsites.net/home/automation")

    // Selecting single check boxes

    //cy.xpath("//input[@type='checkbox' and contains(@id,'monday')]").should('be.visible')//visibility of single check box

    //cy.xpath("//input[@type='checkbox' and contains(@id,'monday')]").check().should('be.checked')//selecting the check box

   // cy.xpath("//input[@type='checkbox' and contains(@id,'monday')]").uncheck().should('not.be.checked')//selecting the check box

    //selecting all the check boxes

    //cy.xpath("//input[@type='checkbox' and contains(@id,'day')]").check().should('be.checked')

    cy.xpath("//input[@type='checkbox' and contains(@id,'day')]").first().check().should('be.checked')

    cy.xpath("//input[@type='checkbox' and contains(@id,'day')]").last().check().should('be.checked')

})

})

Dropdown

describe("DropDownhandling", () => {

  // it("Dropdown with select",()=>{

  //     cy.visit("https://www.zoho.com/commerce/free-demo.html")

  //   //  cy.get("#zcf\_address\_country").select("Nepal").should('have.value','Nepal')

  //     cy.get("#zcf\_address\_country").select("Nepal").should('eq','Nepal')

  // })

  //     it("Dropdown without select",()=>{

  //         cy.visit("https://www.dummyticket.com/dummy-ticket-for-visa-application/")

  //         cy.xpath("//\*[@id='billing\_country\_field']//span[@aria-label='Country']").click()

  //         cy.get('.select2-search\_\_field').type('Nepal').type('{enter}')

  //         cy.get('#select2-billing\_country-container').should('have.text','Nepal')

  //     })

  it("Dynamic dropdown", () => {

    cy.visit("https://www.google.com/")

    cy.get("input[name='q']").type('cypress automation')

    cy.wait(3000)

    cy.get('div.wM6W7d>span').should('have.length','11') //Assertion

      //using js function to itterate the

    cy.get('div.wM6W7d>span').each(($el, index, $list) => {

      if ($el.text() == 'cypress automation salary') {

        cy.wrap($el).click()

      }

    })

  })

})

Handling Alerts **=>** In cypress, alert windows will be automatically handle by cypress itself.

escribe("Alert", () => {

    // it('Js Alert',()=>{

    //     cy.visit("https://the-internet.herokuapp.com/javascript\_alerts")

    //     cy.get("button[onclick='jsAlert()']").click()

    //     //alert window automatically closed by cypress

    //     cy.on('window:alert',(t)=>{

    //         expect(t).to.contain('I am a JS Alert')

    //     })

    //     cy.get('#result').should('have.text','You successfully clicked an alert')

    // })

    // it('Js Alert', () => {

    //     cy.visit("https://the-internet.herokuapp.com/javascript\_alerts")

    //     cy.get(':nth-child(2) > button').click()

    //     //alert window automatically closed by cypress

    //     cy.on('window:confirm', (t) => {

    //         expect(t).to.contain('I am a JS Confirm')

    //     })

    //     cy.get('#result').should('have.text', 'You clicked: Ok')

    // })

    // it('Js Alert-cancle', () => {

    //     cy.visit("https://the-internet.herokuapp.com/javascript\_alerts")

    //     cy.get(':nth-child(2) > button').click()

    //     //alert window automatically closed by cypress

    //     cy.on('window:confirm', (t) => {

    //         expect(t).to.contain('I am a JS Confirm')

    //     })

    //     cy.on('window:confirm', (t) => false)

    //     cy.get('#result').should('have.text', 'You clicked: Cancel')

    // })

    it('Js prompt alert', () => {

        cy.visit("https://the-internet.herokuapp.com/javascript\_alerts")

        cy.window().then((win) => {

            cy.stub(win, 'prompt').returns('Welcome')

        })

        cy.get(':nth-child(3) > button').click()

        //alert window automatically closed by cypres

        cy.on('window:prompt', (t) => {

            expect(t).to.contain('I am a JS prompt')

        })

       // cy.on('window:confirm', (t) => false)

        cy.get('#result').should('have.text', 'You entered: Welcome')

    })

})

Iframe

1. describe('IframeDemo', () => {
2. it('Approch 1', () => {
3. cy.visit("https://the-internet.herokuapp.com/iframe");
4. const iframe = cy.get("#mce\_0\_ifr")
5. .its('0.contentDocument.body')
6. .should('be.visible')
7. .then(cy.wrap);
8. iframe.clear().type("Welcome to approch 1 {ctrl+a}");
9. cy.get("[aria-label='Bold']").click();
10. })
11. it('Approch 2 -using custom command', () => {
12. cy.visit("https://the-internet.herokuapp.com/iframe");
13. cy.getIframe("#mce\_0\_ifr").clear().type("Welcome to approch 2 {ctrl+a}");
14. cy.get("[aria-label='Bold']").click();
15. })

