151. Create new Maven Project and all Framework dependencies

Rightclick->new->click on project->open maven->select maven project->next->next->

here where you have a lot of actual archetypes are present. So basically for every project maven by readymade giving a skeleton. So that means if you work on some backend Java projects like Springboat.

So they're already given one skeleton. On how project structure should look like for Springboard apps. So basically some already defined templates are there.

So if you choose them, then readymade folders will be created for you.

So it's just not the code, just the folders.

Hierarchies will be created for you based upon the artifact you select.

Now coming to automation. So there is one default artifact. So basically this is a simple default maven project This is a default template. So for automation, you can directly choose this default template. All that folders are something which we can create. Just search in the filter field ( maven-archetype-quickstart) – this is artifact id select it -> click on next. After it asks GroupID & ArtifactID

Artifact ID - is nothing but your project name ( SeleniumFrameworkdesign)

group ID - means it's like a upper hierarchy to artifact. So let's say there is one company (rahulshettyacademy) So in this I teach Selenium, APM, rest API testing.

Different modules are there. So each module is one department.

So the main company name here is nothing but a group ID. So here our group ID is Rahul Shetty Academy. Under this group there are multiple sub modules.

So for selenium we have one artifact. but all these come under parent group ID.

package here is automatically constructed, so that means how the packages in your project should be. If you want you can modify are keep it same. Click on finish

one Maven default project is created for us. – you can check the artifactID,GroupID, package name.

pom.xml is default generated when you create a default maven project. Double click the pom.xml file.

once the project is entirely developed, now let's say if you want to publish this project

to Maven repository, you can do that with these three simple details. ( groupID, Artifact ID, version )

So in our entire project now we are going to use selenium as for dependencies. So we need selenium jars to run this project.

Now let's get selenium maven dependencies.

Google – mavenrepository.com -> search for selenium ->select selenium java-> select the latest version (Ex:4.3.2) -> if you want this Selenium Java jars into your project automatically just copy these four lines of code or  copy all these dependencies. ->

Paste it in pom.xml file b/w <dependencies> </dependencies> -> Save

So once it built and there will be one folder will be created for you called maven dependencies. If you expand it here where you will see all the selenium jars, were automatically imported to your maven project by simply passing this dependency.

So similarly, I need testNG dependency. do the same above process.

So in this framework we are going to use testNG as a main testing framework.

152. Ecommerce Application Demo which will be used in the Framework

To design a framework, we need one end to and test case.

Open this URL - <https://rahulshettyacademy.com/client/>

Register the account with you email id & password.

So our goal here is to log in -> add to cart few products -> product added to cart validation msg is displaying->going to the cart section to verify if those products are displayed in the cart.

whatever items we selected in the previous page are displaying in the cart. then let's check out so that you will land on this page.

Fill all the details and click on place to order. make sure that you are on confirmation page. Like Thank you.

And also grab the orders. So if you place order with one product, only one order it will display. So if you place with two orders, then two orders will display and you can come Back to orders tab here just to verify that what all orders you placed are showing up here. The order number, what you see on the confirmation page there, you can grab and you can come to order tab here and verify if it is showing up here.

And then you can click on View and make sure that everything is same,

all the details you can confirm, and finally you can delete your order also.

Now just go back and click on delete.

Message orders, delete it successfully and that order disappears and you can go back to shopping again by selecting go back to shop.

So this is an end to end e-commerce website

So first of all, before we jump into all automation main framework topics, we will discuss to develop one standalone test which places an order.

So that standalone Selenium test, we will try to convert into framework standards using page objects, data-driven, parallel testing.

But before doing, we need one sample selenium test.

153. Selenium Program on WebDriverManager - Login- Get products list

So for next 30 minutes I'll be just writing Selenium test. I'm not writing any framework related things here because we need to build a Selenium test and then convert it into framework standards.

