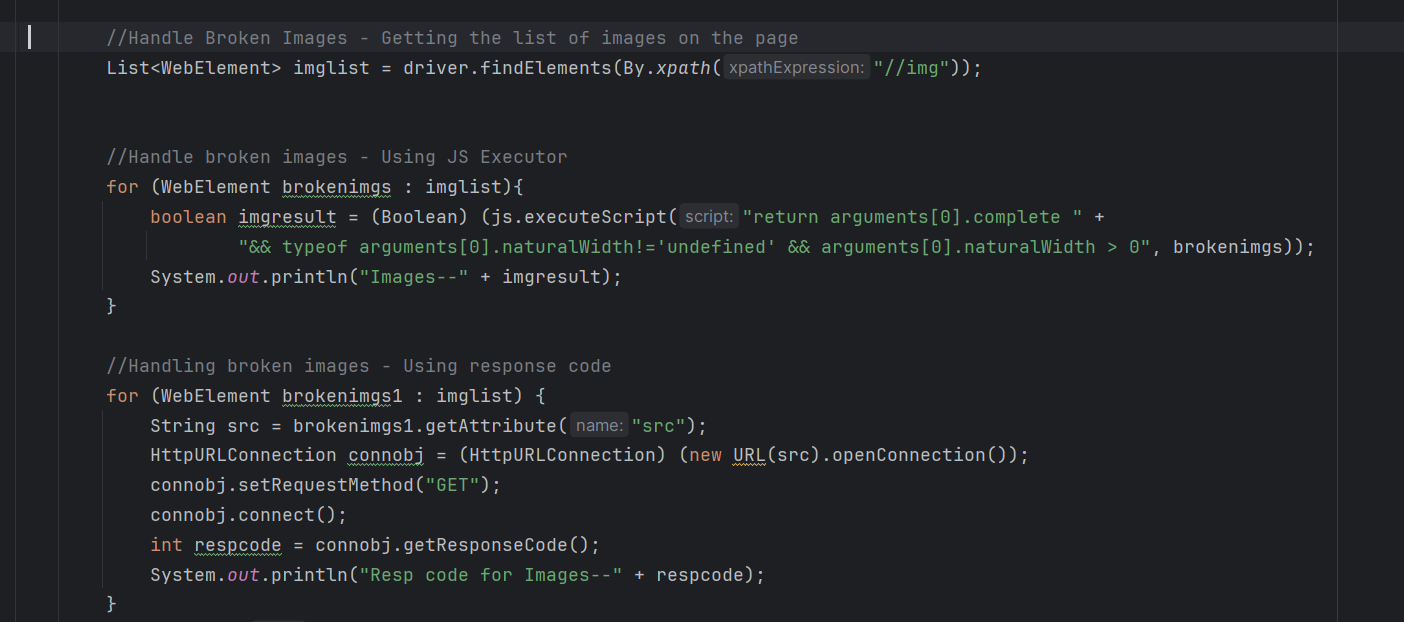
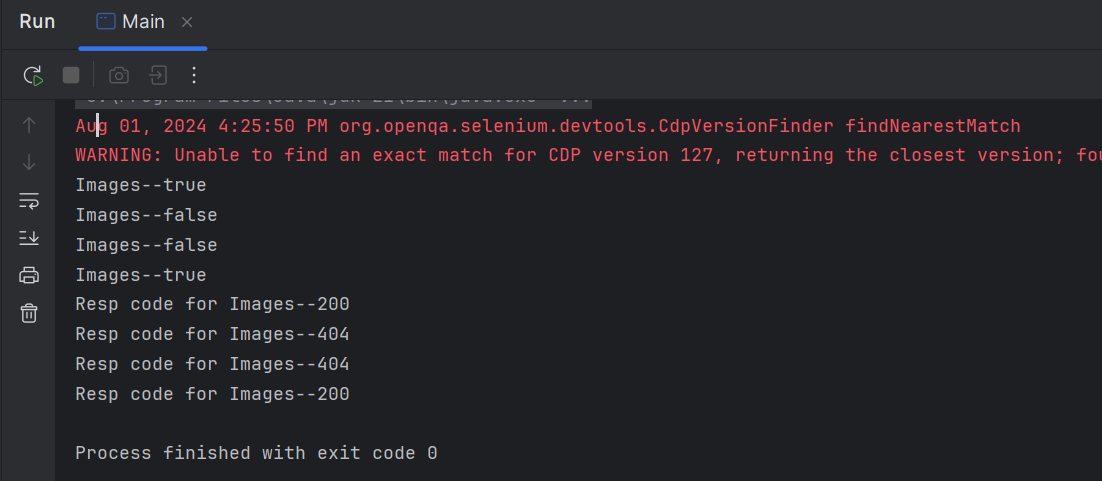
Selenium Basic Methods – Continuation…..

Handling Broken Images

Using Java Script Executor and using HTTP Response code



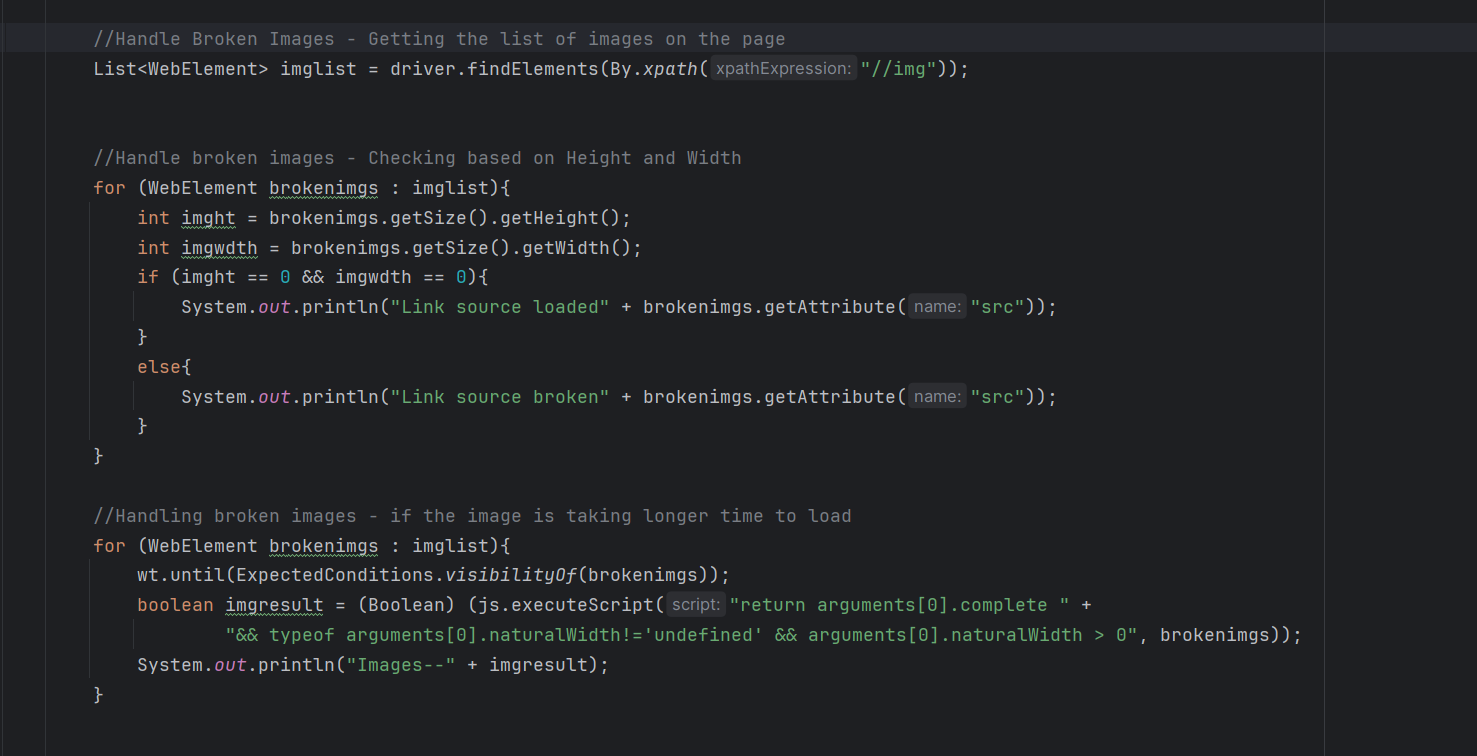


import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.support.ui.Select;  
import org.openqa.selenium.support.ui.Wait;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/broken\_images");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handle Broken Images - Getting the list of images on the page  
 List<WebElement> imglist = driver.findElements(By.*xpath*("//img"));  
  
  
 //Handle broken images - Using JS Executor  
 for (WebElement brokenimgs : imglist){  
 boolean imgresult = (Boolean) (js.executeScript("return arguments[0].complete " +  
 "&& typeof arguments[0].naturalWidth!='undefined' && arguments[0].naturalWidth > 0", brokenimgs));  
 System.*out*.println("Images--" + imgresult);  
 }  
  
 //Handling broken images - Using response code  
 for (WebElement brokenimgs1 : imglist) {  
 String src = brokenimgs1.getAttribute("src");  
 HttpURLConnection connobj = (HttpURLConnection) (new URL(src).openConnection());  
 connobj.setRequestMethod("GET");  
 connobj.connect();  
 int respcode = connobj.getResponseCode();  
 System.*out*.println("Resp code for Images--" + respcode);  
 }  
 Thread.*sleep*(8000);  
 driver.quit();  
 }  
}

Otherways to handle the broken images

Using height and width greater than zero.

Using wait.until(ExpectedConditions) to wait when images taking longer to load.

