selenium topics and test cases

Table of Contents

[basic navigation 1](#_Toc9404614)

[Test cases 1](#_Toc9404615)

[open browser, get page, get title 1](#_Toc9404616)

[Open browser, get page, get URL 1](#_Toc9404617)

[CODE EXAMPLE 1](#_Toc9404618)

[get page, get title 1](#_Toc9404619)

[get, navigate to, back, forward, refresh 2](#_Toc9404620)

[navigate to, back, forward 3](#_Toc9404621)

[get url 3](#_Toc9404622)

[Basic locators, web element class 4](#_Toc9404623)

[Test cases 4](#_Toc9404624)

[Basic authentication Address book 4](#_Toc9404625)

[Basic authentication Zero bank 4](#_Toc9404626)

[Search Amazon 5](#_Toc9404627)

[Retrieve password 5](#_Toc9404628)

[Click on links 5](#_Toc9404629)

[CODE EXAMPLE 5](#_Toc9404630)

[maximize browser, id, name, class name, sendkeys, click 5](#_Toc9404631)

[implicit wait vs thread.sleep, tagname, sendkeys 7](#_Toc9404632)

[id, name, linktext, partial linktext, sendkeys, click 8](#_Toc9404633)

[tagname 8](#_Toc9404634)

[handling different elements,: input, checkbox, radiobutton, dropdown; xpath 9](#_Toc9404635)

[Test cases 9](#_Toc9404636)

[Verify checkbox functionality 9](#_Toc9404637)

[Mercury Tours radio button functionality 9](#_Toc9404638)

[Mercury Tours departure list 9](#_Toc9404639)

[input box 10](#_Toc9404640)

[input 11](#_Toc9404641)

[input 11](#_Toc9404642)

[dropdown 12](#_Toc9404643)

[radio buttons 13](#_Toc9404644)

[Checkboxes 14](#_Toc9404645)

[TestNG Intro 15](#_Toc9404646)

[Assertions 15](#_Toc9404647)

[@beforemethod @aftermethod 16](#_Toc9404648)

[beforemethod vs before class 17](#_Toc9404649)

[after method vs after class 17](#_Toc9404650)

[mixed annotations 18](#_Toc9404651)

[testng order, depends on 19](#_Toc9404652)

[depends on 19](#_Toc9404653)

[Mixed dependencies 20](#_Toc9404654)

[more dependencies 20](#_Toc9404655)

[priority, test that fails 21](#_Toc9404656)

[Tabs, frames, alerts 23](#_Toc9404657)

[LINKS 24](#_Toc9404658)

[pop up 24](#_Toc9404659)

[Iframe 25](#_Toc9404660)

[Iframe switch options 26](#_Toc9404661)

[Javascript alerts 31](#_Toc9404662)

[Switch tabs 32](#_Toc9404663)

[HOMEWORK 35](#_Toc9404664)

[SOFT ASSERTIONS, TEST BASE, ACTIONS, FILE DOWNLOAD 35](#_Toc9404665)

[Test base 35](#_Toc9404666)

[Actions double click 36](#_Toc9404667)

[Actions drag and drop 37](#_Toc9404668)

[Actions hover 38](#_Toc9404669)

[Verify file exists 39](#_Toc9404670)

[Test that extends test base 39](#_Toc9404671)

[Soft assert 1 40](#_Toc9404672)

[Soft assert 2 40](#_Toc9404673)

[HOMEWORK 42](#_Toc9404674)

[FILE UPLOAD. WEB TABLES 42](#_Toc9404675)

[Table: select whole table, select column names, select certain row, select by coordinate, select row, table size, index 42](#_Toc9404676)

[Xpath: index, parent 42](#_Toc9404677)

[HOMEWORK 48](#_Toc9404678)

[EXCEPTIONS REVIEW, VERIFY ELEMENT EXISTS 49](#_Toc9404679)

[Negative verifications Verify element does NOT exist 49](#_Toc9404680)

[PAGE OBJECT MODEL 51](#_Toc9404681)

[Test Cases 51](#_Toc9404682)

[Wrong password 51](#_Toc9404683)

[No username 51](#_Toc9404684)

[No password 51](#_Toc9404685)

[CODE EXAMPLE 51](#_Toc9404686)

[Driver class 51](#_Toc9404687)

[Test Base using driver class 52](#_Toc9404688)

[PrestaShopHomePage 52](#_Toc9404689)

[ListingPage 52](#_Toc9404690)

[HomePage 52](#_Toc9404691)

[Tests using page object model 52](#_Toc9404692)

[Homework 54](#_Toc9404693)

[PROPERTIES FILE, DRIVER CLASS 54](#_Toc9404694)

[ConfigurationReader 54](#_Toc9404695)

[Driver class with configuration reader 55](#_Toc9404696)

[HomePage with configuration reader 56](#_Toc9404697)

[Test Base 57](#_Toc9404698)

[Properties file 58](#_Toc9404699)

[Tests with page object and configuration reader 58](#_Toc9404700)

[Tests with page object and configuration reader 2 59](#_Toc9404701)

[HOMEWORK 61](#_Toc9404702)

[WAITS AND SYNCHRONIZATION 61](#_Toc9404703)

[Explicit wait 61](#_Toc9404704)

[Thread.sleep, implicit wait 62](#_Toc9404705)

# basic navigation

## Test cases

open browser, get page, get title  
1. Open chrome browser  
2. go to url "https://google.com"  
3. Verify title  
Expected title: Google

driver.getTitle(); --> get the title of the page, returns a string

Open browser, get page, get URL  
1. Open chrome browser  
2. go to url "https://google.com"  
3. Verify url  
Expected url: "https://www.google.com/"

driver.getCurrentUrl() --> gets the url of the current page

## CODE EXAMPLE

### get page, get title

|  |  |
| --- | --- |
|  | public class FirstSeleniumEver { |
|  | public static void main(String[] args) { |
|  | // Step 1 open chrome |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | // Step 2. go to url "https://google.com" |
|  | driver.get("https://google.com"); |
|  | // Step 3. Verify title |
|  | // Expected title: Google |
|  | String actualTitle = driver.getTitle(); |
|  | System.out.println("actualTitle = " + actualTitle); |
|  | String expectedTitle = "Google"; |
|  | // verifying |
|  | if (expectedTitle.equals(actualTitle)){ |
|  | System.out.println("PASS"); |
|  | } else { |
|  | System.out.println("FAIL"); |
|  | System.out.println("Expected: " + expectedTitle); |
|  | System.out.println("Actual: " + actualTitle); |
|  | } |
|  | } |
|  | } |

### get, navigate to, back, forward, refresh

|  |  |
| --- | --- |
|  | public class FirstSelenium { |
|  |  |
|  | public static void main(String[] args) { |
|  | // we cannot open a browser without setting the path drivers |
|  | // this line must come before creating the webdriver object |
|  |  |
|  | // first argument is the type driver |
|  | System.setProperty("webdriver.chrome.driver", |
|  | // location of the driver |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | // this line open a chrome browser |
|  | WebDriver driver = new ChromeDriver(); |
|  | // open a website |
|  | driver.get("https://google.com"); |
|  | driver.get("https://etsy.com"); |
|  | driver.navigate().to("https://amazon.com"); |
|  | driver.navigate().back(); |
|  | driver.navigate().forward(); |
|  | driver.navigate().refresh(); // refreshes the page |
|  | // BREAK |
|  | } |
|  | } |

### navigate to, back, forward

|  |  |
| --- | --- |
|  | public class SecondSelenium { |
|  | public static void main(String[] args) { |
|  | System.setProperty("webdriver.gecko.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/geckodriver"); |
|  | WebDriver driver = new FirefoxDriver(); // this line open firefox driver |
|  | driver.get("https://etsy.com"); |
|  | driver.navigate().to("https://amazon.com"); |
|  | driver.navigate().back(); |
|  | driver.navigate().forward(); |
|  | } |
|  | } |

### get url

|  |
| --- |
| /\* |
|  | 1. Open chrome browser |
|  | 2. go to url "https://google.com" |
|  | 3. Verify url |
|  | Expected url: "https://www.google.com/" |
|  |  |
|  | \*/ |
|  | public class VerifyURLTest { |
|  | // THIS TEST IS SUPPOSED TO FAIL, mmy code is failing |
|  | // your may pass |
|  | public static void main(String[] args) { |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.get("http://google.com"); |
|  | String expectedUrl = "https://www.google.com/"; |
|  | String actualUrl = driver.getCurrentUrl() ; |
|  | if (expectedUrl.equals(actualUrl)){ |
|  | System.out.println("PASS"); |
|  | } else { |
|  | System.out.println("FAIL"); |
|  | System.out.println("actualUrl = " + actualUrl); |
|  | System.out.println("expectedUrl = " + expectedUrl); |
|  | } |
|  | System.out.println(driver.getPageSource()); // gets the HTML of the page |
|  | } |
|  | } |
|  |  |