There is 3rd way to get iframe using plug in too

npm install -D cypress-iframe

**import** 'cypress-iframe';

*// or*

require('cypress-iframe');

Add ///<reference types="cypress-iframe" /> to the top of your cypress

Add a globals.d.ts in the root of your cypress directory and add ///<reference types="cypress-iframe" /> to it

1. import 'cypress-iframe';
2. describe('IframeDemo', () => {
3. it.only('Approch 3 -using iframe plugin', () => {
4. cy.visit("https://the-internet.herokuapp.com/iframe");
5. cy.frameLoaded("#mce\_0\_ifr")// load the frame
6. cy.iframe("#mce\_0\_ifr").clear().type("Welcome to approch 3 {ctrl+a}"); // select the text
7. cy.get("[aria-label='Bold']").click(); // bold the text
8. })
9. })

**Action –Mousehover and Rightclick action**

describe('Mousehover demo', ()=>{

    it('Mousehover', ()=>{

        cy.visit("https://demo.opencart.com/");

        cy.wait(3000);

        cy.xpath('//\*[@id="narbar-menu"]/ul/li[1]/div/div/ul/li[2]/a').should('not.be.visible');

        cy.xpath("//\*[text()='Desktops']").trigger('mousehover').click();

        cy.xpath('//\*[@id="narbar-menu"]/ul/li[1]/div/div/ul/li[2]/a').should('be.visible');

   cy.xpath("//\*[text()='Desktops']").trigger('contextmenu'); //right click

    })

})

**Double click**

describe('Mousehover demo', () => {

it('Doubleclick', ()=>{

      cy.visit("https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5\_ev\_ondblclick3");

        cy.wait(3000);

        cy.getIframe('#iframeResult').find("button[ondblclick='myFunction()']").trigger('dblclick');

        cy.getIframe('#iframeResult').find("#field2").should('have.value', 'Hello World!');

    })

**Drag and Drop**

This will need to install the plugin for this.

Install using npm:

npm install --save-dev @4tw/cypress-drag-drop

or yarn

yarn add --dev @4tw/cypress-drag-drop

Before Cypress is loaded (usually in your commands.js) put the following line:

require('@4tw/cypress-drag-drop')

Or, if you are using ES module syntax:

import '@4tw/cypress-drag-drop'

describe("Drag and drop", () => {

    it('Drag and drop', () => {

        cy.visit("http://dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-3.html")

        cy.wait(2000)

        cy.get("#box2").drag('box106');

    })

})

**Scroll**

describe("Scrolldown",()=>{

    it('Scroll it',()=>{

        cy.visit("https://www.opencart.com/");

        cy.wait(2000);

        cy.xpath(" //\*[@id='marketplace']//child::a[@class='btn btn-primary btn-xl']").scrollIntoView({duration:2000});

    })

})

**Navigate(go() command)**

describe('mysuite',()=>{

    it('Navigationtest',()=>{

        cy.visit("https://demo.opencart.com/");

        cy.wait(2000)

        cy.xpath("//\*[text()='Components']").trigger('mousehover').click();

        cy.xpath("//\*[@id='narbar-menu']/ul/li[3]/div/div").should('be.visible');

        cy.xpath("//\*[text()='Monitors (2)']").trigger('mousehover').click();

        cy.title().should('eq','Monitors')

        cy.go('back'); // navigating back

        cy.wait(2000)

        cy.go('forward')// navigating forward

        cy.wait(1000)

        cy.go(-1); // navigating back again using -1

        cy.wait(1000)

        cy.go(1);//navigating forward using 1

        cy.reload(); //reloading or refress

    })

})

**Screenshot=>** as soon as test is failed on cypress when run in command prompt, we automatically get the screenshot and videos.

describe('test',()=>{

    it('screenshot',()=>{

        cy.visit("https://demo.nopcommerce.com/");

        cy.screenshot("homepage");

        cy.get('.header-logo > a > img').screenshot("logo");

    })

})

**Hooks**

4types of hooks provided by Cypress

Before: execute only once before IT block

After: execute only once after IT block

BeforeEach🡺 multiple time for before each IT block

AfterEach🡺 multiple time for after each IT block

describe('MyTestSuite',()=>{

    before(()=>{

        cy.log("\*\*\*\*\* launch app \*\*\*\*\*\*");