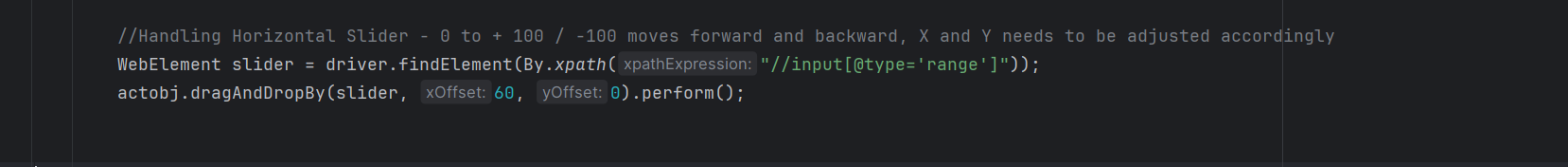


import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/broken\_images");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handle Broken Images - Getting the list of images on the page  
 List<WebElement> imglist = driver.findElements(By.*xpath*("//img"));  
  
  
 //Handle broken images - Checking based on Height and Width  
 for (WebElement brokenimgs : imglist){  
 int imght = brokenimgs.getSize().getHeight();  
 int imgwdth = brokenimgs.getSize().getWidth();  
 if (imght == 0 && imgwdth == 0){  
 System.*out*.println("Link source loaded" + brokenimgs.getAttribute("src"));  
 }  
 else{  
 System.*out*.println("Link source broken" + brokenimgs.getAttribute("src"));  
 }  
 }  
  
 //Handling broken images - if the image is taking longer time to load  
 for (WebElement brokenimgs : imglist){  
 wt.until(ExpectedConditions.*visibilityOf*(brokenimgs));  
 boolean imgresult = (Boolean) (js.executeScript("return arguments[0].complete " +  
 "&& typeof arguments[0].naturalWidth!='undefined' && arguments[0].naturalWidth > 0", brokenimgs));  
 System.*out*.println("Images--" + imgresult);  
 }  
  
  
 Thread.*sleep*(8000);  
 driver.quit();  
 }  
}

Handling Horizontal Slider

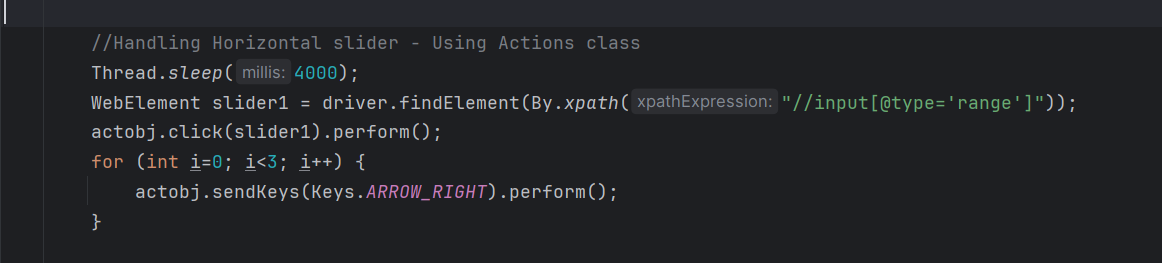
Using Actions class





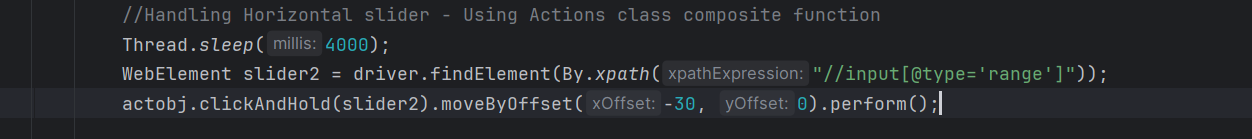
import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/horizontal\_slider");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling Horizontal Slider - 0 to + 100 / -100 moves forward and backward, X and Y needs to be adjusted accordingly  
 WebElement slider = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.dragAndDropBy(slider, 60, 0).perform();  
  
 Thread.*sleep*(4000);  
 driver.quit();  
 }  
}

Using Actions class – Keys class



import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/horizontal\_slider");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling Horizontal Slider - 0 to + 100 / -100 moves forward and backward, X and Y needs to be adjusted accordingly  
 WebElement slider = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.dragAndDropBy(slider, 60, 0).perform();  
  
 driver.navigate().refresh();  
  
 //Handling Horizontal slider - Using Actions class  
 Thread.*sleep*(4000);  
 WebElement slider1 = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.click(slider1).perform();  
 for (int i=0; i<3; i++) {  
 actobj.sendKeys(Keys.*ARROW\_RIGHT*).perform();  
 }  
  
 Thread.*sleep*(4000);  
 driver.quit();  
 }  
}

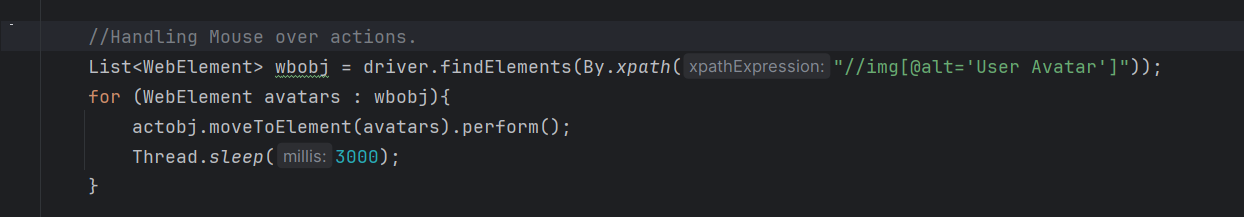
Using Actions Composite function



import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/horizontal\_slider");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling Horizontal Slider - 0 to + 100 / -100 moves forward and backward, X and Y needs to be adjusted accordingly  
 WebElement slider = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.dragAndDropBy(slider, 60, 0).perform();  
  
 driver.navigate().refresh();  
  
 //Handling Horizontal slider - Using Actions class  
 Thread.*sleep*(4000);  
 WebElement slider1 = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.click(slider1).perform();  
 for (int i=0; i<3; i++) {  
 actobj.sendKeys(Keys.*ARROW\_RIGHT*).perform();  
 }  
  
  
 driver.navigate().refresh();  
 //Handling Horizontal slider - Using Actions class composite function  
 Thread.*sleep*(4000);  
 WebElement slider2 = driver.findElement(By.*xpath*("//input[@type='range']"));  
 actobj.clickAndHold(slider2).moveByOffset(-30, 0).perform();  
  
 Thread.*sleep*(4000);  
 driver.quit();  
 }  
}

Handling Mouse Over for information



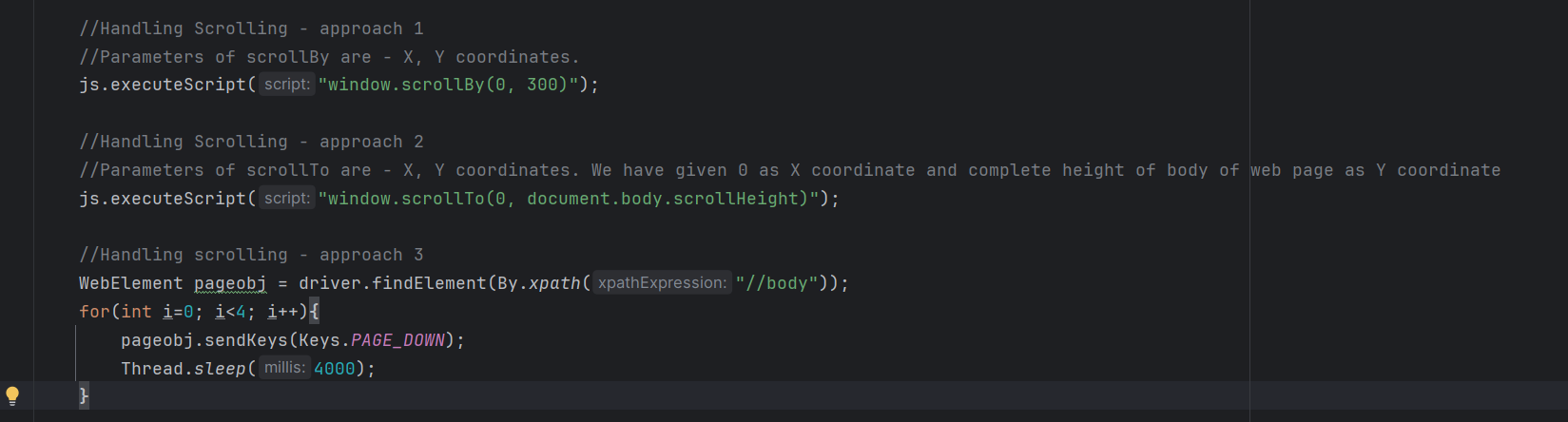


import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/hovers");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling Mouse over actions.  
 List<WebElement> wbobj = driver.findElements(By.*xpath*("//img[@alt='User Avatar']"));  
 for (WebElement avatars : wbobj){  
 actobj.moveToElement(avatars).perform();  
 Thread.*sleep*(3000);  
 }  
  