# Basic locators, web element class

## Test cases

### Basic authentication Address book

1. Open browser
2. Go to *http://a.testaddressbook.com/sign\_in*
3. Enter username *kexesobepu@zsero.com*
4. Enter password *password*
5. Click sign in button
6. Verify username is displayed on page
7. Verify the title does not include *Sign In*

### Basic authentication Zero bank

1. Open browser
2. Go to *http://zero.webappsecurity.com/login.html*
3. Enter username *username*
4. Enter password *password*
5. Click sign in button
6. Verify *username* is displayed on page
7. Verify the title *Zero - Account Summary*

### Search Amazon

1. Open browser
2. Go to *https://amazon.com*
3. Enter any search term
4. Click on search button
5. Verify title contains the search term

### Retrieve password

1. Open browser
2. Go to *https://* *the-internet.herokuapp.com/forgot\_password*
3. Enter any email
4. Click on *Retrieve password*
5. Verify url contains *email\_sent*

### Click on links

1. Open browser
2. Go to *https://* *the-internet.herokuapp.com/*
3. Click link *File download*
4. Verify title contains on *download*
5. Navigate to previous page
6. Click link *File upload*
7. Verify title contains on *upload*

## CODE EXAMPLE

### maximize browser, id, name, class name, sendkeys, click

|  |
| --- |
|  |
|  | public class BasicAuthenticationAddressBook { |
|  | public static void main(String[] args) { |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | // HOW TO MAXIMIZE THE PAGE |
|  | // driver.manage().window().fullscreen(); // mac |
|  | // driver.manage().window().maximize(); // windows |
|  | driver.get("http://a.testaddressbook.com/sign\_in"); |
|  | // finding element from page |
|  | WebElement inputEmail = driver.findElement(By.id("session\_email")); |
|  | // enter the username |
|  | inputEmail.sendKeys("kexesobepu@zsero.com"); |
|  | // find the password input box |
|  | WebElement inputPassword = driver.findElement(By.id("session\_password")); |
|  | // enter password |
|  | inputPassword.sendKeys("password"); |
|  | // locates the sign in button |
|  | WebElement signinButton = driver.findElement(By.name("commit")); |
|  | System.out.println(driver.getTitle()); |
|  | // clicks on the button |
|  | signinButton.click(); |
|  | System.out.println(driver.getTitle()); |
|  | // located the username element |
|  | WebElement username = driver.findElement(By.className("navbar-text")); |
|  | String actualUsername = username.getText(); |
|  | System.out.println(actualUsername); |
|  |  |
|  | // Verify username is displayed on page |
|  | if ("kexesobepu@zsero.com".equals(actualUsername)){ |
|  | System.out.println("PASS"); |
|  | } else { |
|  | System.out.println("FAIL"); |
|  | System.out.println("EXPECTED: kexesobepu@zsero.com"); |
|  | System.out.println("actualUsername = " + actualUsername); |
|  | } |
|  | // verify title |
|  | if ( !driver.getTitle().contains("Sign In") ) { |
|  | System.out.println("PASS"); |
|  | } |
|  | else { |
|  | System.out.println("FAIL"); |
|  | System.out.println(driver.getTitle()); |
|  | } |
|  | } |
|  | } |
|  |  |

### implicit wait vs thread.sleep, tagname, sendkeys

|  |  |
| --- | --- |
|  | public class ImplicitWait { |
|  | public static void main(String[] args) throws InterruptedException { |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | // this line makes the findElement method wait for some time when it cannot find an element |
|  | // findElement method will keep trying to locate the element during the given duration |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | driver.get("http://www.cheapfansedge.info"); |
|  | driver.findElement(By.tagName("inpsdfsdfut")).sendKeys("asdfasdfd"); |
|  | // HARD CODED WAIT |
|  | Thread.sleep(5000); |
|  | driver.get("https://google.com"); |
|  | } |
|  | } |

### id, name, linktext, partial linktext, sendkeys, click

|  |  |
| --- | --- |
|  | public class LogOutTest { |
|  | public static void main(String[] args) { |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.get("http://a.testaddressbook.com/sign\_in"); |
|  | WebElement inputEmail = driver.findElement(By.id("session\_email")); |
|  | inputEmail.sendKeys("kexesobepu@zsero.com"); |
|  | WebElement inputPassword = driver.findElement(By.id("session\_password")); |
|  | inputPassword.sendKeys("password"); |
|  | WebElement signinButton = driver.findElement(By.name("commit")); |
|  | signinButton.click(); |
|  |  |
|  | WebElement lougOutLink = driver.findElement(By.linkText("Sign out")); |
|  | lougOutLink.click(); |
|  |  |
|  | WebElement homeLink = driver.findElement(By.partialLinkText("Hom")); |
|  | homeLink.click(); |
|  | } |
|  | } |

### tagname

|  |
| --- |
|  |
|  | public class RetrievePassword { |
|  | public static void main(String[] args) { |
|  | System.setProperty("webdriver.chrome.driver", |
|  | "/Users/cybertekschool/Documents/selenium dependencies/drivers/chromedriver"); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.get("https://the-internet.herokuapp.com/forgot\_password"); |
|  | WebElement email = driver.findElement(By.tagName("input")); |
|  | email.sendKeys("example@gmail.com"); |
|  | WebElement button = driver.findElement(By.tagName("button")); |
|  | button.click(); |
|  | } } |

# handling different elements,: input, checkbox, radiobutton, dropdown; xpath

## Test cases

### Verify checkbox functionality

1. Open browser
2. Go to <http://the-internet.herokuapp.com/checkboxes>
3. Verify the first checkbox is not selected
4. Verify the second checkbox is selected
5. Click on the first checkbox
6. Verify the first checkbox is selected
7. Verify the second checkbox is selected
8. Click on the second checkbox
9. Verify the first checkbox is selected
10. Verify the second checkbox is not selected

### Mercury Tours radio button functionality

1. Open browser
2. Go to <http://www.newtours.demoaut.com/>
3. Login using username tutorial and password tutorial
4. Verify Round Trip is selected by default
5. Verify One Way is not selected by default
6. Click one way
7. Verify Round Trip selected not selected
8. Verify One Way is selected

### Mercury Tours departure list

1. Open browser
2. Go to <http://www.newtours.demoaut.com/>
3. Login using username tutorial and password tutorial
4. Verify default Departing From value is Acapulco
5. Verify list of locations: Acapulco, Frankfurt, London, New York, Paris, Portland, San Francisco, Seattle, Sydney, Zurich
6. Test Departing From dropdown functionality by changing different values

## input box

|  |
| --- |
| /\* |
|  | 1. open browser |
|  | 2. go to amazon |
|  | 3. enter search term and submit |
|  | 4. verify search box still contains the same search term |
|  | \*/ |
|  | public class AmazonSearch { |
|  | public static void main(String[] args) { |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | // get --> go to some url |
|  | driver.get("https://amazon.com"); |
|  | String searchterm = "wooden spoon"; |
|  | driver.findElement(By.id("twotabsearchtextbox")) |
|  | .sendKeys(searchterm+ Keys.ENTER); |
|  | String actual = driver. |
|  | findElement(By.id("twotabsearchtextbox")) |
|  | .getAttribute("value"); |
|  | if (searchterm.equals(actual)) { |
|  | System.out.println("PASS"); |
|  | } else { |
|  | System.out.println("FAIL"); |
|  | System.out.println("actual = " + actual); |
|  | System.out.println("searchterm = " + searchterm); |
|  | } |

## input

|  |  |
| --- | --- |
|  | public class EtsyExample { |
|  | public static void main(String[] args) { |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | driver.get("https://etsy.com"); |
|  | WebElement input = driver.findElement(By.id("search-query")); |
|  | input.sendKeys("decal"); |
|  | WebElement submit = driver.findElement(By.xpath("//button[@class='btn btn-primary']")); |
|  | submit.click(); |
|  | } |
|  | } |

## input

|  |
| --- |
| /\* |
|  | go to https://the-internet.herokuapp.com/forgot\_password |
|  | verify enter email and submit |
|  | verify email sent message on the next page |
|  | "Your e-mail's been sent!" |
|  | \*/ |
|  | public class SubmitEmail { |
|  | public static void main(String[] args) { |
|  | // setting driver path for chrome using the webdriver manager |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.get("https://the-internet.herokuapp.com/forgot\_password"); |
|  | WebElement input = driver.findElement(By.id("email")); |
|  | // input.sendKeys("somemeail@somemail.com"+ Keys.ENTER); |
|  | input.sendKeys("somemeail@somemail.com"); |
|  | input.submit(); |
|  | String expected = "Your e-mail's been sent!"; |
|  | String actual = |
|  | driver.findElement(By.id("content")).getText(); |
|  | if (actual.equals(expected)) { |
|  | System.out.println("PASS"); |
|  | } else { |
|  | System.out.println("FAIL"); |
|  | System.out.println("actual = " + actual); |
|  | System.out.println("expected = " + expected); |
|  | } |
|  | } |
|  | } |