Expand the src folder -> right click on the package -> click on class-> class name ( standAloneTest) -> check the public static void main -> finish

let's use that web driver manager again to invoke Chrome browser.

By default, web driver manager is not available in Selenium JARs.

So go to Maven repository search WebDriverManager->click on latest version. Copy the 4 lines of code and paste it in pom.xml file.

Now your test can understand if you write any code related to web driver manager.

1. WebDriverManager.chromedriver().setup()

So traditionally we used to give system.setproperty, WebDriver.chrome.driver.

And here with a simple step, now Chrome driver will be automatically downloaded

into your system based upon your Chrome browser version.

Now you can directly start by creating object for your chrome driver

WebDriver driver = new ChromeDriver();

//put some implicit weight on global level – to neglect any sync issues,

driver.manage().timeouts().implicitlywait(Duration.ofSeconds(10));

driver.get(“hit the url”);

// enter username & password & click on login

d.fE(By.id(“ “).sendskey(“emailid “);

d.fE(By.id(“ “).sendskey(“password “);

d.fE(By.id(“ “).click; // click on login

there are a lot of products displayed on the screen. So how we will exactly click on this?

So for that, first you need to get all these products list and then you iterate through each and every item in the list and verify for which product this title(ZARA COAT 3) is displayed and then capture that product and click on Add to Cart.

// wait until all the products are displayed

WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(5));

Wait.until(ExpectedConditions.visibilityofElementLocated(By.Css(“.mb-3”)));

 //find a common locator for all the products in to list

List<WebElement> products = driver.findElements(By.css(“.mb-3”));

154. Selenium Program to retrieve product and Add to Cart based on Java

stream will help you to iterate through each and every item present in the products. Every iteration it pic one item then I will apply filter to check one condition

// prod, product is new variable we created

WebElement prod = products.stream().filter(product->product.fE(By.CssSelector(“b”)).getText().equals(“ZARA COAT 3”)).findFirst().orElse(null);

// click on add to cart

Prod.findElement(By.CssSelector(“.card-body button:last-of-type”)).click();

if you see error anywhere for the above 2 line of code that means your Java compliance is on the older version. just right click go to Properties, and go to Java Compiler, So make sure the minimum is 1.8 is what you need to select. Then click on apply and close then you won’t get any error here.

155. Implementation of explicit wait to handle application synchronously on loading

once you add to cart, you see some two things are happening here.

See this is loading and you will get a toast to message that product added to cart validation msg.

So you have to make sure first of all, toast message is displaying and also this loading gets invisible. that loading icon, we have to wait until that loading

gets invisible before you perform any operations. So basically you want to wait until that toast is displayed then only you can confirm that product is successfully added to the cart. Until then you have to wait. We use explicit wait

// validation msg

Wait.until(ExpectedConditions.visibilityofElementLocated(By.Css(“#toast-container”)));

So we have to wait until that loading icon disappears.

// loading icon – ( webelement – ng-animating )

Wait.until(ExpectedConditions.invisibilityof(driver.findElement(By.Css(“.ng-animating”)));

//click on cart

driver.fE(By.Css(“[routerlink\*='cart']”)).click();

156. Logic to verify items in the cart with Streams and Checkout

once we add items to CART and we are going to cart page.

our goal is to verify whatever we added the product in the dashboard page is

appearing in this cart or not we need to validate.

we have to scan all these products showing in the My Cart page to see if there is a product with name of this ZARA COAT 3.

String productName = “ZARA COAT 3”;

// check the list of items present in my cart

List<WebElement> cartpoducts = d.fEs(By.css(“.cartSection h3”));

Boolean match = cartproducts.stream().anyMatch(cartproduct->cartproduct.getText().equalsIgnorecase(productName);

Assert.assertTrue(match);

If it matches for any of the product, it will return boolean value as the return type as a true.