  
 Thread.*sleep*(4000);  
 driver.quit();  
 }  
}

Handling Scrolling the Web page

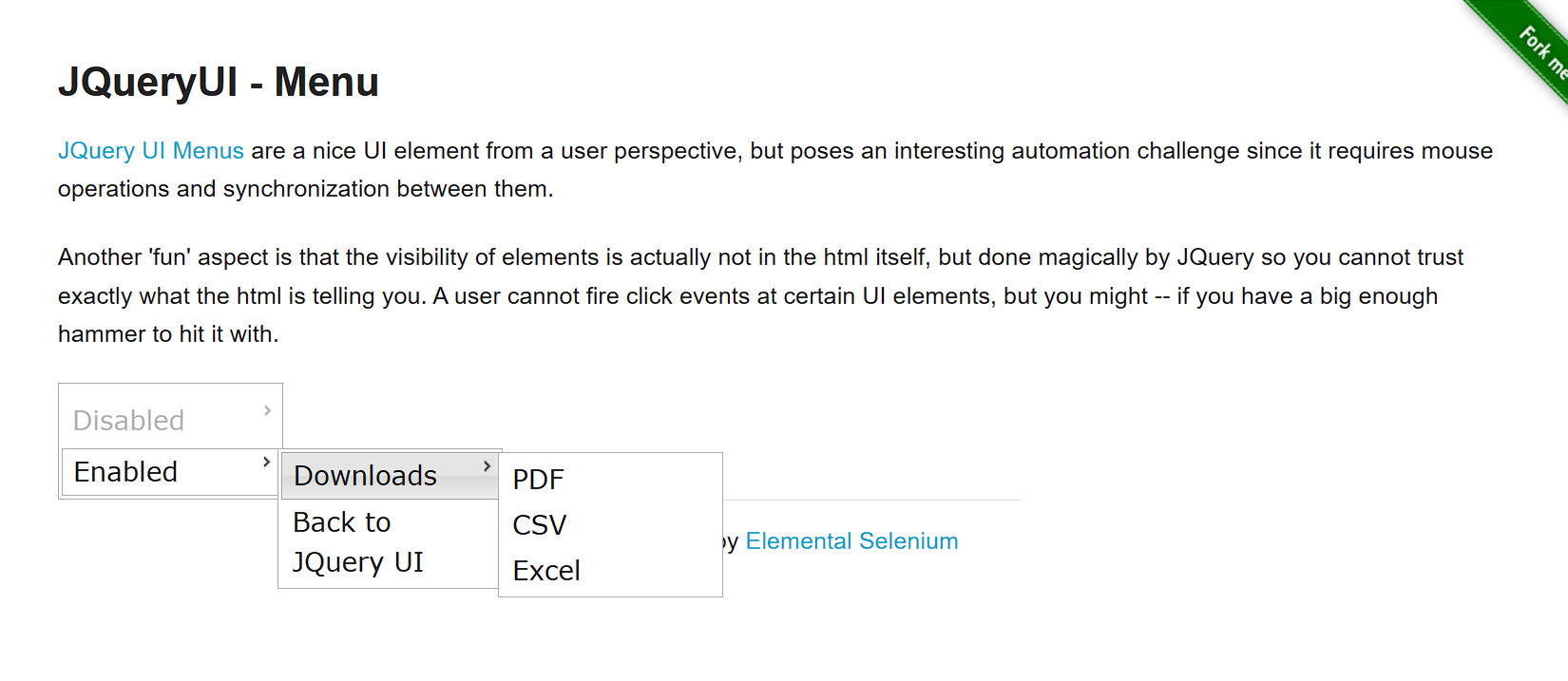
It can be done using below ways,

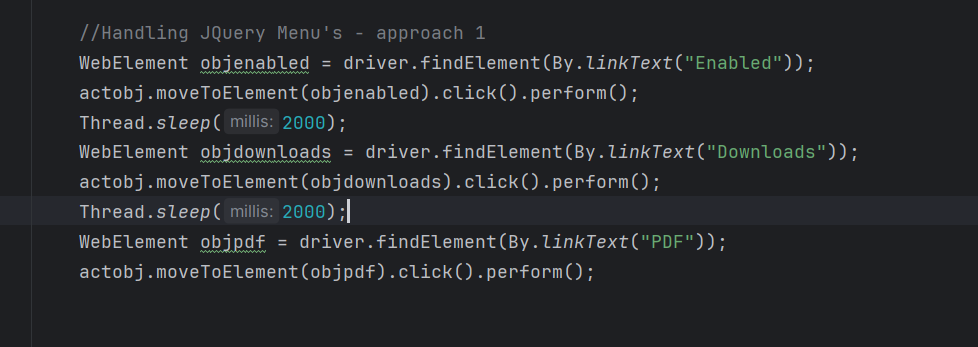
1. Using java script function - window.scrollby()
2. Using java script function – window.scrollto()
3. Using Keys.Pagedown()



import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/infinite\_scroll");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling Scrolling - approach 1  
 //Parameters of scrollBy are - X, Y coordinates.  
 js.executeScript("window.scrollBy(0, 300)");  
  
 //Handling Scrolling - approach 2  
 //Parameters of scrollTo are - X, Y coordinates. We have given 0 as X coordinate and complete height of body of web page as Y coordinate  
 js.executeScript("window.scrollTo(0, document.body.scrollHeight)");  
  
 //Handling scrolling - approach 3  
 WebElement pageobj = driver.findElement(By.*xpath*("//body"));  
 for(int i=0; i<4; i++){  
 pageobj.sendKeys(Keys.*PAGE\_DOWN*);  
 Thread.*sleep*(4000);  
 }  
  
Thread.*sleep*(7000);  
 driver.quit();  
 }  
}

Handling Jquery UI Menu

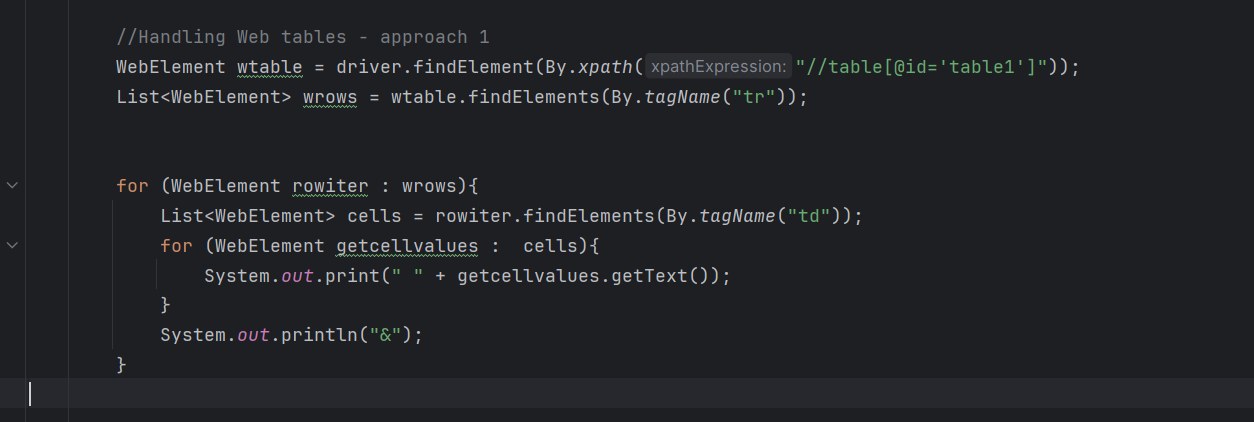




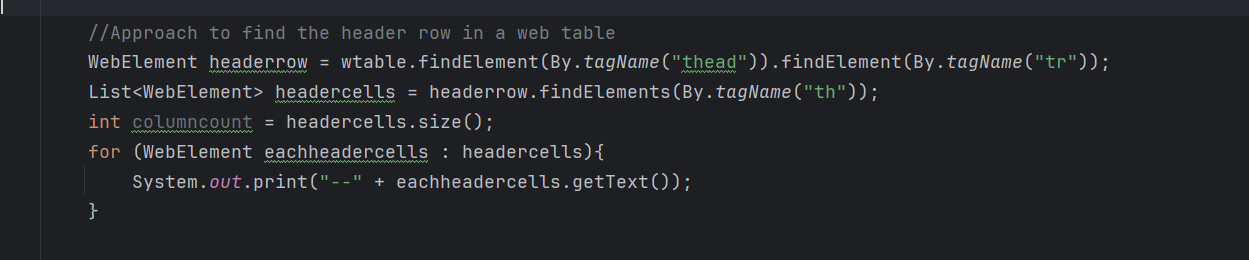
import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/jqueryui/menu");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Handling JQuery Menu's - approach 1  
 WebElement objenabled = driver.findElement(By.*linkText*("Enabled"));  
 actobj.moveToElement(objenabled).click().perform();  
 Thread.*sleep*(2000);  
 WebElement objdownloads = driver.findElement(By.*linkText*("Downloads"));  
 actobj.moveToElement(objdownloads).click().perform();  
 Thread.*sleep*(2000);  
 WebElement objpdf = driver.findElement(By.*linkText*("PDF"));  
 actobj.moveToElement(objpdf).click().perform();  
  
  
  
 Thread.*sleep*(7000);  
 driver.quit();  
 }  
}

Handling Web Tables

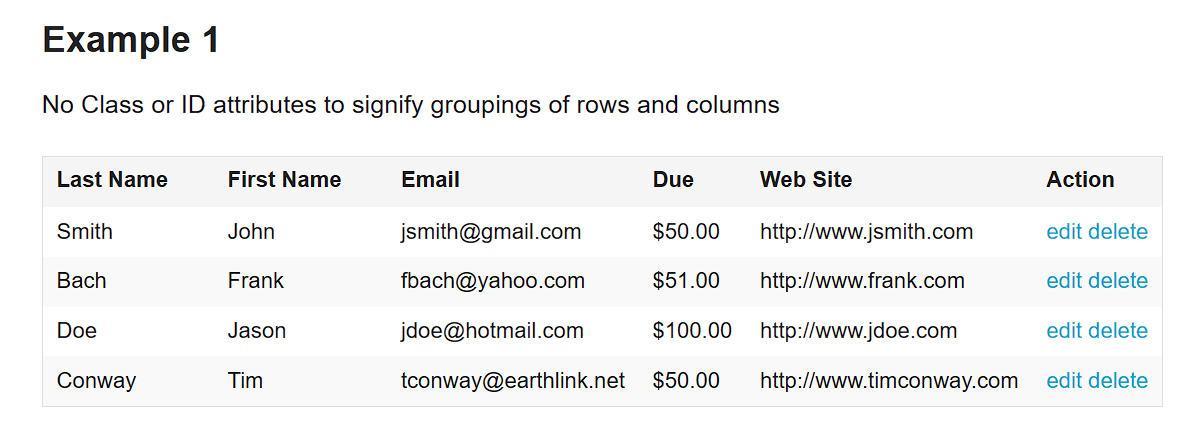
Identifying a web table using table unique property and then extracting the TD and TR using By.tag name method.

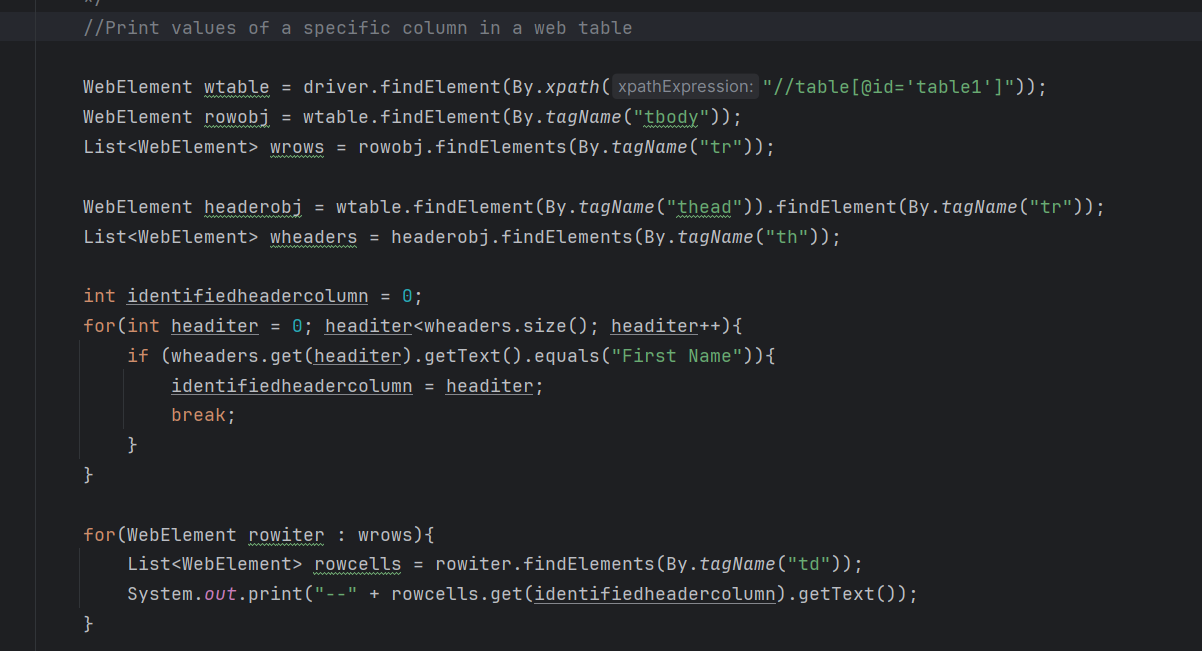


Identify the header column alone based on the already identified web table.