## dropdown

|  |  |
| --- | --- |
|  |  |
|  | public static void main(String[] args) { |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | driver.get("https://the-internet.herokuapp.com/dropdown"); |
|  | // 1. we have to locate an element with tag select |
|  | WebElement selectElement = driver.findElement(By.id("dropdown")); |
|  |  |
|  | // 2. create select object using the webelement |
|  | Select list = new Select(selectElement); |
|  | // print the selected option |
|  | String selectedOption = list.getFirstSelectedOption().getText(); |
|  | System.out.println(selectedOption); |
|  | // print all available options |
|  |  |
|  | List<WebElement> options = list.getOptions(); |
|  | System.out.println("options.size() = " + options.size()); |
|  | for (WebElement option:options) { |
|  | System.out.println(option.getText()); |
|  | } |
|  | // select different options |
|  | // 1. by visible text |
|  | list.selectByVisibleText("Option 1"); |
|  | System.out.println("Selected option: " + |
|  | list.getFirstSelectedOption().getText()); |
|  |  |
|  | // 2. by visible index, count is 0 based |
|  | list.selectByIndex(2); |
|  | System.out.println("Selected option: " + |
|  | list.getFirstSelectedOption().getText()); |
|  | // 3. by value attribute |
|  | list.selectByValue("1"); |
|  | System.out.println("Selected option: " + |
|  | list.getFirstSelectedOption().getText()); |
|  | } |
|  | } |

## radio buttons

|  |  |
| --- | --- |
|  | public class RadioButtons { |
|  | public static void main(String[] args) { |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  |  |
|  | driver.get("http://samples.gwtproject.org/samples/Showcase/Showcase.html#!CwRadioButton"); |
|  | // verify blue and red not checked |
|  | WebElement blue=driver.findElement(By.id("gwt-debug-cwRadioButton-color-blue-input")); |
|  | WebElement red=driver.findElement(By.id("gwt-debug-cwRadioButton-color-red-input")); |
|  | System.out.println("DEFAULT"); |
|  | System.out.println("blue: " + blue.isSelected()); |
|  | System.out.println("red: " + red.isSelected()); |
|  | // check blue |
|  | System.out.println("Clicking on blue"); |
|  | blue.click(); |
|  | // verify blue checked, red not checked |
|  | System.out.println("blue: " + blue.isSelected()); |
|  | System.out.println("red: " + red.isSelected()); |
|  | // check red |
|  | System.out.println("Clicking red"); |
|  | red.click(); |
|  | // verify blue not checked, red is checked |
|  | System.out.println("blue: " + blue.isSelected()); |
|  | System.out.println("red: " + red.isSelected()); |
|  | } // 3.15 break |
|  | } |

## Checkboxes

|  |
| --- |
| /\* |
|  | go to http://samples.gwtproject.org/samples/Showcase/Showcase.html#!CwCheckBox |
|  | verify monday is not selected |
|  | select monday |
|  | verify monday is selected |
|  | unselect monday |
|  | \*/ |
|  | public class ReallyCheckboxes { |
|  | public static void main(String[] args) { |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | WebElement monday = driver.findElement(By.id("gwt-debug-cwCheckBox-Monday-input")); |
|  | // System.out.println(monday.isSelected()); |
|  | //// monday.click(); |
|  | //// System.out.println(monday.isSelected()); |
|  | //// monday.click(); |
|  | //// System.out.println(monday.isSelected()); |
|  | //// |
|  | System.out.println(monday.isSelected()); |
|  | if (!monday.isSelected()){ |
|  | monday.click(); |
|  | } |
|  | System.out.println(monday.isSelected()); |
|  | } |
|  | } |

# TestNG Intro

## Assertions

|  |  |
| --- | --- |
|  | public class AssertionsDemo { |
|  | @Test |
|  | public void test1(){ |
|  | String a = "A"; |
|  | String b = "A"; |
|  | Assert.assertEquals(a, b); // comparing a to b |
|  | } |
|  | @Test |
|  | public void test2(){ |
|  | String a = "A"; |
|  | String b = "AAA"; |
|  | Assert.assertEquals(a, b); |
|  | // comparing a to b, first param is actual second is expected |
|  | } |
|  |  |
|  | @Test |
|  | public void test3(){ |
|  | String a = "ADAFD"; |
|  | String b = "AAA"; |
|  | Assert.assertTrue(b.contains(a)); |
|  | Assert.assertTrue(true); |
|  | System.out.println("DONE"); |
|  | } |
|  | @Test |
|  | public void test4(){ |
|  | String a = "ADAFD"; |
|  | String b = "AAA"; |
|  | Assert.assertTrue(b.contains(a), |
|  | b + " should have contained "+ a); |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void test5(){ |
|  | Assert.assertTrue(false); |
|  | } |
|  | } |

## @beforemethod @aftermethod

|  |  |
| --- | --- |
|  | public class GoogleTest { |
|  | WebDriver driver; |
|  | @BeforeMethod |
|  | public void setUp(){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  | driver.get("https://google.com"); |
|  | } |
|  | @Test |
|  | public void test(){ |
|  | Assert.assertTrue(driver.getTitle().contains("asdf")); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDown(){ |
|  | driver.quit(); |
|  | } |
|  | } |

## beforemethod vs before class

|  |
| --- |
|  |
|  | public class DifferentAnnotations {  @Test |
|  | public void testOne(){ |
|  | System.out.println("This is test one"); |
|  | } |
|  | @BeforeMethod |
|  | public void beforeMethod(){ |
|  | System.out.println("This is before method"); |
|  | } |
|  |  |
|  | @Test |
|  | public void testTwo(){ |
|  | System.out.println("This is test two"); |
|  | } |
|  | @BeforeClass |
|  | public void beforeClass(){ |
|  | System.out.println("This is before class"); |
|  | } |
|  | } |

## after method vs after class

|  |  |
| --- | --- |
|  | public class DifferentAnnotations2 { |
|  | public void testOne(){ |
|  | System.out.println("This is test one"); |
|  | } |
|  | @Test |
|  | public void testTwo(){ |
|  | System.out.println("This is test two"); |
|  | } |
|  | @AfterMethod |
|  | public void afterMethod(){ |
|  | System.out.println("This is after method"); |
|  | } |
|  | @AfterClass |
|  | public void afterClass(){ |
|  | System.out.println("This is after class"); |
|  | } |
|  | } |

## mixed annotations

|  |
| --- |
|  |
|  | public class MixedAnnotations { |
|  | Webriver driver; |
|  | @BeforeClass |
|  | public void beforeClass(){ |
|  | System.out.println("Setting up the drivers here"); |
|  | WebDriverManager.chromedriver().setup(); |
|  | } |
|  | @BeforeMethod |
|  | public void setUpTMethod(){ |
|  | System.out.println("Opening a browser"); |
|  | driver = new ChromeDriver(); |
|  | System.out.println("go to website and login"); |
|  | driver.get("https://google.com"); |
|  | } |
|  | @Test |
|  | public void testOne(){ |
|  | System.out.println("Testing testing testing: account info"); |
|  | } |
|  | @AfterMethod |
|  | public void afterMethod(){ |
|  | System.out.println("logout"); |
|  | System.out.println("closing browser"); |
|  | driver.quit(); |
|  |  |
|  | } |
|  | @AfterClass |
|  | public void afterClass(){ |
|  | System.out.println("report everything"); |
|  | } |
|  | @Test |
|  | public void testTwo(){ |
|  | System.out.println("Testing testing testing: payment"); |
|  | } |
|  | @Test |
|  | public void testThree(){ |
|  | System.out.println("Testing testing testing: another thing"); |
|  | } |
|  | } |

# testng order, depends on

## depends on

|  |  |
| --- | --- |
|  | public class DependenciesTest { |
|  | @Test |
|  | public void login(){ |
|  | System.out.println("login"); |
|  | Assert.assertTrue(false); |
|  | } |
|  | @Test(dependsOnMethods = "login") |
|  | public void zbuySth(){ |
|  | System.out.println("buy buy baby"); |
|  | } |
|  | @Test(priority = 5) |
|  | public void homePage(){ |
|  | System.out.println("home page"); |
|  | } |