    })

    after(()=>{

        cy.log("\*\*\*\*app quit \*\*\*\*\*\*");

    })

    beforeEach(()=>{

        cy.log("\*\*\*\*\*  Login  \*\*\*\*\*\*");

    })

    afterEach(()=>{

        cy.log("\*\*\* Logout \*\*\*\*\*\*");

    })

    it('search', ()=>{

        cy.log(" \*\*\*\*\* searching \*\*\*\*\*\*");

    })

    it('advance search',()=>{

        cy.log(" \*\*\*\*\* advance searching \*\*\*\*\*\*");

    }

    )

    it('listingProduct', ()=>{

        cy.log(" \*\*\*\*\* Listing product \*\*\*\*\*\*");

    })

}

)

**UPLOAD FILE**

First we have to install the file upload package

npm install --save-dev cypress-file-upload

and will have to import ‘cypress-file-upload’; we need to have file saved inside fixtures folder in cypress project

**ShadowDom**{includeShadowDom:true}

describe('ShadowDome Test', ()=>{

    it('Shadow Dome Approch',()=>

    {

        cy.visit('https://the-internet.herokuapp.com/shadowdom');

        cy.get('li:nth-child(1)',{includeShadowDom:true}).then((shadowtext)=>{

            let innertext = shadowtext.text();

            let expecttext = "Let's have some different text!";

            cy.log(innertext);

            assert.equal(innertext,expecttext);

        })

    })

})

**CustomCommand**

///<reference types="Cypress" />

Cypress.Commands.add('getIframe',(iframe)=>{

     return cy.get(iframe)

    .its('0.contentDocument.body')

    .should('be.visible')

    .then(cy.wrap);

})

//Custom command for clicking on link using label

Cypress.Commands.add('clickLink',(label)=>{

    cy.get('a').contains(label).click();

})

//Overwrite Contains function

Cypress.Commands.overwrite('contains',(originalFn,subject,filter,text,options = {})=>{

    if(typeof text === 'object'){

        options = text

        text = filter

        filter = undefined

    }

    options.matchCase = false

   return originalFn(subject,filter,text,options)

})

//Custom Command for login

Cypress.Commands.add('loginApp',(email,password)=>{

    cy.get('#Email').type(email);

    cy.get('#Password').type(password);

    cy.get("button[class='button-1 login-button']").click();

})

describe("Custom Command Demo", () => {

    it("handling link", () => {

        cy.visit("https://demo.nopcommerce.com/");

        // //  without using custom command

        //         cy.get('body > div:nth-child(7) > div:nth-child(3) > div:nth-child(1) > div:nth-child(1) > div:nth-child(1) > div:nth-child(1) > div:nth-child(4) > div:nth-child(2) > div:nth-child(2) > div:nth-child(1) > div:nth-child(2) > h2:nth-child(1) > a:nth-child(1)').click();

        //         cy.get("div[class='product-name'] h1").should('have.text', 'Apple MacBook Pro 13-inch');

        // Using custom command

        cy.clickLink("Apple MacBook Pro 13-inch");

        cy.get("div[class='product-name'] h1").should('have.text', 'Apple MacBook Pro 13-inch');

    })

    it("overwriting existing command", () => {

        cy.visit("https://demo.nopcommerce.com/");

        cy.clickLink("APPLE macbook PRO 13-inch");

        cy.get("div[class='product-name'] h1").should('have.text', 'Apple MacBook Pro 13-inch');

    })

    it.only("login command", () => {

        cy.visit("https://demo.nopcommerce.com/");

        cy.clickLink('Log in');

        cy.loginApp("xyz", "123");

    })

})

**DataDriven Test**

All the file that need to read by cypress should be the part of Features file in project.