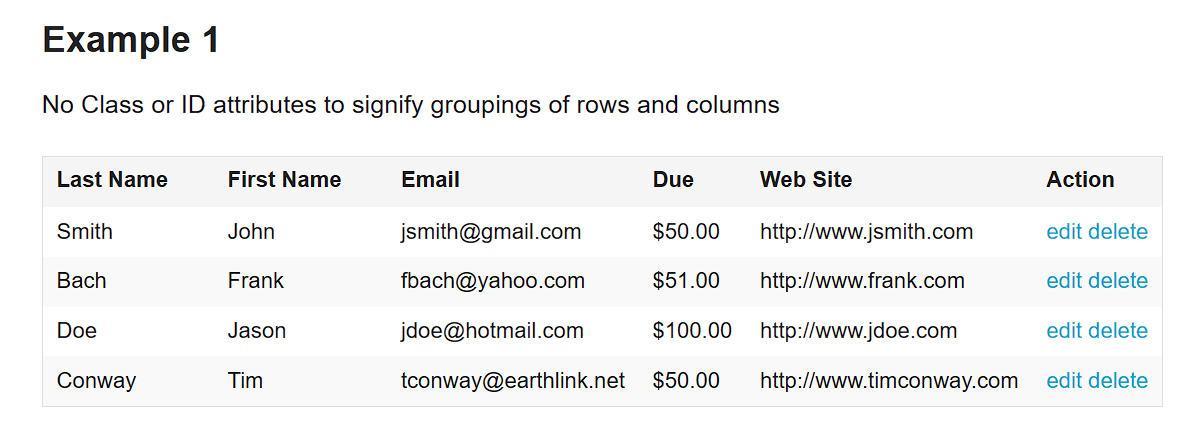


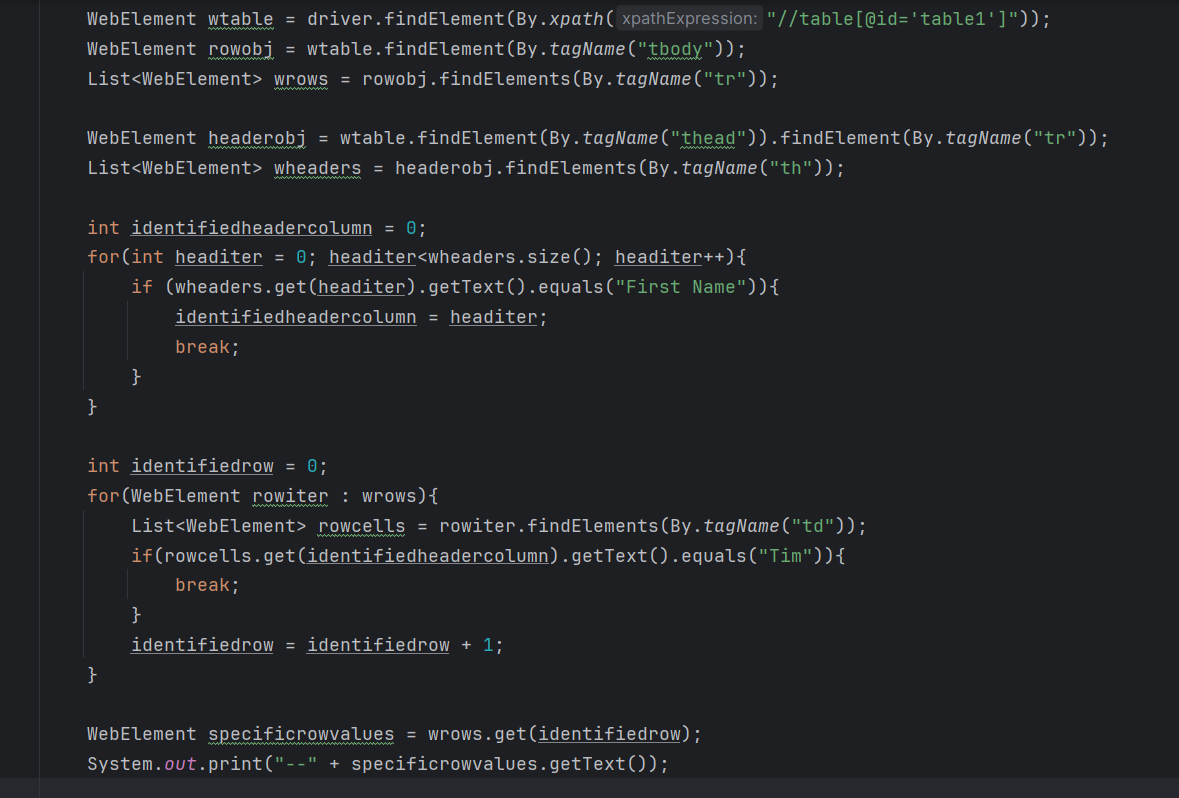
Print the values of specific column in a table. – For this we have to first identify the index of the required column.





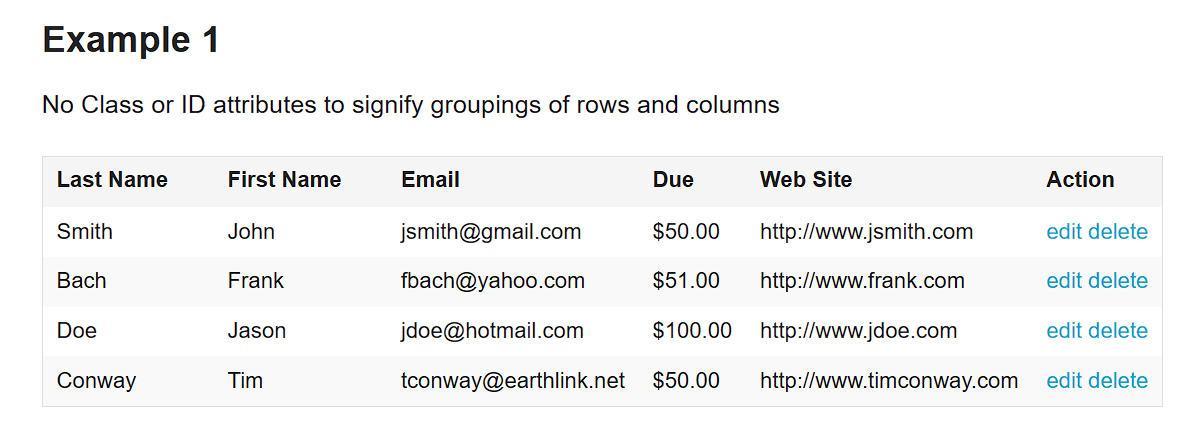
Print values of specific row in a web table

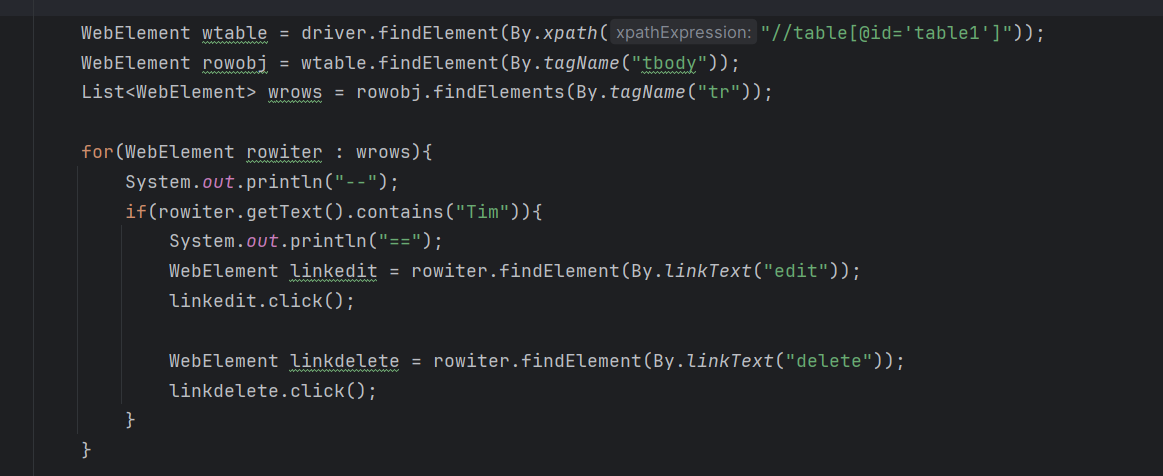




import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/tables");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
   
 //Handling Web tables - approach to find the web table and td and tr  
 WebElement wtable = driver.findElement(By.*xpath*("//table[@id='table1']"));  
 List<WebElement> wrows = wtable.findElements(By.*tagName*("tr"));  
 int rowscount = wrows.size();  
  
 //Approach to find the header row in a web table  
 WebElement headerrow = wtable.findElement(By.*tagName*("thead")).findElement(By.*tagName*("tr"));  
 List<WebElement> headercells = headerrow.findElements(By.*tagName*("th"));  
 int columncount = headercells.size();  
 for (WebElement eachheadercells : headercells){  
 System.*out*.print("--" + eachheadercells.getText());  
 }  
  
 //approach to find the row cells and print the values of the cell one by one  
 for (WebElement rowiter : wrows){  
 List<WebElement> cells = rowiter.findElements(By.*tagName*("td"));  
 for (WebElement getcellvalues : cells){  
 System.*out*.print("--" + getcellvalues.getText());  
 }  
 System.*out*.println(" ");  
 }  
  
 //Print values of a specific column in a web table  
  
 WebElement wtable = driver.findElement(By.*xpath*("//table[@id='table1']"));  
 WebElement rowobj = wtable.findElement(By.*tagName*("tbody"));  
 List<WebElement> wrows = rowobj.findElements(By.*tagName*("tr"));  
  
 WebElement headerobj = wtable.findElement(By.*tagName*("thead")).findElement(By.*tagName*("tr"));  
 List<WebElement> wheaders = headerobj.findElements(By.*tagName*("th"));  
  
 int identifiedheadercolumn = 0;  
 for(int headiter = 0; headiter<wheaders.size(); headiter++){  
 if (wheaders.get(headiter).getText().equals("First Name")){  
 identifiedheadercolumn = headiter;  
 break;  
 }  
 }  
  
 for(WebElement rowiter : wrows){  
 List<WebElement> rowcells = rowiter.findElements(By.*tagName*("td"));  
 System.*out*.print("--" + rowcells.get(identifiedheadercolumn).getText());  
 }  
  
  
 //Print values of a specific row in a web table  
  
 WebElement wtable = driver.findElement(By.*xpath*("//table[@id='table1']"));  
 WebElement rowobj = wtable.findElement(By.*tagName*("tbody"));  
 List<WebElement> wrows = rowobj.findElements(By.*tagName*("tr"));  
  
 WebElement headerobj = wtable.findElement(By.*tagName*("thead")).findElement(By.*tagName*("tr"));  
 List<WebElement> wheaders = headerobj.findElements(By.*tagName*("th"));  
  
 int identifiedheadercolumn = 0;  
 for(int headiter = 0; headiter<wheaders.size(); headiter++){  
 if (wheaders.get(headiter).getText().equals("First Name")){  
 identifiedheadercolumn = headiter;  
 break;  
 }  
 }  
  
 int identifiedrow = 0;  
 for(WebElement rowiter : wrows){  
 List<WebElement> rowcells = rowiter.findElements(By.*tagName*("td"));  
 if(rowcells.get(identifiedheadercolumn).getText().equals("Tim")){  
 break;  
 }  
 identifiedrow = identifiedrow + 1;  
 }  
  