## Mixed dependencies

|  |  |
| --- | --- |
|  | public class MixedUpDependencies { |
|  | @Test() |
|  | public void openBrowser(){ |
|  | System.out.println("opening browser"); |
|  | } |
|  | @Test(dependsOnMethods = "openBrowser") |
|  | public void login(){ |
|  | System.out.println("login"); |
|  | } |
|  | @AfterMethod |
|  | public void cleanUp(){ |
|  | System.out.println("close browser"); |
|  | } |
|  | } |

## more dependencies

|  |
| --- |
|  |
|  | public class MoreDependenciesTest { |
|  | @Test() |
|  | public void openBrowser(){ |
|  | System.out.println("opening browser"); |
|  | } |
|  | @Test(dependsOnMethods = "openBrowser") |
|  | public void login(){ |
|  | System.out.println("login"); |
|  | Assert.assertTrue(false); |
|  | } |
|  | @Test(dependsOnMethods = "login" ) |
|  | public void logOut(){ |
|  | System.out.println("logout"); |
|  | } |
|  |  |

## priority, test that fails

|  |
| --- |
|  |
|  | // THIS TEST WILL FAIL |
|  | /\* |
|  | 1. open browser |
|  | 2.go to https://amazon.com |
|  | 3.verify that default dropdown option is all |
|  | 4.verify that all options are sorted alphabetically |
|  | 5.Click on the menu icon on the left |
|  | 6.click on Full Store directory |
|  | 7.capture all of the main department names in the page. |
|  | 8.verify that departments names are sorted alphabetically |
|  | 9.verify that all departments names are listed in the dropdown from step 4 |
|  | \*/ |
|  | public class DepartmentsTest { |
|  |  |
|  | WebDriver driver; |
|  | Select allDepartments; |
|  |  |
|  | @BeforeClass |
|  | public void setUpMethod() { |
|  | WebDriverManager.chromedriver().setup(); |
|  |  |
|  | driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | driver.get("https://amazon.com"); |
|  | } |
|  |  |
|  |  |
|  | // verify that default dropdown option is all |
|  | @Test (priority = 0) |
|  | public void testDefaultValue() { |
|  | // finding the element with a select tag |
|  | WebElement selectElement = driver.findElement(By.id("searchDropdownBox")); |
|  | // create a Select class object |
|  | allDepartments = new Select(selectElement); |
|  | // get the current selected option |
|  | String selectedOption = allDepartments.getFirstSelectedOption().getText(); |
|  | Assert.assertEquals(selectedOption, "All"); |
|  |  |
|  | } |
|  |  |
|  | // 4.verify that all options are sorted alphabetically |
|  | @Test (priority = 1) |
|  | public void sortAlphabetical(){ |
|  | // using the compare to method, we will compare each element in the list to the next one |
|  |  |
|  | for (int i = 0; i < allDepartments.getOptions().size()-1; i++) { |
|  | String current = allDepartments.getOptions().get(i).getText(); |
|  | String next = allDepartments.getOptions().get(i+1).getText(); |
|  |  |
|  | System.out.println("comparing: " + current + " with "+ next); |
|  |  |
|  | Assert.assertTrue(current.compareTo(next)<=0); |
|  |  |
|  | } |
|  | } |
|  | // 5.Click on the menu icon on the left |
|  | // 6.click on Full Store directory |
|  | // 7.capture all of the main department names in the page. |
|  | // 8.verify that departments names are sorted alphabetically |
|  |  |
|  | @Test |
|  | public void mainDepartmentsTest() throws InterruptedException { |
|  | // 5.Click on the menu icon on the left |
|  |  |
|  | driver.findElement(By.cssSelector("i.hm-icon.nav-sprite")).click(); |
|  |  |
|  | // click on Full Store directory |
|  | // waiting because it is taking time for element to load and be clickable |
|  | Thread.sleep(2000); |
|  | driver.findElement(By.cssSelector("a[href='/gp/site-directory?ref\_=nav\_em\_T1\_0\_2\_2\_33\_\_fullstore']")).click(); |
|  |  |
|  | // capture all of the main department names in the page. |
|  |  |
|  | List<WebElement> mainDepts = driver.findElements(By.cssSelector("h2.fsdDeptTitle")); |
|  |  |
|  | for (WebElement option : |
|  | mainDepts) { |
|  | System.out.println(option.getText()); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @AfterClass |
|  | public void tearDown(){ |
|  | // driver.quit(); |
|  | } |
|  | } |

# Tabs, frames, alerts

## LINKS

Alerts

<http://the-internet.herokuapp.com/javascript_alerts>

<https://sweetalert.js.org/>

Pop ups

<https://sweetalert.js.org/>

<https://www.primefaces.org/showcase/ui/overlay/confirmDialog.xhtml>

<https://www.w3schools.com/howto/howto_css_modals.asp>

Windows

<http://the-internet.herokuapp.com/windows>

## pop up

|  |  |
| --- | --- |
|  | public class HTMLPopup { |
|  |  |
|  | WebDriver driver; |
|  |  |
|  | @BeforeMethod |
|  | public void setUp() { |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  |  |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void cleanUp() throws InterruptedException { |
|  | Thread.sleep(5000); |
|  | driver.quit(); |
|  | } |
|  |  |
|  | @Test |
|  | public void doNotDestroyTheWorldTest() { |
|  | driver.get("https://www.primefaces.org/showcase/ui/overlay/confirmDialog.xhtml"); |
|  | driver.findElement(By.xpath("//span[.='Destroy the World']")).click(); |
|  | // locate an element in the pop up window and click |
|  | driver.findElement(By.xpath("//span[.='No']")).click(); |
|  | } |
|  |  |
|  | @Test |
|  | public void oopsTest() { |
|  | driver.get("https://sweetalert.js.org/"); |
|  | // trigger the pop up |
|  | driver.findElement(By.xpath("(//button[@class='preview'])[2]")).click(); |
|  | // close the pop up |
|  | driver.findElement(By.xpath("//button[.='OK']")).click(); |
|  |  |
|  |  |
|  | } |
|  | } |

## Iframe

|  |
| --- |
|  |
|  | /\*1. create a new class IframeDemo |
|  | 2. go to url https://the-internet.herokuapp.com/tinymce |
|  | 3. enter text in the text editor |
|  |  |
|  | \*/ |
|  |  |
|  | import io.github.bonigarcia.wdm.WebDriverManager; |
|  | import org.openqa.selenium.By; |
|  | import org.openqa.selenium.WebDriver; |
|  | import org.openqa.selenium.WebElement; |
|  | import org.openqa.selenium.chrome.ChromeDriver; |
|  | import org.testng.annotations.Test; |
|  |  |
|  | import java.util.concurrent.TimeUnit; |
|  |  |
|  | public class IframeDemo { |
|  |  |
|  | @Test |
|  | public void test(){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | WebDriver driver = new ChromeDriver(); |
|  |  |
|  | driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); |
|  | driver.get("https://the-internet.herokuapp.com/tinymce"); |
|  |  |
|  | // locate the iframe |
|  | WebElement iframe = driver.findElement(By.id("mce\_0\_ifr")); |
|  | // switch to iframe |
|  | driver.switchTo().frame(iframe); |
|  |  |
|  |  |
|  |  |
|  | driver.findElement(By.tagName("body")). |
|  | sendKeys("ddeee dddedd dd"); |
|  |  |
|  |  |
|  | } |
|  | } |
|  |  |
|  |  |

## Iframe switch options

|  |  |
| --- | --- |
|  | public class SwitchOptions { |
|  |  |
|  | WebDriver driver; |
|  |  |
|  | @BeforeMethod |
|  | public void setUp(){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  | driver.get("https://the-internet.herokuapp.com/tinymce"); |
|  |  |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void leanUp(){ |
|  | driver.quit(); |
|  | } |
|  |  |
|  | @Test |
|  | public void switchByElement(){ |
|  | // locate the iframe |
|  | WebElement iframe = driver.findElement(By.id("mce\_0\_ifr")); |
|  | // switch to iframe |
|  | driver.switchTo().frame(iframe); |
|  | driver.findElement(By.tagName("body")). |
|  | sendKeys("ddeee dddedd dd"); |
|  |  |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @Test |
|  | public void switchById(){ |
|  | // switch to iframe |
|  | driver.switchTo().frame("mce\_0\_ifr"); |
|  | driver.findElement(By.tagName("body")). |
|  | sendKeys("ddeee dddedd dd"); |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void switchByIndex(){ |
|  | // switch to iframe |
|  | driver.switchTo().frame(0); |
|  | driver.findElement(By.tagName("body")). |
|  | sendKeys("ddeee dddedd dd"); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |

## Javascript alerts

|  |
| --- |
|  |
|  | public class JSAlerts { |
|  | WebDriver driver; |
|  |  |
|  | @BeforeMethod |
|  | public void setUp() { |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  |  |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void cleanUp() throws InterruptedException { |
|  | Thread.sleep(5000); |
|  | driver.quit(); |
|  | } |
|  |  |
|  | @Test |
|  | public void oopsTest(){ |
|  | driver.get("https://sweetalert.js.org/"); |
|  | // trigger the pop up |
|  | driver.findElement(By.xpath("(//button[@class='preview'])[1]")).click(); |
|  |  |
|  | // switch to alert |
|  | Alert alert = driver.switchTo().alert(); |
|  |  |
|  | alert.accept(); // clicking OK on a pop up |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void dismiss(){ |
|  | driver.get("http://the-internet.herokuapp.com/javascript\_alerts"); |
|  | // trigger the pop up |
|  | driver.findElement(By.xpath("//button[@onclick='jsConfirm()']")).click(); |
|  | Alert alert = driver.switchTo().alert(); |
|  | alert.dismiss(); // clicking Cancel on a pop up |
|  |  |
|  | } |
|  |  |
|  | @Test |
|  | public void alertText(){ |
|  | driver.get("http://the-internet.herokuapp.com/javascript\_alerts"); |
|  | // trigger the pop up |
|  | driver.findElement(By.xpath("//button[@onclick='jsPrompt()']")).click(); |
|  | Alert alert = driver.switchTo().alert(); |
|  | alert.sendKeys("Hello"); // typing on the alert |
|  | alert.accept(); |
|  | } |
|  |  |
|  | } |
|  |  |

## Switch tabs

|  |
| --- |
|  |
|  | public class TabsExample { |
|  |  |
|  | WebDriver driver; |
|  |  |
|  | @BeforeMethod |
|  | public void setUp(){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  | driver.get("http://the-internet.herokuapp.com/windows"); |
|  |  |
|  | } |
|  |  |
|  | // @AfterMethod |
|  | public void cleanUp() throws InterruptedException { |
|  | Thread.sleep(5000); |
|  | driver.quit(); |
|  | } |
|  | /\*Test case: |
|  | go to http://the-internet.herokuapp.com/windows |
|  | click on click "Click here" |
|  |  |
|  | verify that new title is "New Window" |
|  | verify that new title is not "The Internet" |
|  |  |
|  | \*/ |
|  | @Test |
|  | public void changeTab(){ |
|  | System.out.println(driver.getWindowHandles().size()); |
|  | System.out.println(driver.getWindowHandle()); |
|  |  |
|  | driver.findElement(By.linkText("Click Here")).click(); |
|  |  |
|  | System.out.println(driver.getWindowHandles().size()); |
|  | // change to the new tab |
|  |  |
|  |  |
|  | Set<String> windowHandles = driver.getWindowHandles(); |
|  |  |
|  | System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); |
|  | for (String handle : windowHandles) { |
|  | System.out.println(handle); |
|  | // switching tabs |
|  | driver.switchTo().window(handle); |
|  |  |
|  | System.out.println(driver.getTitle()); |
|  |  |
|  | if (driver.getTitle().equals("New Window")){ |
|  | break; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | Assert.assertEquals( driver.getTitle(),"New Window"); |
|  | Assert.assertNotEquals( driver.getTitle(),"The Internet"); |
|  | } |
|  |  |
|  |  |
|  | /\* |
|  | \* go to https://www.w3schools.com/html/ |
|  | \* click ont Try it Yourself |
|  | \* verify the url of the new tab contains "default" |
|  | \* |
|  | \*/ |
|  | @Test |
|  | public void changeTab2(){ |
|  | driver.get("https://www.w3schools.com/html/"); |
|  | driver.findElement(By.linkText("Try it Yourself »")).click(); |
|  |  |
|  | // title of the second Tryit Editor v3.6 |
|  | // get all handles |
|  | // switch 1 by one |
|  | // if the title is the right, break the loop |
|  |  |
|  | for(String handle: driver.getWindowHandles()){ |
|  | driver.switchTo().window(handle); |
|  | if (driver.getTitle().equals("Tryit Editor v3.6")){ |
|  | break; |
|  | } |
|  | } |
|  | Assert.assertTrue(driver.getCurrentUrl().contains("default")); |
|  |  |
|  | System.out.println(driver.getWindowHandles().size()); |
|  | driver.close(); |
|  |  |
|  | System.out.println(driver.getWindowHandles().size()); |
|  |  |
|  | } |
|  |  |
|  | } |

## HOMEWORK

[**Web Order Login Tests Homework.pdf**](https://learn.cybertekschool.com/courses/135/modules/items/5900)

[**PRESTA SHOP PART I.pdf**](https://learn.cybertekschool.com/courses/135/modules/items/5899)

# SOFT ASSERTIONS, TEST BASE, ACTIONS, FILE DOWNLOAD

## Test base

|  |  |
| --- | --- |
|  | public abstract class TestBase { |
|  | protected WebDriver driver; |
|  | protected Actions actions; |
|  | protected SoftAssert softAssert; |
|  |  |
|  | @BeforeClass |
|  | public void setUpClass(){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | } |
|  |  |
|  | @BeforeMethod |
|  | public void setUpMethod(){ |
|  | driver = new ChromeDriver(); |
|  | driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); |
|  | actions = new Actions(driver); |
|  | softAssert = new SoftAssert(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDownMethod(){ |
|  | driver.quit(); |
|  | softAssert.assertAll(); |
|  | } |
|  | } |

## Actions double click

|  |  |
| --- | --- |
|  | public class DoubleClickTest extends TestBase { |
|  |  |
|  | @Test |
|  | public void test() { |
|  | driver.get("https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5\_ev\_ondblclick2"); |
|  | driver.switchTo().frame("iframeResult"); |
|  |  |
|  | // create actons object |
|  | Actions actions = new Actions(driver); |
|  |  |
|  | // identify the webelement on which we double click |
|  | WebElement text = driver.findElement(By.id("demo")); |
|  |  |
|  | // double click on element |
|  | // we need to add perform at the end of any action |
|  | actions.doubleClick(text).perform(); |
|  |  |
|  | Assert.assertTrue(text.getAttribute("style").contains("red")); |
|  |  |
|  | } |
|  | } |

## Actions drag and drop

|  |
| --- |
|  |
|  | public class DragAndDrop extends TestBase { |
|  | @Test |
|  | public void test(){ |
|  | driver.get("https://demos.telerik.com/kendo-ui/dragdrop/index"); |
|  |  |
|  | WebElement source = driver.findElement(By.id("draggable")); |
|  | WebElement target = driver.findElement(By.id("droptarget")); |
|  | // pass the variable in the correct order, first one is source, second one is target |
|  | actions.dragAndDrop(source, target).perform(); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void test2(){ |
|  | driver.get("https://demos.telerik.com/kendo-ui/dragdrop/index"); |
|  |  |
|  | WebElement source = driver.findElement(By.id("draggable")); |
|  | WebElement target = driver.findElement(By.id("droptarget")); |
|  |  |
|  | // write the steps for doing the drag and drop operation |
|  | // without using the dragAndDrop(); method |
|  | // when we use multiple actions we need to add build() before we perform() |
|  | // |
|  |  |
|  | actions.clickAndHold(source).moveToElement(target).release().build().perform(); |
|  |  |
|  |
|  |
|  | } |
|  |  |
|  | } |

## Actions hover

|  |  |
| --- | --- |
|  | public class HoverTest extends TestBase { |
|  |  |
|  | @Test |
|  | public void test(){ |
|  | driver.get("https://www.amazon.com/"); |
|  | WebElement signIn = driver.findElement(By.xpath("//span[.='Hello. Sign in']")); |
|  | // moving the mouse on top of the given element |
|  | actions.moveToElement(signIn).perform(); |
|  |  |
|  | driver.findElement(By.xpath("//span[.='Your Hearts']")).click(); |
|  |  |
|  | Assert.assertTrue(driver.getTitle().contains("Interesting")); |
|  | } |
|  |  |
|  | @Test |
|  | public void test2(){ |
|  | driver.get("https://www.amazon.com/"); |
|  | WebElement help = driver.findElement(By.xpath("(//a[.='Help'])[2]")); |
|  |  |
|  | actions.moveToElement(help).perform(); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |

## Verify file exists

|  |  |
| --- | --- |
|  | public class FileExistsTest { |
|  |  |
|  | @Test |
|  | public void test() { |
|  | // get the current folder |
|  | String current = System.getProperty("user.dir"); |
|  | System.out.println(current); |
|  |  |
|  | // get the user folder |
|  | String userFolder = System.getProperty("user.home"); |
|  | System.out.println(userFolder); |
|  |  |
|  | // MAC USE THIS LINE |
|  | String path = userFolder + "/Downloads/test.txt"; |
|  |  |
|  | // WINDOWS USE THIS LINE |
|  | // String path = userFolder + "\\Downloads\\test.txt"; |
|  |  |
|  | System.out.println(path); |
|  |  |
|  | // Files.exists(Paths.get() : returns true if file exists in the parameter provided |
|  |  |
|  |  |
|  | Assert.assertTrue(Files.exists(Paths.get(path))); |
|  | } |
|  | } |