// click on checkout button

d.fE(By.css(“.totalRow button”)).click();

157. Wrapping up end to end automation Script on Purchasing Order in Ecommerce app.

there is a class called actions in Java for advanced selenium interactions. if you want to send information into that edit box, you can do driver or find elements and send keys. Instead I am trying to do using actions class

Actions a = new Actions(driver);

// entire country field

a.sendkeys(d.fE(By.css(“[placeholdr=’select Country’]”)), “india”).build().perform();

// wait until the list should be visible ( Ex : if we enter “In” list should be display)

wait.until(ExpectedConditions.visbilityofElementLocated(By.css(“.ta-results”)));

// in the visible list select 2nd one in to the visible list

d.fE(By.xpath(By.xpath("(//button[contains(@class,'ta-item')])[2]")).click();

// click on place holder button

d.fE(By.css(“.action\_submit”)).click();

// check the displaying text is correct or not

String confirmMessage = d.fE(By.css(“.hero-primary”)).getText();

Assert.assertTrue(confirmMessage.equalsIgnoreCase(“THANKYOU FOR THE ORDER.”));

driver.close():

**package** rahulshettyacademy.SeleniumFrameworkDesign;

**import** java.time.Duration;

**import** java.util.List;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.edge.EdgeDriver;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.testng.Assert;

**import** io.github.bonigarcia.wdm.WebDriverManager;

**public** **class** StandAlone\_Test {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

WebDriverManager.*chromedriver*().setup();

WebDriver driver = **new** ChromeDriver();

driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10));

driver.get("https://rahulshettyacademy.com/client");

driver.findElement(By.*id*("userEmail")).sendKeys("harisankar722@gmail.com");

driver.findElement(By.*id*("userPassword")).sendKeys("H@rish777");

driver.findElement(By.*id*("login")).click();

// wait until all the products are displayed

WebDriverWait wait = **new** WebDriverWait(driver,Duration.*ofSeconds*(5));

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*(".mb-3")));

//find a common locator for all the products in to list

List<WebElement> products = driver.findElements(By.*cssSelector*(".mb-3"));

WebElement prod = products.stream().filter(product->product.findElement(By.*cssSelector*("b")).getText().equals("ZARA COAT 3")).findFirst().orElse(**null**);

// click on Add to Cart

prod.findElement(By.*cssSelector*(".card-body button:last-of-type")).click();

//validation msg

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*("#toast-container")));

// wait until that loading icon disappears - ( WebElement – .ng-animating )

wait.until(ExpectedConditions.*invisibilityOf*(driver.findElement(By.*cssSelector*(".ng-animating"))));

// click on cart button

driver.findElement(By.*cssSelector*("[routerlink\*='cart']")).click();

String productName = "ZARA COAT 3";

// check the list of items present in my cart

List<WebElement> cartpoducts = driver.findElements(By.*cssSelector*(".cartSection h3"));

Boolean match = cartpoducts.stream().anyMatch(cartproduct->cartproduct.getText().equalsIgnoreCase(productName));

Assert.*assertTrue*(match);

// If it matches for any of the product, it will return boolean value as the return type as true.

// click on checkout button

driver.findElement(By.*cssSelector*(".totalRow button")).click();

Actions a = **new** Actions(driver);

// entire country field

a.sendKeys(driver.findElement(By.*cssSelector*("[placeholder='Select Country']")), "india").build().perform();

// wait until the list should be visible ( Ex : if we enter "In" list should be display)

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*(".ta-results")));

// in the visible list select 2nd one in to the visible list

driver.findElement(By.*xpath*("(//button[contains(@class,'ta-item')])[2]")).click();

// click on place holder button

driver.findElement(By.*cssSelector*(".action\_\_submit")).click();

// check the displaying text is correct or not

String confirmMessage = driver.findElement(By.*cssSelector*(".hero-primary")).getText();

Assert.*assertTrue*(confirmMessage.equalsIgnoreCase("THANKYOU FOR THE ORDER."));

driver.close();

}

}