 WebElement specificrowvalues = wrows.get(identifiedrow);  
 System.*out*.print("--" + specificrowvalues.getText());  
  
 Thread.*sleep*(2000);  
 driver.quit();  
 }  
}

Perform an action on a web element which is present inside an cell based on a condition





Perform an action on a web element which is present inside an cell based on a condition



import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofMillis*(6000));  
  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://the-internet.herokuapp.com/tables");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Print values of a specific row in a web table  
  
 WebElement wtable = driver.findElement(By.*xpath*("//table[@id='table1']"));  
 WebElement rowobj = wtable.findElement(By.*tagName*("tbody"));  
 List<WebElement> wrows = rowobj.findElements(By.*tagName*("tr"));  
  
 for(WebElement rowiter : wrows){  
 System.*out*.println("--");  
 if(rowiter!=null){  
 System.*out*.println("==");  
 WebElement linkedit = rowiter.findElement(By.*linkText*("edit"));  
 linkedit.click();  
  
 WebElement linkdelete = rowiter.findElement(By.*linkText*("delete"));  
 linkdelete.click();  
 }  
 }  
  
 Thread.*sleep*(5000);  
 driver.quit();  
 }  
}

Handling File Upload

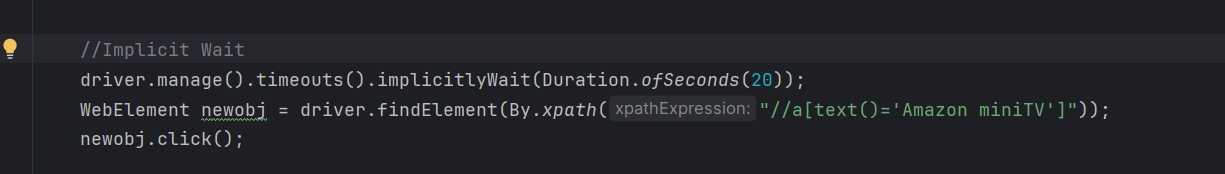
AutoIT or the Robot class is used to handle the file upload.

Waits

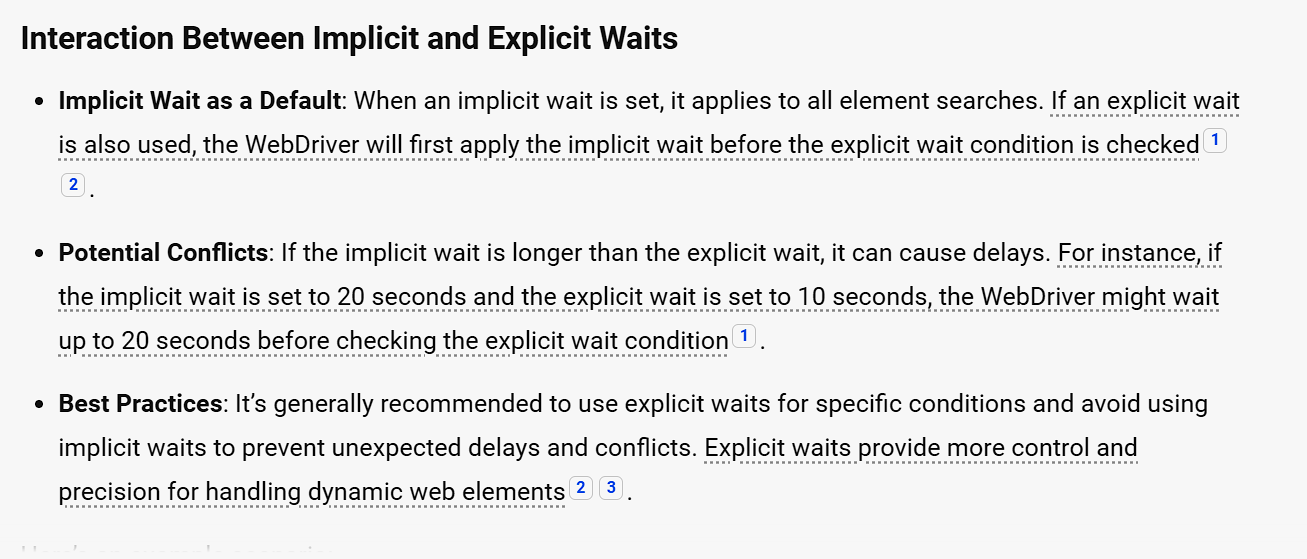
1. Implicit Wait
2. Explicit Wait
3. Fluent Wait
4. Page Load timeout
5. Script timeout

Implicit Wait

Selenium would wait for 20 seconds to identify the element based on the given locator and then only it will throw exception

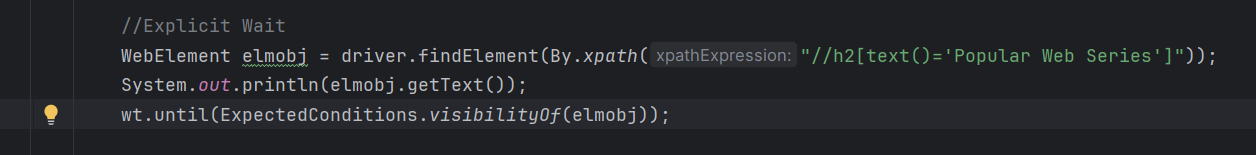


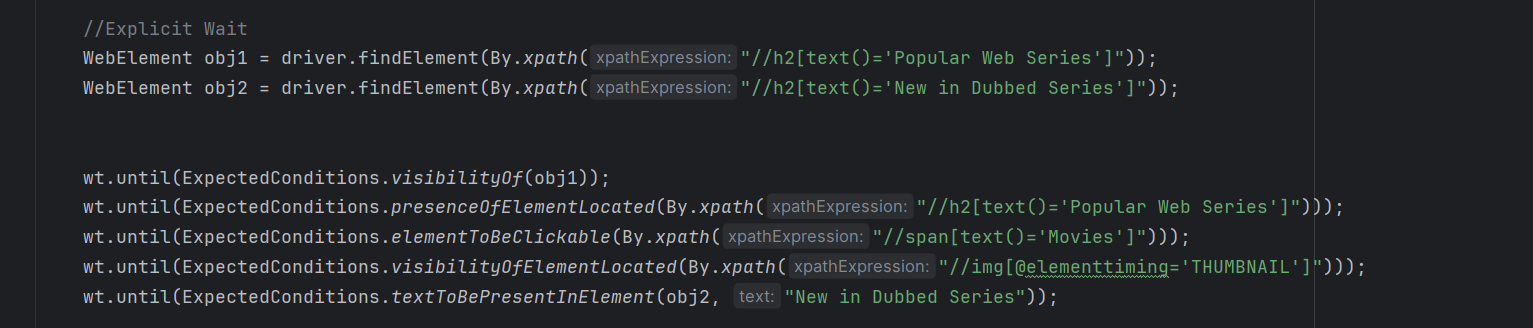
Explicit Wait



It is better to use Explicit wait alone in the Framework.

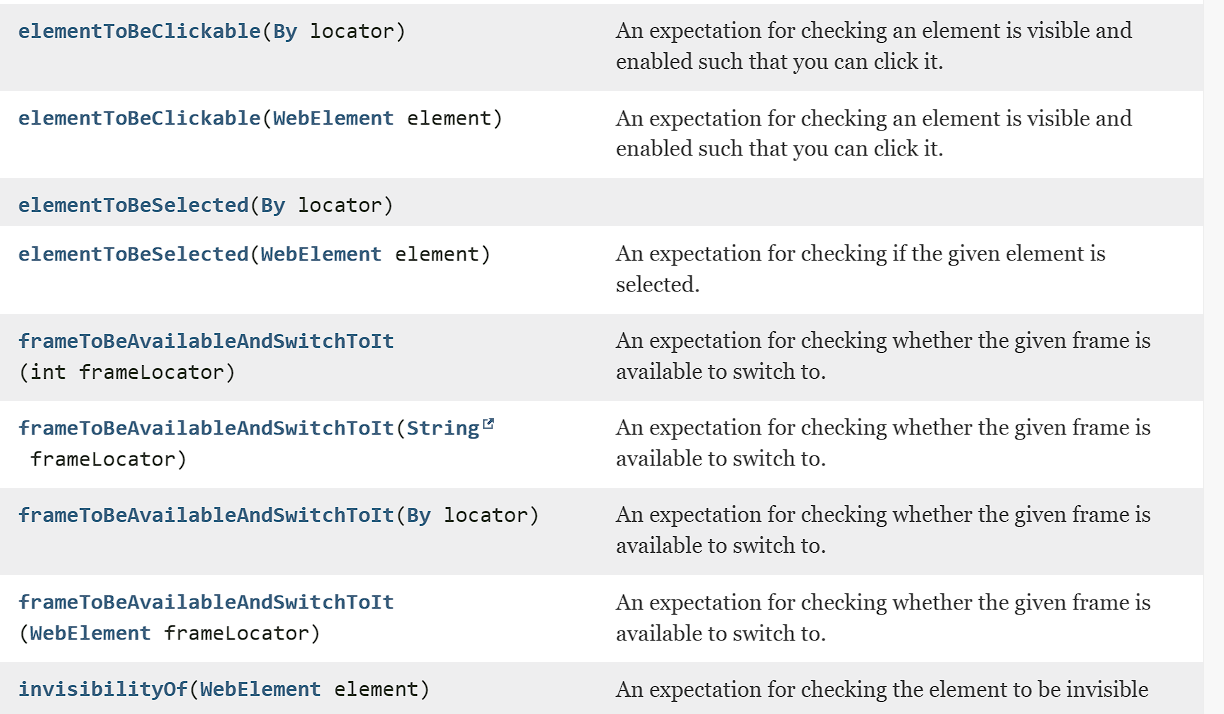
Lets see the inbuilt methods of Explicit wait