## Test that extends test base

|  |
| --- |
|  |
| /\* |
|  | open chrome |
|  | go to google |
|  | search for selenium cookbook |
|  | verify title contains the search term |
|  | \*/ |
|  | public class GoogleDemo extends TestBase { |
|  |  |
|  | @Test |
|  | public void titleTest(){ |
|  | driver.get("https://google.com"); |
|  | driver.findElement(By.name("q")).sendKeys("selenium cookbook"+ Keys.ENTER); |
|  | Assert.assertTrue(driver.getTitle().contains("selenium cookbook")); |
|  | } |
|  |  |

## Soft assert 1

|  |
| --- |
|  |
|  |
|  | public class EtsyWithSoftAssert extends TestBase { |
|  | @Test |
|  | public void test() { |
|  | driver.get("https://etsy.com"); |
|  | driver.findElement(By.id("search-query")).sendKeys("gift" + Keys.ENTER); |
|  |  |
|  | softAssert.assertTrue(driver.getTitle().contains("gift"), "title did not match"); |
|  |  |
|  | Assert.assertTrue(driver.getTitle().contains("AAA")); |
|  |  |
|  |  |
|  |  |
|  | } |
|  | } |
|  |  |

## Soft assert 2

|  |  |
| --- | --- |
|  | public class SoftAssertionExample { |
|  |  |
|  | @Test |
|  | public void test1(){ |
|  | System.out.println("starting"); |
|  |  |
|  | Assert.assertTrue(false); |
|  |  |
|  | System.out.println("done"); |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void test2(){ |
|  | SoftAssert softAssert = new SoftAssert(); |
|  |  |
|  | System.out.println("starting"); |
|  |  |
|  | softAssert.assertTrue(false); |
|  |  |
|  | softAssert.assertEquals("google", "etsy"); |
|  |  |
|  | System.out.println("done"); |
|  |  |
|  | softAssert.assertAll(); |
|  |  |
|  | System.out.println("really"); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void test3(){ |
|  | SoftAssert softAssert = new SoftAssert(); |
|  |  |
|  | System.out.println("starting"); |
|  |  |
|  | softAssert.assertTrue(false); |
|  |  |
|  | Assert.assertTrue(false); |
|  |  |
|  | System.out.println("done"); |
|  |  |
|  | softAssert.assertAll(); |
|  | } |
|  | } |

## HOMEWORK

[**PRESTA SHOP PART II.pdf**](https://learn.cybertekschool.com/courses/135/modules/items/5923)

# FILE UPLOAD. WEB TABLES

## Table: select whole table, select column names, select certain row, select by coordinate, select row, table size, index

## Xpath: index, parent

|  |
| --- |
|  |
|  | public class WebTablesExample extends TestBase { |
|  | /\* |
|  | Prints the whole table |
|  | \*/ |
|  | @Test |
|  | public void printWholeTable(){ |
|  | login(); |
|  | WebElement table = driver.findElement(By.id("ctl00\_MainContent\_orderGrid")); |
|  | System.out.println(table.getText()); |
|  | } |
|  |  |
|  | public void login(){ |
|  | driver.get("http://secure.smartbearsoftware.com/samples/testcomplete12/weborders/"); |
|  | driver.findElement(By.id("ctl00\_MainContent\_username")). |
|  | sendKeys("Tester"); |
|  | driver.findElement(By.id("ctl00\_MainContent\_password")). |
|  | sendKeys("test"+ Keys.ENTER); |
|  |  |
|  | } |
|  | /\* |
|  | printing the headers of the table |
|  | \*/ |
|  | @Test |
|  | public void printAllHeaders(){ |
|  | login(); |
|  | List<WebElement> headers = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']//th")); |
|  |  |
|  | for (WebElement header:headers) { |
|  | System.out.println(header.getText()); |
|  |  |
|  | } |
|  | } |
|  | /\* |
|  | prints the number of rows |
|  |  |
|  | prints all rows === prints the whole again |
|  |  |
|  |  |
|  | prints the certain based on index |
|  | we got all rows in a list, and selected a row from that list |
|  | \*/ |
|  |  |
|  | @Test |
|  | public void printRow(){ |
|  | login(); |
|  | List<WebElement> allRows = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr")); |
|  | System.out.println("number of rows: " + allRows.size()); |
|  |  |
|  | for (WebElement header:allRows) { |
|  | System.out.println(header.getText()); |
|  |  |
|  | } |
|  |  |
|  | // print the nth |
|  | System.out.println("printing row " + 4); |
|  | System.out.println(allRows.get(3).getText()); |
|  |  |
|  | } |
|  | /\* |
|  | print the size: columns and rows in a table |
|  |  |
|  | \*/ |
|  | @Test |
|  | public void printTableSize(){ |
|  | login(); |
|  | List<WebElement> allRows = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr")); |
|  | System.out.println("number of rows: " + allRows.size()); |
|  |  |
|  | List<WebElement> allHeaders = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr[1]/th")); |
|  | System.out.println("number of cols "+allHeaders.size()); |
|  |  |
|  | } |
|  |  |
|  | /\* |
|  | print a row without using a list |
|  | \*/ |
|  | @Test |
|  | public void printRow2(){ |
|  | login(); |
|  | WebElement row = driver.findElement(By.xpath( |
|  | "//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr[5]")); |
|  | System.out.println(row.getText()); |
|  | } |
|  |  |
|  | /\* |
|  | print all the cells of a a row whose index is given in the xpath |
|  | \*/ |
|  | @Test |
|  | public void printAllCellsInOneRow(){ |
|  | login(); |
|  | List<WebElement> cells = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr[4]/td")); |
|  |  |
|  | for (WebElement header:cells) { |
|  | System.out.println(header.getText()); |
|  |  |
|  | } |
|  | } |
|  | /\* |
|  | print by coordinates : based on row and column number |
|  | \*/ |
|  | @Test |
|  | public void printByCoordinates(){ |
|  | login(); |
|  | WebElement cell = driver.findElement( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr[6]/td[11]")); |
|  | System.out.println(cell.getText()); |
|  |  |
|  | System.out.println(getCell(6,11).getText()); |
|  | } |
|  |  |
|  | public WebElement getCell(int row, int col){ |
|  | String xpath = "//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr["+row+"]/td["+col+"]"; |
|  | return driver.findElement(By.xpath(xpath)); |
|  | } |
|  |  |
|  |  |
|  | /\* |
|  | print all the values in single column |
|  | \*/ |
|  | @Test |
|  | public void printColumn(){ |
|  | login(); |
|  | List<WebElement> allNames = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr/td[2]")); |
|  |  |
|  | for (WebElement header:allNames) { |
|  | System.out.println(header.getText()); |
|  |  |
|  | } |
|  | } |
|  |  |
|  | /\* |
|  | find the select checkbox for a given name |
|  | \*/ |
|  | @Test |
|  | public void selectCheckbox(){ |
|  | login(); |
|  | WebElement checkbox = driver.findElement |
|  | (By.xpath("//td[.='Susan McLaren']/../td[1]/input")); |
|  | System.out.println(checkbox.isSelected()); |
|  | checkbox.click(); |
|  | System.out.println(checkbox.isSelected()); |
|  | } |
|  |  |
|  | /\* |
|  | verify that name 'Mark Smith' exists in the names column |
|  | verify that city 'Whitestone, British' exists in the cities column |
|  |  |
|  | \*/ |
|  | @Test |
|  | public void test(){ |
|  | login(); |
|  | int nameIndex = getColumnIndex("Name"); |
|  | System.out.println(nameIndex); |
|  |  |
|  | List<WebElement> allNames = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr/td["+nameIndex+"]")); |
|  | boolean found =false; |
|  |  |
|  | for (int i = 0; i < allNames.size(); i++) { |
|  | if (allNames.get(i).getText().equals("Mark Smith")){ |
|  | found = true; |
|  | break; |
|  | } |
|  | } |
|  | Assert.assertTrue(found); |
|  |  |
|  | found = false; |
|  | int cityIndex = getColumnIndex("City"); |
|  | List<WebElement> allCities = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']/tbody/tr/td["+cityIndex+"]")); |
|  |  |
|  | for (int i = 0; i < allCities.size(); i++) { |
|  | if (allCities.get(i).getText().equals("Whitestone, British")){ |
|  | found = true; |
|  | break; |
|  | } |
|  | } |
|  | Assert.assertTrue(found); |
|  | } |
|  |  |
|  | /\* |
|  | gets a column name as a parameters |
|  | returns the index of the column name |
|  | \*/ |
|  | public int getColumnIndex(String column){ |
|  | List<WebElement> allHeader = driver.findElements( |
|  | By.xpath("//table[@id='ctl00\_MainContent\_orderGrid']//th")); |
|  |  |
|  | for (int i = 0; i < allHeader.size(); i++) { |
|  | if (allHeader.get(i).getText().equals(column)){ |
|  | return i+1; |
|  | } |
|  | } |
|  | return 0; |
|  | } |
|  | } |
|  |  |