{

    "username": "Admin",

    "password": "admin123",

    "expected": "PIM"

}

describe("MyTestSuit", () => {

    it("DataDriven login", () => {

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

        ////hardcore

        // cy.get("input[placeholder='Username']").type('Admin');

        // cy.get("input[placeholder='Password']").type('admin123');

        // cy.get("button[type='submit']").click();

        // cy.get('body > div:nth-child(3) > div:nth-child(1) > div:nth-child(1) > aside:nth-child(1) > nav:nth-child(1) > div:nth-child(2) > ul:nth-child(2) > li:nth-child(2) > a:nth-child(1) > span:nth-child(2)').should('have.text','PIM');

        //Using DataDriven

        cy.fixture('orangehrm').then((data) => {

            cy.get("input[placeholder='Username']").type(data.username);

            cy.get("input[placeholder='Password']").type(data.password);

            cy.get("button[type='submit']").click();

            cy.get('body > div:nth-child(3) > div:nth-child(1) > div:nth-child(1) > aside:nth-child(1) > nav:nth-child(1) > div:nth-child(2) > ul:nth-child(2) > li:nth-child(2) > a:nth-child(1) > span:nth-child(2)').should('have.text', data.expected);

        })

    })

    let userdata;

    before(() => {

        cy.fixture('orangehrm').then((data) => {

            userdata = data;

        })

        it.only("FixtureDemo Test", () => {

            cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

            cy.get("input[placeholder='Username']").type(userdata.username);

            cy.get("input[placeholder='Password']").type(userdata.password);

            cy.get("button[type='submit']").click();

            cy.get('body > div:nth-child(3) > div:nth-child(1) > div:nth-child(1) > aside:nth-child(1) > nav:nth-child(1) > div:nth-child(2) > ul:nth-child(2) > li:nth-child(2) > a:nth-child(1) > span:nth-child(2)').should('have.text', userdata.expected);

        })

    })

})

[{

    "username": "Admin",

    "password": "admin123",

    "expected": "PIM"

},

{

    "username": "Azymin",

    "password": "admin123",

    "expected": "Invalid credentials"

},

{

    "username": "Admin",

    "password": "admin",

    "expected": "Invalid credentials"

}]

describe("MyTestSuit DataDriven test2", () => {

    it("DataDriven login", () => {

        cy.fixture('orangehrm2').then((data) => {

            cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

            data.forEach((userdata) => {

                cy.get("input[placeholder='Username']").type(userdata.username);

                cy.get("input[placeholder='Password']").type(userdata.password);

                cy.get("button[type='submit']").click();

                if (userdata.username == "Admin" && userdata.password == "admin123") {

                    cy.get('body > div:nth-child(3) > div:nth-child(1) > div:nth-child(1) > aside:nth-child(1) > nav:nth-child(1) > div:nth-child(2) > ul:nth-child(2) > li:nth-child(2) > a:nth-child(1) > span:nth-child(2)').should('have.text', userdata.expected);

                    cy.get('.oxd-userdropdown-tab > .oxd-icon').click();

                    cy.get(':nth-child(4) > .oxd-userdropdown-link').click();

                }

                else {

                    cy.get(".oxd-text.oxd-text--p.oxd-alert-content-text").should("have.text", userdata.expected);

                }

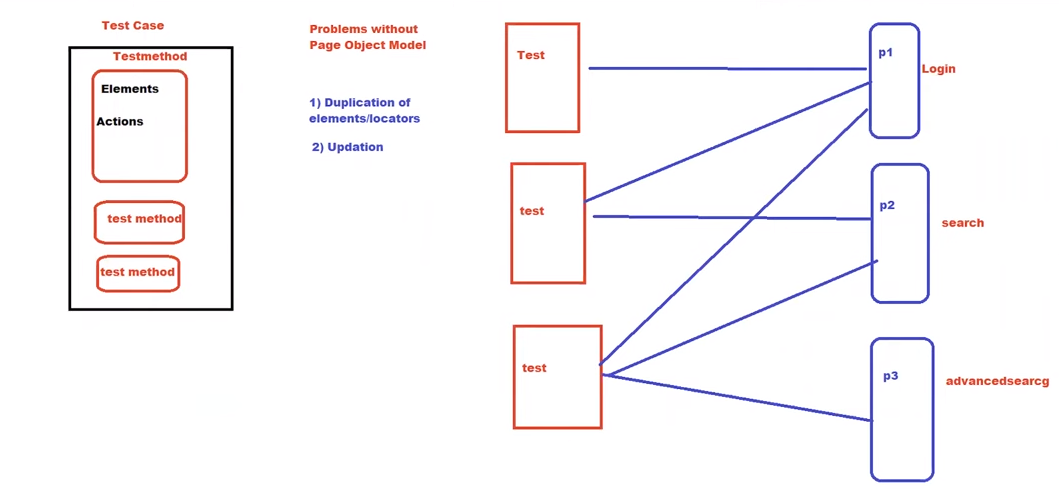
            })