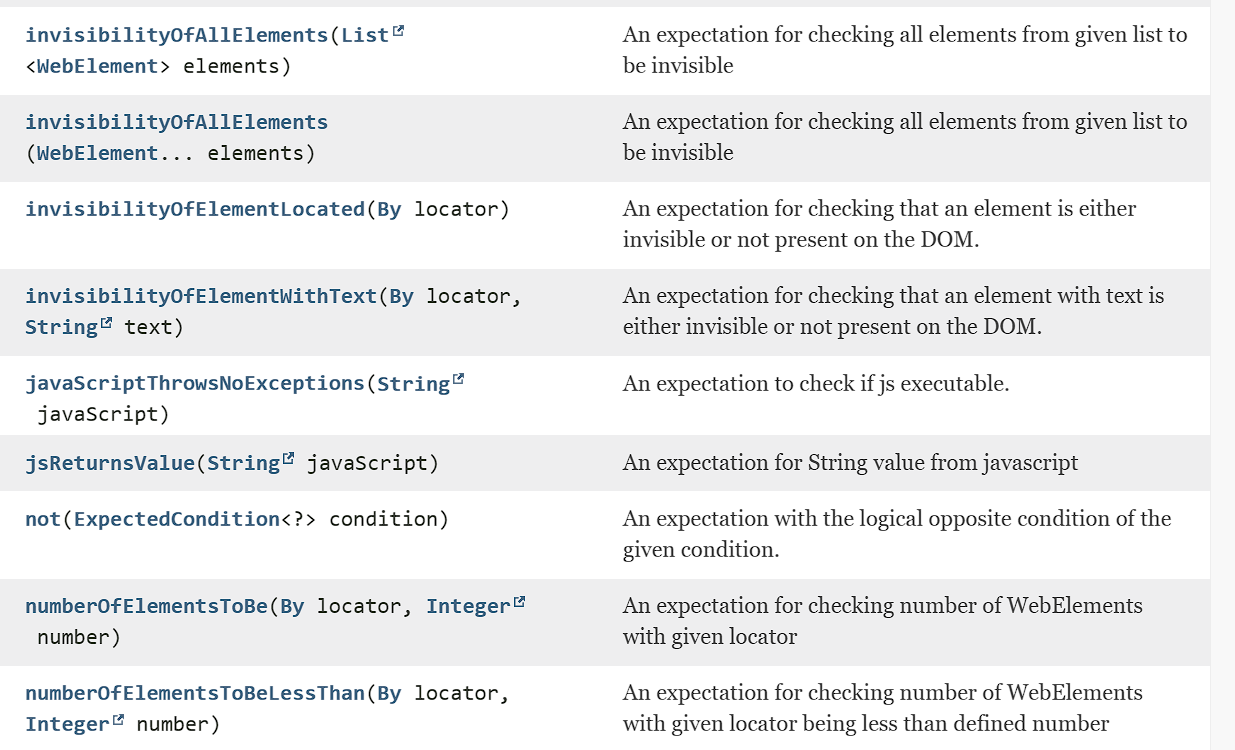


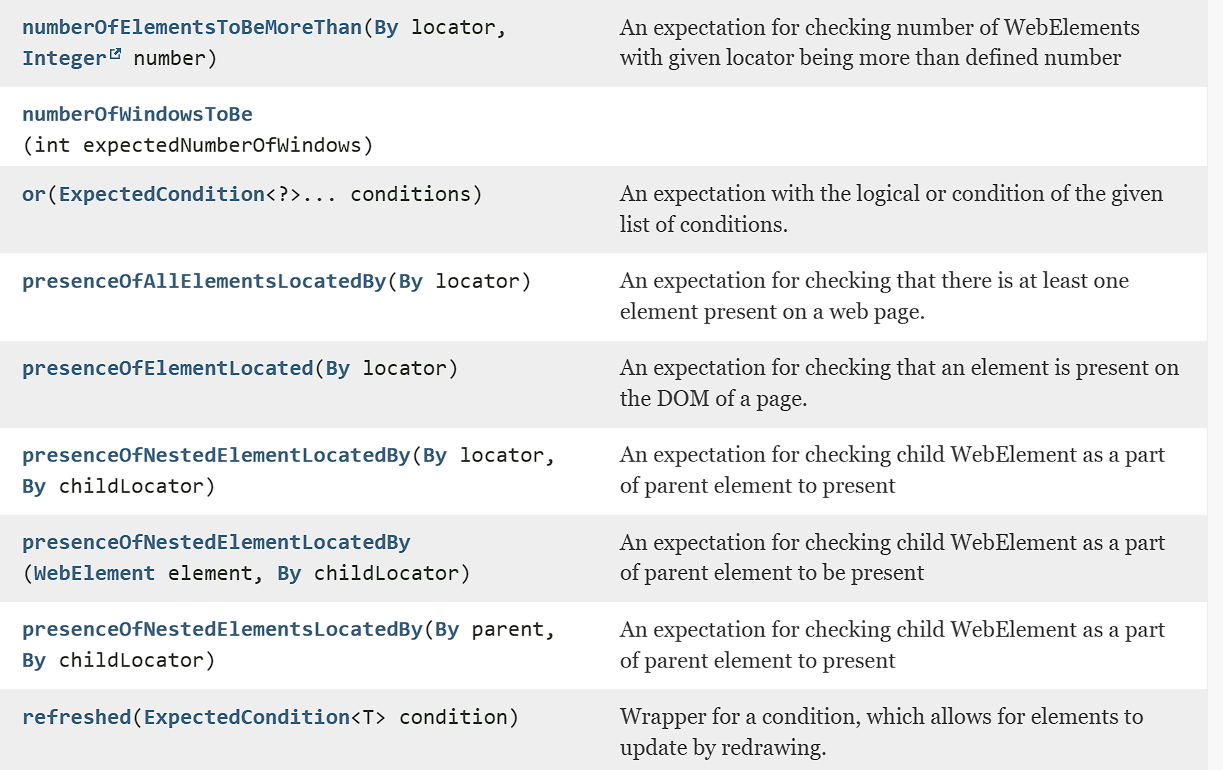


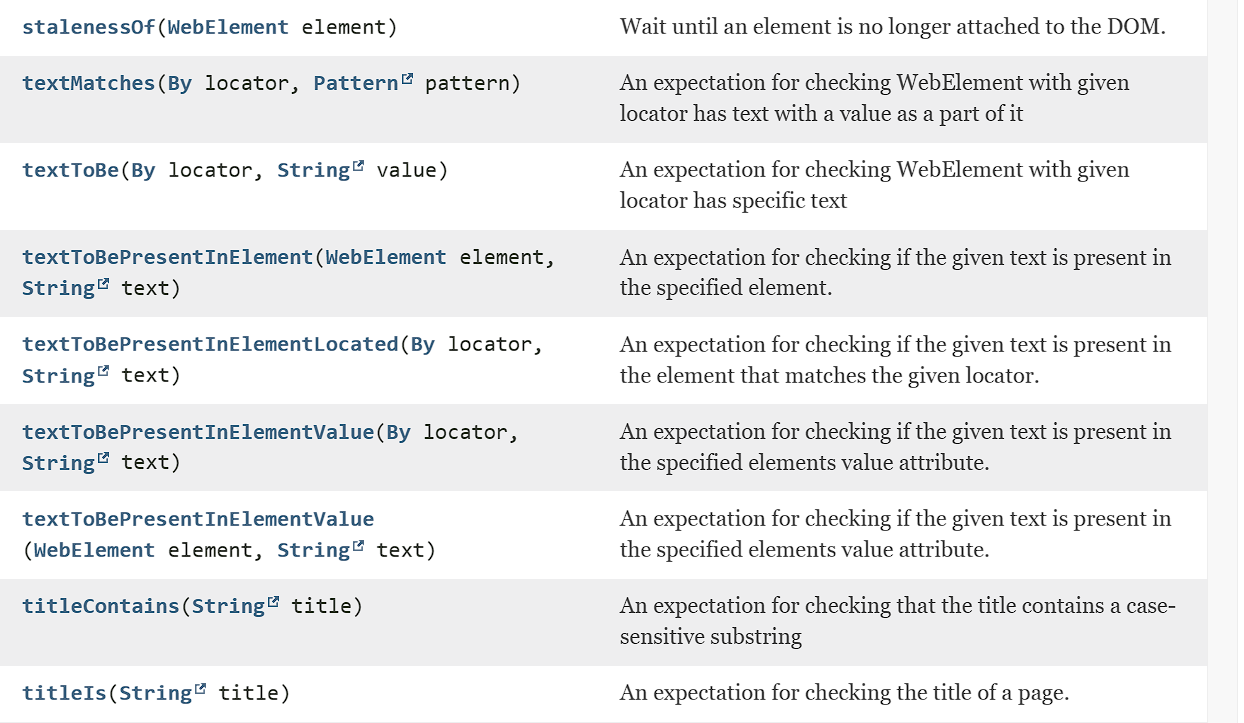
import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.IOException;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofSeconds*(20));  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://www.amazon.in/");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //Implicit Wait  
 //driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));  
 WebElement newobj = driver.findElement(By.*xpath*("//a[text()='Amazon miniTV']"));  
 newobj.click();  
  
 //Explicit Wait  
 WebElement obj1 = driver.findElement(By.*xpath*("//h2[text()='Popular Web Series']"));  
 WebElement obj2 = driver.findElement(By.*xpath*("//h2[text()='New in Dubbed Series']"));  
  
  
 wt.until(ExpectedConditions.*visibilityOf*(obj1));  
 wt.until(ExpectedConditions.*presenceOfElementLocated*(By.*xpath*("//h2[text()='Popular Web Series']")));  
 wt.until(ExpectedConditions.*elementToBeClickable*(By.*xpath*("//span[text()='Movies']")));  
 wt.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//img[@elementtiming='THUMBNAIL']")));  
 wt.until(ExpectedConditions.*textToBePresentInElement*(obj2, "New in Dubbed Series"));  
  
  
  
  
  
  
  
  
  