## HOMEWORK

* [Web Orders web table tests](https://learn.cybertekschool.com/courses/135/modules/items/5967)

* [2016 Olympics](https://learn.cybertekschool.com/courses/135/modules/items/5971)

* [Porsche-1.pdf](https://learn.cybertekschool.com/courses/135/modules/items/5973)

# EXCEPTIONS REVIEW, VERIFY ELEMENT EXISTS

## Negative verifications Verify element does NOT exist

|  |
| --- |
|  |
|  | public class verifyTextDisappearTest extends TestBase { |
|  | @Test |
|  | public void verify1() { |
|  | driver.get("https://www.w3schools.com/howto/howto\_js\_toggle\_hide\_show.asp"); |
|  |  |
|  | WebElement text = driver.findElement(By.id("myDIV")); |
|  |  |
|  | Assert.assertTrue(text.isDisplayed()); |
|  |  |
|  | WebElement button = driver.findElement(By.xpath("//button[.='Toggle Hide and Show']")); |
|  |  |
|  | button.click(); |
|  |  |
|  | Assert.assertFalse(text.isDisplayed()); |
|  |  |
|  | } |
|  |  |
|  | @Test |
|  | public void verify2() { |
|  | driver.get("https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery\_html\_remove"); |
|  |  |
|  | driver.switchTo().frame("iframeResult"); |
|  |  |
|  | WebElement text = driver.findElement(By.xpath("//body/p")); |
|  | Assert.assertTrue(text.isDisplayed()); |
|  |  |
|  | WebElement button = driver.findElement(By.xpath("//body/button")); |
|  | button.click(); |
|  |  |
|  | Assert.assertFalse(elementDisplayed(By.xpath("//body/p"))); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* takes a locator and returns if element matching this locator exists |
|  | \* |
|  | \* @param by |
|  | \* @return |
|  | \*/ |
|  |  |
|  | public boolean elementDisplayed(By by) { |
|  | // try find it |
|  | // if cant find it, return false |
|  | try { |
|  | return driver.findElement(by).isDisplayed(); |
|  | } catch (NoSuchElementException e) { |
|  | return false; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | public boolean elementDisplayedS(By by) { |
|  | // if the locator mathches the list will not be empty |
|  | // if the locator does not match anything the list will be empty |
|  | return !driver.findElements(by).isEmpty(); |
|  | } |
|  |  |
|  |  |
|  | } |

# PAGE OBJECT MODEL

## Test Cases

#### Wrong username

1. go to website [http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx (Links to an external site.)Links to an external site.](http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx)   
2. try to login with wrong username  
3. verify error message **Invalid Login or Password**.

Wrong password  
1. go to website  http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx  
2. try to login with wrong password

3. verify error message **Invalid Login or Password.**

No username  
1. go to website http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx  
2. try to login no username

3. verify error message **Invalid Login or Password**.

### No password

1. go to website http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx  
2. try to login with no password

3. verify error message **Invalid Login or Password.**

## CODE EXAMPLE

### Driver class

|  |
| --- |
|  |
|  | public class Driver { |
|  |  |
|  | private static WebDriver driver; |
|  |  |
|  | private Driver(){} |
|  |  |
|  | public static WebDriver getDriver(){ |
|  | if (driver==null){ |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  | } |
|  | return driver; |
|  | } |
|  |  |
|  | public static void closeDriver(){ |
|  | if (driver!=null){ |
|  | driver.quit(); |
|  | driver = null; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |

### Test Base using driver class

|  |  |
| --- | --- |
|  | public abstract class TestBase { |
|  | protected WebDriver driver; |
|  | protected Actions actions; |
|  | protected SoftAssert softAssert; |
|  |  |
|  |  |
|  | @BeforeMethod |
|  | public void setUpMethod(){ |
|  |  |
|  | driver = Driver.getDriver(); |
|  |  |
|  | driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); |
|  | actions = new Actions(driver); |
|  | softAssert = new SoftAssert(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDownMethod(){ |
|  | Driver.closeDriver(); |
|  | softAssert.assertAll(); |
|  | } |
|  | } |

### PrestaShopHomePage

|  |  |
| --- | --- |
|  | public class PrestashopHomePage { |
|  |  |
|  | public PrestashopHomePage(){ |
|  | PageFactory.initElements(Driver.getDriver(), this); |
|  | } |
|  |  |
|  | @FindBy (linkText = "Sign in") |
|  | public WebElement signIn; |
|  |  |
|  | public WebElement product(String productName){ |
|  | String xpath = "(//h5/a[@title='"+productName+"'])[1]"; |
|  | return Driver.getDriver().findElement(By.xpath(xpath)); |
|  | } |
|  | } |

### ListingPage

|  |  |
| --- | --- |
|  | public class ListingPage { |
|  |  |
|  | public ListingPage(){ |
|  | PageFactory.initElements(Driver.getDriver(), this); |
|  | } |
|  |  |
|  | @FindBy(xpath = "asdfasfda") |
|  | public WebElement address; |
|  |  |
|  |  |
|  | @FindBy(xpath = "asdfasfda") |
|  | public WebElement agent; |
|  |  |
|  |  |
|  | @FindBy(xpath = "asdfasfda") |
|  | public WebElement map; |
|  |  |
|  |  |
|  | } |

### HomePage

|  |
| --- |
|  |
|  | public class HomePage { |
|  |  |
|  | public HomePage(){ |
|  | PageFactory.initElements(Driver.getDriver(), this); |
|  | } |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_username") |
|  | public WebElement username; |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_password") |
|  | public WebElement password; |
|  |  |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_login\_button") |
|  | public WebElement loginButton; |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_status") |
|  | public WebElement errorMessage; |
|  |  |
|  |  |
|  | public void login(String username, String password){ |
|  | this.username.sendKeys(username); |
|  | this.password.sendKeys(password); |
|  | loginButton.click(); |
|  | } |
|  |  |
|  | public void open(){ |
|  | Driver.getDriver().get("http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx"); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |

### Tests using page object model

|  |
| --- |
|  |
|  |
|  | public class ErrorMessageValidationTest extends TestBase { |
|  | @Test |
|  | public void wrongEmailTest(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | driver.get("http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx"); |
|  |  |
|  | homePage.username.sendKeys("admin"); |
|  | homePage.password.sendKeys("test"); |
|  | homePage.loginButton.click(); |
|  |  |
|  | String error = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  |  |
|  | Assert.assertEquals(actualError, error); |
|  | } |
|  |  |
|  | @Test |
|  | public void wrongPasswordTest(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | driver.get("http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx"); |
|  | homePage.username.sendKeys("Tester"); |
|  | homePage.password.sendKeys("wrong password"); |
|  | homePage.loginButton.click(); |
|  |  |
|  | String expextedError = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  | Assert.assertEquals(actualError, expextedError); |
|  |  |
|  | } |
|  |  |
|  | @Test |
|  | public void blankUsername(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | homePage.open(); |
|  | homePage.login("", "test"); |
|  |  |
|  | String expextedError = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  | Assert.assertEquals(actualError, expextedError); |
|  | } |
|  |  |
|  |  |
|  | // this is just a method which does not do anything |
|  | // @Test |
|  | public void test(){ |
|  | ListingPage nilesLane = new ListingPage(); |
|  |  |
|  | ListingPage palmetLane = new ListingPage(); |
|  |  |
|  | PrestashopHomePage prestashopHomePage = new PrestashopHomePage(); |
|  |  |
|  | WebElement tshirt = prestashopHomePage.product("tshirt"); |
|  |  |
|  | WebElement dress = prestashopHomePage.product("dress"); |
|  |  |
|  |  |
|  | }  } |
|  |  |

## Homework

[**PRESTA SHOP PART III-1.pdf**](https://learn.cybertekschool.com/courses/135/modules/items/6140)