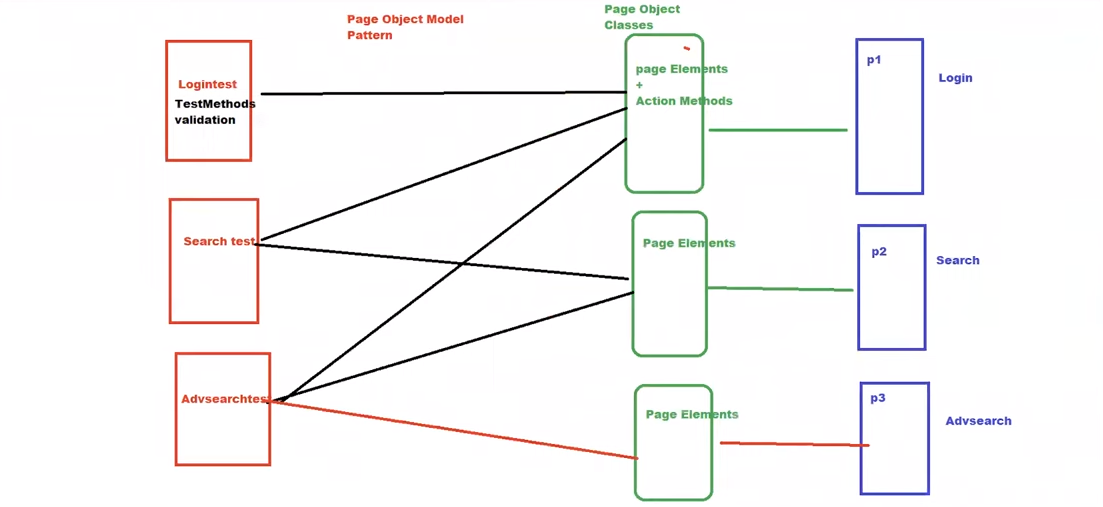
        })

    })

})

**PAGE OBJECT MODEL PATTERN**

****

****

**Login object class**

class Login {

    txtUserName="input[placeholder='Username']";

    txtPassword="input[placeholder='Password']";

    btnSubmit="button[type='submit']"

    lblmsg=".oxd-text.oxd-text--h6.oxd-topbar-header-breadcrumb-module";

    setUserName(username) {

       cy.get(this.txtUserName).type(username);

    }

    setPassword(password) {

        cy.get(this.txtPassword).type(password);

    }

    clickSubmit() {

        cy.get(this.btnSubmit).click();

    }

    verifyLogin() {

        cy.get(this.lblmsg).should('have.text', 'Dashboard');

    }

}

export default Login;

**fixture file**

{

    "username": "Admin",

    "password": "admin123",

    "expected": "Dashboard"

}

**Test.js**

import Login from "../PageObjectModel/LoginPage2"

describe("MyTestSuit", () => {

    //using POM

    it.only("LoginTest using POM", () => {

        cy.visit("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

        cy.fixture('orangehrm').then((data)=>{

            const ln = new Login(); // creating object

            ln.setUserName(data.username);

            ln.setPassword(data.password);

            ln.clickSubmit();

            ln.verifyLogin();

        })

**HTML REPORT FOR CYPRESS**

**Setup**

This setup tutorial works with Cypress >= v10, looking for older version setup? [**here**](https://github.com/LironEr/cypress-mochawesome-reporter/blob/9c11e7005351e8750fe48b90d010c9bf29539956/README.md#setup)

1. install cypress-mochawesome-reporter
2. npm i --save-dev cypress-mochawesome-reporter

or

yarn add -D cypress-mochawesome-reporter

1. Change cypress reporter & setup hooks

Edit config file (cypress.config.js by default)

const { defineConfig } = require('cypress');

module.exports = defineConfig({

reporter: 'cypress-mochawesome-reporter',

e2e: {

setupNodeEvents(on, config) {

require('cypress-mochawesome-reporter/plugin')(on);

},

},

});

If you are override before:run or after:run hooks, use this:

const { defineConfig } = require('cypress');

const { beforeRunHook, afterRunHook } = require('cypress-mochawesome-reporter/lib');

module.exports = defineConfig({

reporter: 'cypress-mochawesome-reporter',

e2e: {

setupNodeEvents(on, config) {

on('before:run', async (details) => {

console.log('override before:run');

await beforeRunHook(details);

});

on('after:run', async () => {

console.log('override after:run');

await afterRunHook();

});

},

},

});

1. Add to cypress/support/e2e.js

import 'cypress-mochawesome-reporter/register';

1. run cypress

**npx cypress run –headed --spec cypress\e2e\MyFirstTest.cy.js –browser chrome**