 Thread.*sleep*(5000);  
 driver.quit();  
 }  
}

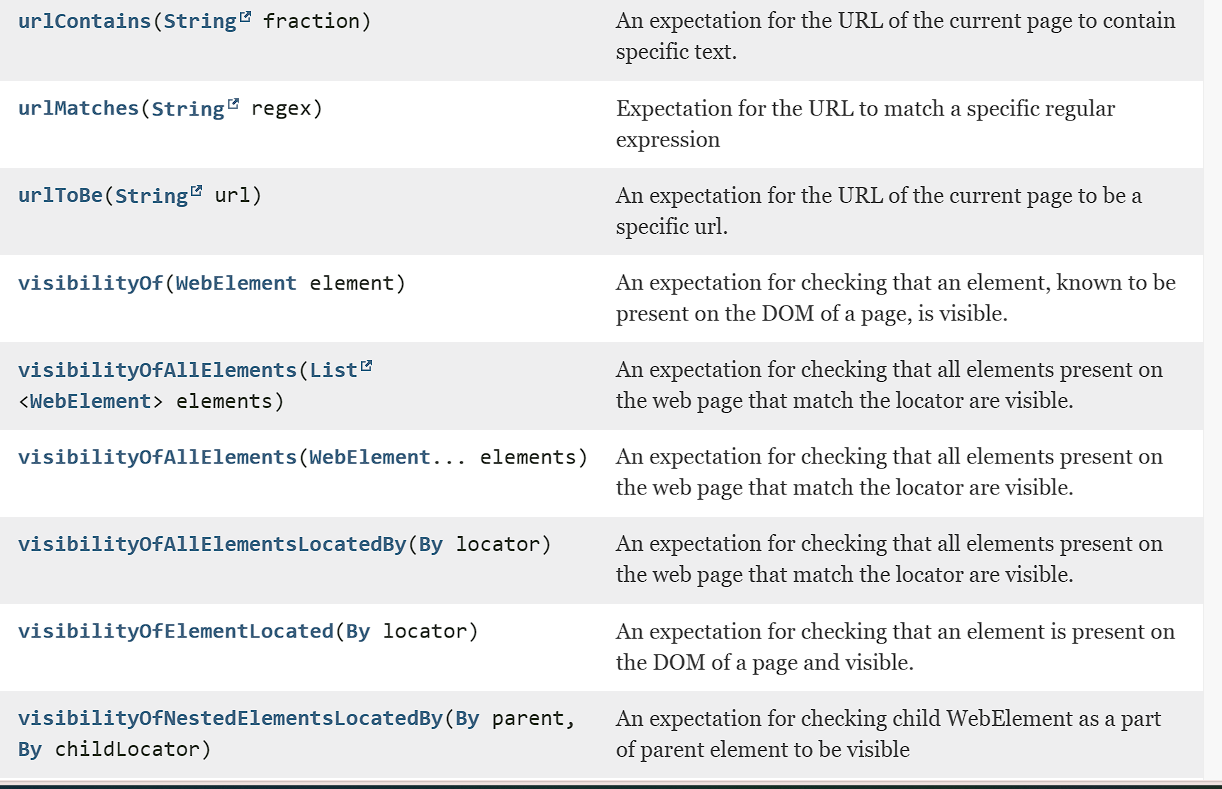


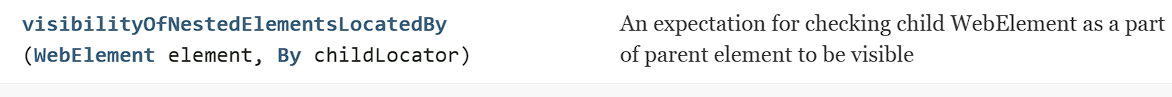




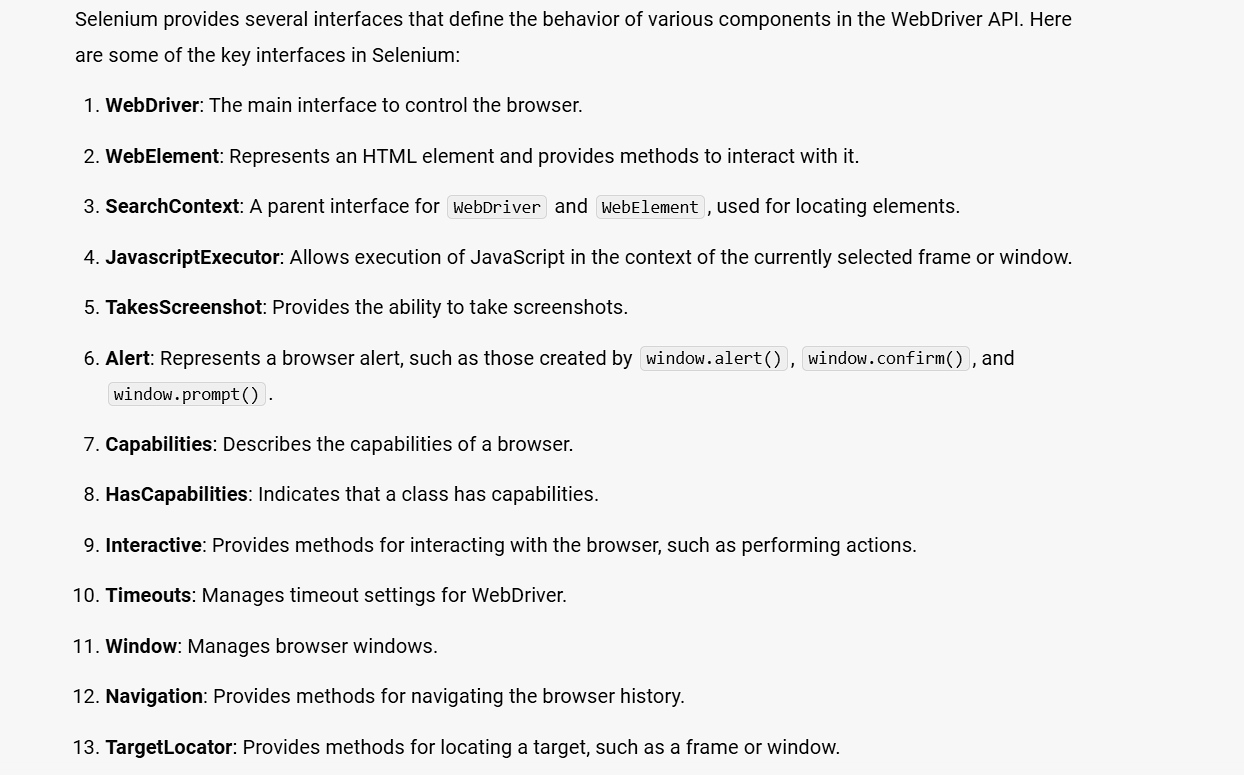




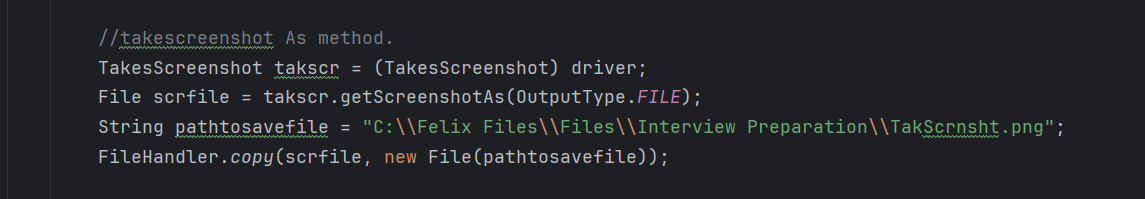




Interfaces in Selenium

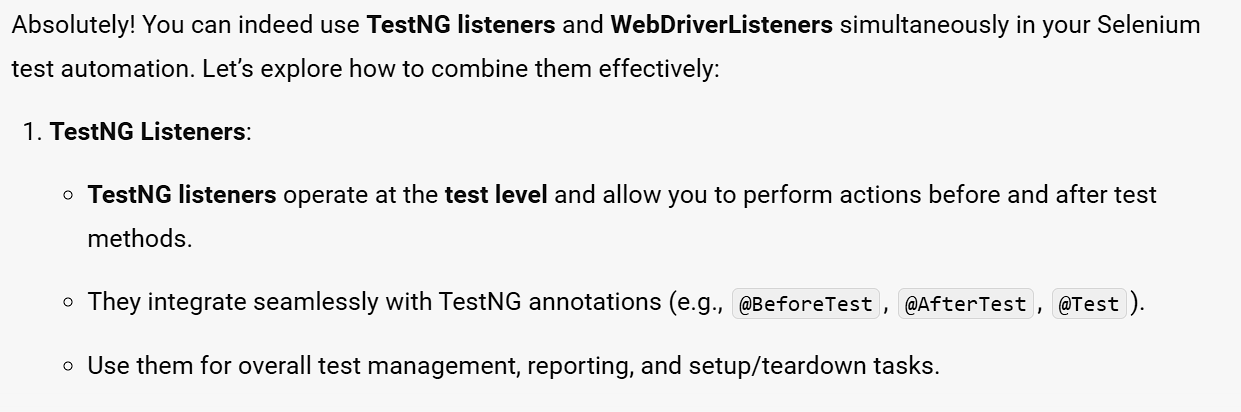


Taking screenshot in Selenium



import io.opentelemetry.exporter.logging.SystemOutLogRecordExporter;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.edge.EdgeDriver;  
import org.openqa.selenium.edge.EdgeOptions;  
import org.openqa.selenium.interactions.Actions;  
import org.openqa.selenium.io.FileHandler;  
import org.openqa.selenium.remote.tracing.opentelemetry.SeleniumSpanExporter;  
import org.openqa.selenium.support.ui.\*;  
  
import java.io.File;  
import java.io.IOException;  
  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.time.Duration;  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
import java.util.concurrent.TimeUnit;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException, IOException {  
 // Set the Path to Edge Driver.exe  
 System.*setProperty*("webdriver.edge.driver",  
 "C:\\Users\\axfel\\SeleniumANDtestNG\\Drivers\\msedgedriver.exe");  
  
 // Initialize the Drivers  
  
 WebDriver driver = new EdgeDriver();  
 Actions actobj = new Actions(driver);  
 JavascriptExecutor js = (JavascriptExecutor) driver;  
 WebDriverWait wt = new WebDriverWait(driver, Duration.*ofSeconds*(20));  
  
  
 //.get() = Opens the specified URL in the Browser type Initialized.  
 driver.get("https://www.amazon.in/");  
 Thread.*sleep*(3000);  
 driver.manage().window().maximize();  
  
 //takescreenshot As method.  
 TakesScreenshot takscr = (TakesScreenshot) driver;  
 File scrfile = takscr.getScreenshotAs(OutputType.*FILE*);  
 String pathtosavefile = "C:\\Felix Files\\Files\\Interview Preparation\\TakScrnsht.png";  
 FileHandler.*copy*(scrfile, new File(pathtosavefile));  
 //Thread.sleep(5000);  
 driver.quit();  
 }  
}

**Listeners in Selenium**

****

****

****

Implementing WebdriverListeners.

Create a separate class with all the required WebdriverListeners implemented as per your project needs.

As of now, I don’t have any requirement and hence just given a print statement within the listener methods implementation.

****

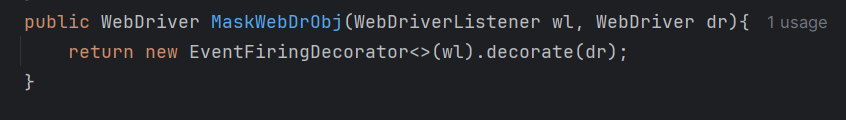
Create a object for the above class where we have added implementation for the methods available in the WebDriverListener Interface.



wrap the Web Driver object into Event Firing Decorator

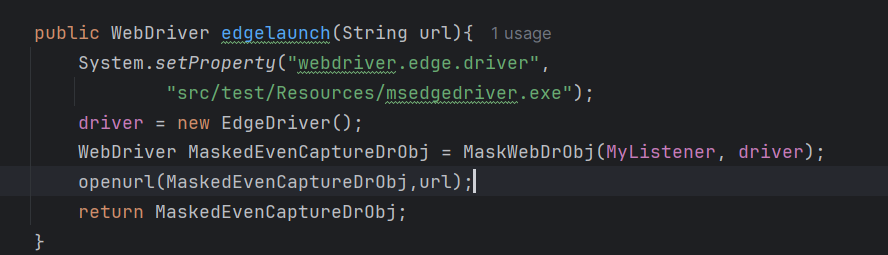
The below code should be added once the web driver object is declared as firefox/chrome/edge browser object.