# PROPERTIES FILE, DRIVER CLASS

## ConfigurationReader

|  |  |
| --- | --- |
|  | public class ConfigurationReader { |
|  | private static Properties properties; |
|  |  |
|  | // read the properties file |
|  | static { |
|  | String path = "configuration.properties"; |
|  |  |
|  | try { |
|  | FileInputStream fileInputStream = new FileInputStream(path); |
|  |  |
|  | properties = new Properties(); |
|  | properties.load(fileInputStream); |
|  |  |
|  | fileInputStream.close(); |
|  |  |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | // returns the value of specific property |
|  | public static String getProperty(String property){ |
|  | return properties.getProperty(property); |
|  | } |
|  |  |
|  | } |

## Driver class with configuration reader

|  |
| --- |
|  |
|  |
|  | public class Driver { |
|  | private static WebDriver driver; |
|  |  |
|  | private Driver(){} |
|  |  |
|  | public static WebDriver getDriver(){ |
|  | if (driver==null){ |
|  | String browser = ConfigurationReader.getProperty("browser"); |
|  | switch (browser){ |
|  | case "chrome": |
|  | WebDriverManager.chromedriver().setup(); |
|  | driver = new ChromeDriver(); |
|  | break; |
|  | case "firefox": |
|  | WebDriverManager.firefoxdriver().setup(); |
|  | driver = new FirefoxDriver(); |
|  | break; |
|  | case "ie": |
|  | WebDriverManager.iedriver().setup(); |
|  | driver = new InternetExplorerDriver(); |
|  | break; |
|  |  |
|  | } |
|  | } |
|  | return driver; |
|  | } |
|  |  |
|  | public static void closeDriver(){ |
|  | if (driver!=null){ |
|  | driver.quit(); |
|  | driver = null; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |

## HomePage with configuration reader

|  |
| --- |
|  |
|  |
|  | public class HomePage { |
|  | public HomePage(){ |
|  | PageFactory.initElements(Driver.getDriver(), this); |
|  | } |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_username") |
|  | public WebElement username; |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_password") |
|  | public WebElement password; |
|  |  |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_login\_button") |
|  | public WebElement loginButton; |
|  |  |
|  | @FindBy(id="ctl00\_MainContent\_status") |
|  | public WebElement errorMessage; |
|  |  |
|  |  |
|  | public void login(String username, String password){ |
|  | this.username.sendKeys(username); |
|  | this.password.sendKeys(password); |
|  | loginButton.click(); |
|  | } |
|  |  |
|  | public void open(){ |
|  | Driver.getDriver().get(ConfigurationReader.getProperty("url")); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |

## Test Base

|  |  |
| --- | --- |
|  | public abstract class TestBase { |
|  | protected WebDriver driver; |
|  | protected Actions actions; |
|  | protected SoftAssert softAssert; |
|  |  |
|  | @BeforeMethod |
|  | public void setUpMethod(){ |
|  |  |
|  | driver = Driver.getDriver(); |
|  |  |
|  | driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); |
|  | actions = new Actions(driver); |
|  | softAssert = new SoftAssert(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDownMethod(){ |
|  | Driver.closeDriver(); |
|  | softAssert.assertAll(); |
|  | } |
|  | } |

## Properties file

|  |  |
| --- | --- |
|  | url=http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx |
|  | browser=chrome |
|  | username=Tester |
|  | password=test |

## Tests with page object and configuration reader

|  |  |
| --- | --- |
|  | public class LoginTestWithProperties extends TestBase { |
|  |  |
|  | @Test |
|  | public void readPropertyTest(){ |
|  | driver.get(ConfigurationReader.getProperty("url")); |
|  |  |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | String username = ConfigurationReader.getProperty("username"); |
|  | String password = ConfigurationReader.getProperty("password"); |
|  | homePage.login(username, password); |
|  |  |
|  |  |
|  |  |
|  | } |
|  | } |

## Tests with page object and configuration reader 2

|  |  |
| --- | --- |
|  | public class ErrorMessageValidationTest extends TestBase { |
|  |  |
|  | @Test |
|  | public void wrongEmailTest(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | driver.get(ConfigurationReader.getProperty("url")); |
|  |  |
|  | homePage.username.sendKeys("admin"); |
|  | homePage.password.sendKeys("test"); |
|  | homePage.loginButton.click(); |
|  |  |
|  | String error = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  |  |
|  | Assert.assertEquals(actualError, error); |
|  | } |
|  |  |
|  | @Test |
|  | public void wrongPasswordTest(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | driver.get("http://secure.smartbearsoftware.com/samples/TestComplete12/WebOrders/Login.aspx"); |
|  | homePage.username.sendKeys("Tester"); |
|  | homePage.password.sendKeys("wrong password"); |
|  | homePage.loginButton.click(); |
|  |  |
|  | String expextedError = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  | Assert.assertEquals(actualError, expextedError); |
|  |  |
|  | } |
|  |  |
|  | @Test |
|  | public void blankUsername(){ |
|  | HomePage homePage = new HomePage(); |
|  |  |
|  | homePage.open(); |
|  | homePage.login("", "test"); |
|  |  |
|  | String expextedError = "Invalid Login or Password."; |
|  | String actualError = homePage.errorMessage.getText(); |
|  | Assert.assertEquals(actualError, expextedError); |
|  | } |
|  |  |
|  |  |
|  | // this is just a method which does not do anything |
|  | // @Test |
|  | public void test(){ |
|  | ListingPage nilesLane = new ListingPage(); |
|  |  |
|  | ListingPage palmetLane = new ListingPage(); |
|  |  |
|  | PrestashopHomePage prestashopHomePage = new PrestashopHomePage(); |
|  |  |
|  | WebElement tshirt = prestashopHomePage.product("tshirt"); |
|  |  |
|  | WebElement dress = prestashopHomePage.product("dress"); |
|  |  |
|  |  |
|  | } |
|  | } |

## HOMEWORK

[**PRESTA SHOP PART IV.pdf**](https://learn.cybertekschool.com/courses/135/modules/items/6139)

# WAITS AND SYNCHRONIZATION

## Explicit wait

|  |  |
| --- | --- |
|  | public class ExplicitWaitExample { |
|  |  |
|  | @Test |
|  | public void test() throws InterruptedException { |
|  | Driver.getDriver().get("https://the-internet.herokuapp.com/dynamic\_controls"); |
|  |  |
|  | // click on enable |
|  | Driver.getDriver().findElement(By.xpath("//form[@id='input-example']//button")).click(); |
|  | // WebDriverWait class used to explicit waits |
|  | // just creating onject, waiting does not happen yet |
|  | WebDriverWait wait = new WebDriverWait(Driver.getDriver(), 10); |
|  | // wait.until() : this is when wait happens |
|  | // we are waiting for certain element this this xpath to be clickable |
|  | wait.until( |
|  | ExpectedConditions.elementToBeClickable( |
|  | By.xpath("//input[@type='text']"))); |
|  |  |
|  | // enter text |
|  | Driver.getDriver().findElement(By.xpath("//input[@type='text']")).sendKeys("hello world"); |
|  | }} |

## Thread.sleep, implicit wait

|  |  |
| --- | --- |
|  | public class SleepAndImplicitWaitExample { |
|  | WebDriver driver; |
|  | @BeforeMethod |
|  | public void setUp(){ |
|  | driver = Driver.getDriver(); |
|  | } |
|  | @AfterMethod |
|  | public void tearDown() throws InterruptedException { |
|  | // Driver.closeDriver(); |
|  | } |
|  |  |
|  | @Test |
|  | public void testSleep() throws InterruptedException { |
|  | driver.get("https://the-internet.herokuapp.com/dynamic\_loading/2"); |
|  | driver.findElement(By.tagName("button")).click(); |
|  | Thread.sleep(10000); |
|  | System.out.println(driver.findElement(By.id("finish")).getText()); |
|  | } |
|  | @Test |
|  | public void implicitWait(){ |
|  | // please wait when you are trying to find and element but cannot find it |
|  | driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); |
|  | driver.get("https://the-internet.herokuapp.com/dynamic\_loading/2"); |
|  | driver.findElement(By.tagName("button")).click(); |
|  | System.out.println(driver.findElement(By.id("finish")).getText()); |
|  | } |
|  | @Test |
|  | public void implicitWait2(){ |
|  | driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); |
|  |  |
|  | driver.get("https://the-internet.herokuapp.com/dynamic\_controls"); |
|  | // click on remove button |
|  | driver.findElement(By.cssSelector("form#checkbox-example>button")).click(); |
|  | // print the message |
|  | System.out.println(driver.findElement(By.id("message")).getText()); |
|  |  |
|  | // click on add button |
|  | driver.findElement(By.cssSelector("form#checkbox-example>button")).click(); |
|  | // verify checkbox is displayed |
|  | System.out.println(driver.findElement(By.id("checkbox")).isDisplayed()); |
|  | } |
|  | } |