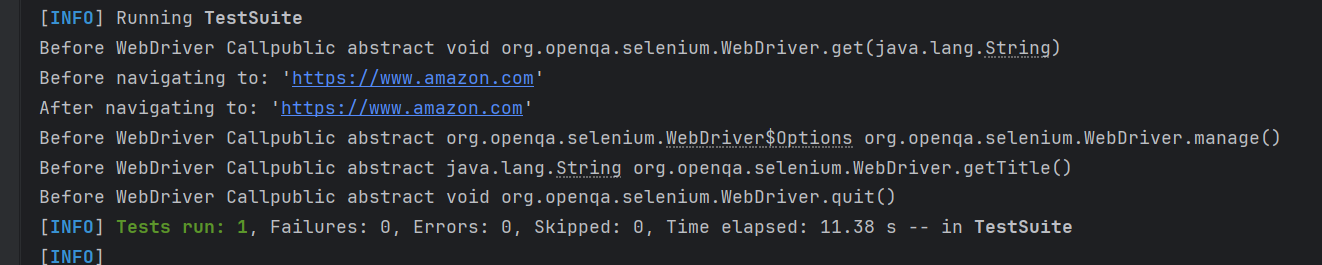
This will be the exact code to implement the web driver listener during the web driver declaration area.

Here we can see the webdriver object is wrapped by the EventFiringDecorator. The usual webdriver methods can be called using this MaskedEvenCaptureDrObj.

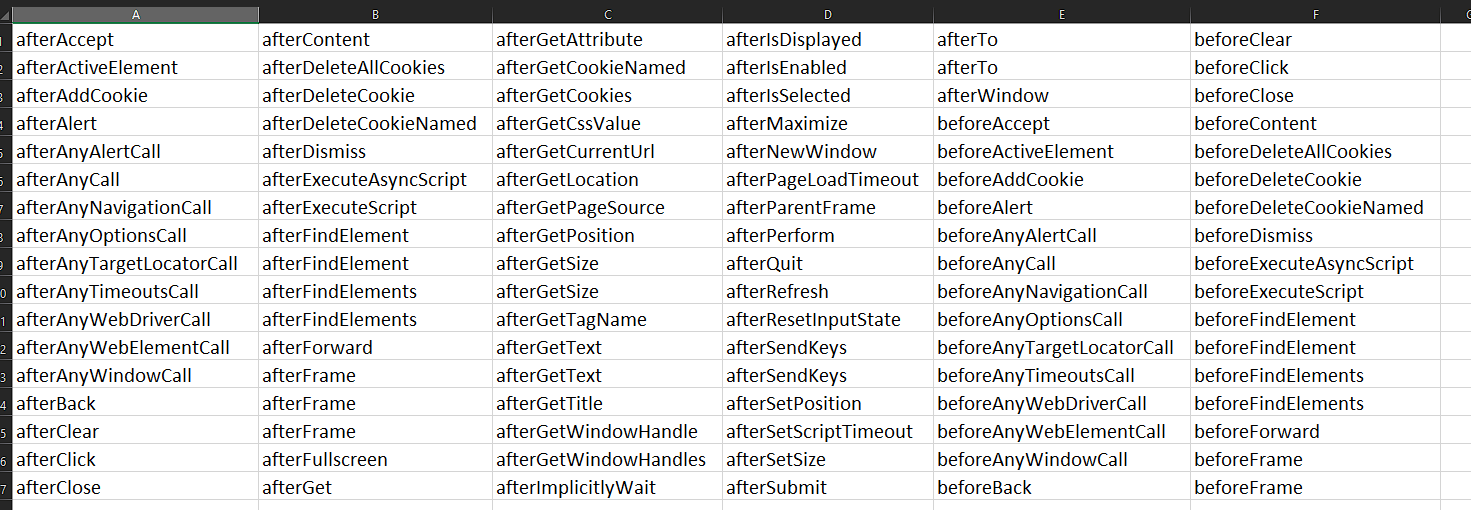


Output: Print statement would be printed in the console, it will not get reflected in the TestNG reports.

Output will be like this in the console,



There are more than 30 WebDriverListener methods available, out of which only a few is implemented for learning purpose. I will list out the remaining below





Selenium Exception Handling

Exception handling in Selenium (and in general programming) refers to the process of managing errors or unexpected events that occur during the execution of a program. In Selenium, exception handling is crucial because it allows your automation scripts to deal with various issues that might arise while interacting with a web application, such as elements not being found, timeouts, or unexpected alerts.

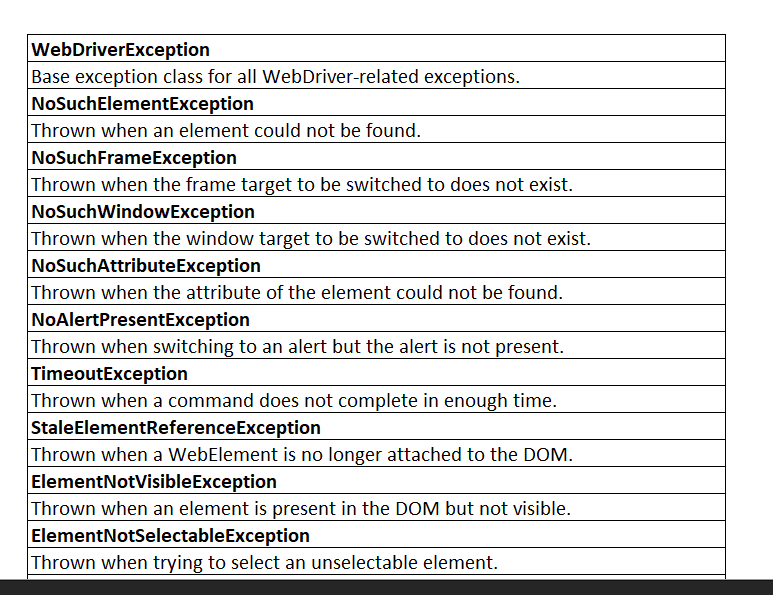
**Why Exception Handling is Important in Selenium:**

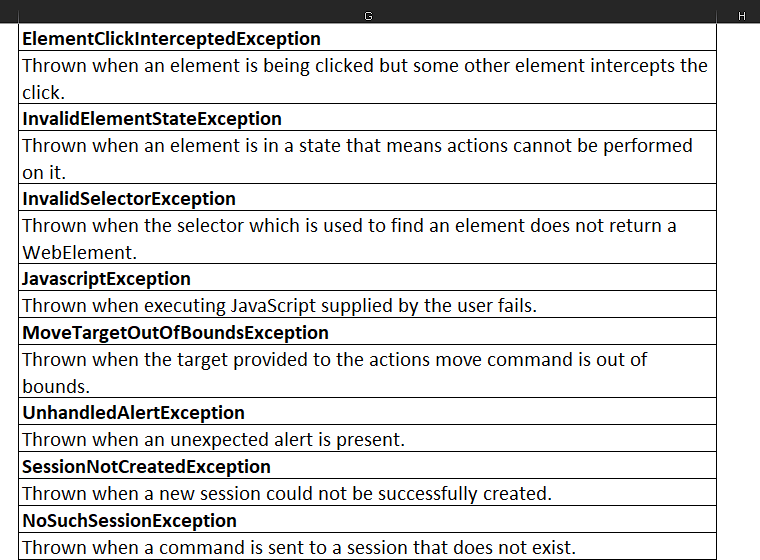
1. **Improves Script Robustness**: Properly handling exceptions ensures that your Selenium scripts are more resilient to common issues that may occur, such as changes in the web page's structure or slow loading elements.
2. **Prevents Script Termination**: Without exception handling, any unhandled exception will cause the script to terminate abruptly. By handling exceptions, you can continue the execution or log the error for further investigation.
3. **Provides Better Debugging**: Exception handling helps in identifying and logging issues, making it easier to debug the script and understand what went wrong.
4. **Enhances Automation Reliability**: By managing exceptions, you ensure that your automation tests are reliable and can run smoothly across different environments or application states.

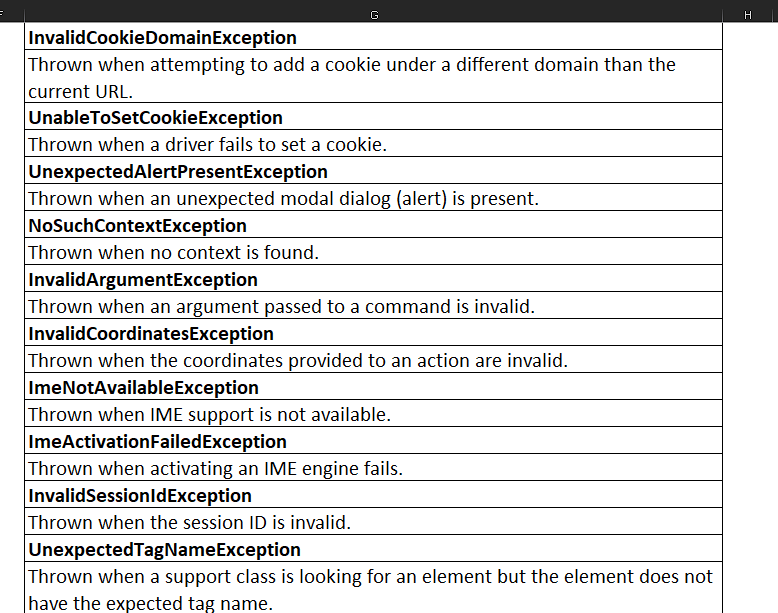
**How Exception Handling Works in Selenium:**

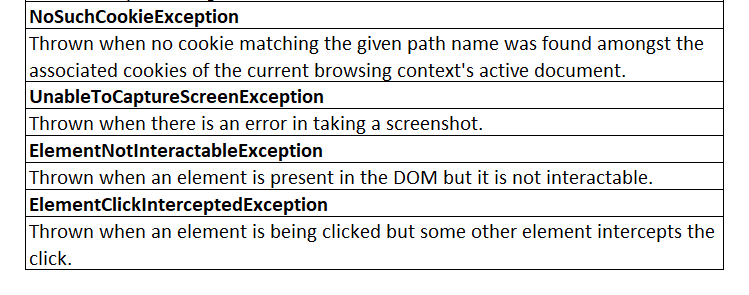
Exception handling in Selenium is typically implemented using try-catch blocks in Java (or equivalent in other languages). Here's how it works:

1. **try Block**: The code that might throw an exception is placed inside the try block. This is the code where you perform actions like finding elements, clicking buttons, or waiting for conditions.
2. **catch Block**: If an exception occurs within the try block, the program control is transferred to the corresponding catch block. Here, you can handle the exception by logging it, taking corrective actions, or retrying the operation.
3. **finally Block**: (Optional) The finally block contains code that will execute regardless of whether an exception occurred or not. It's often used for cleanup activities like closing the browser or releasing resources.





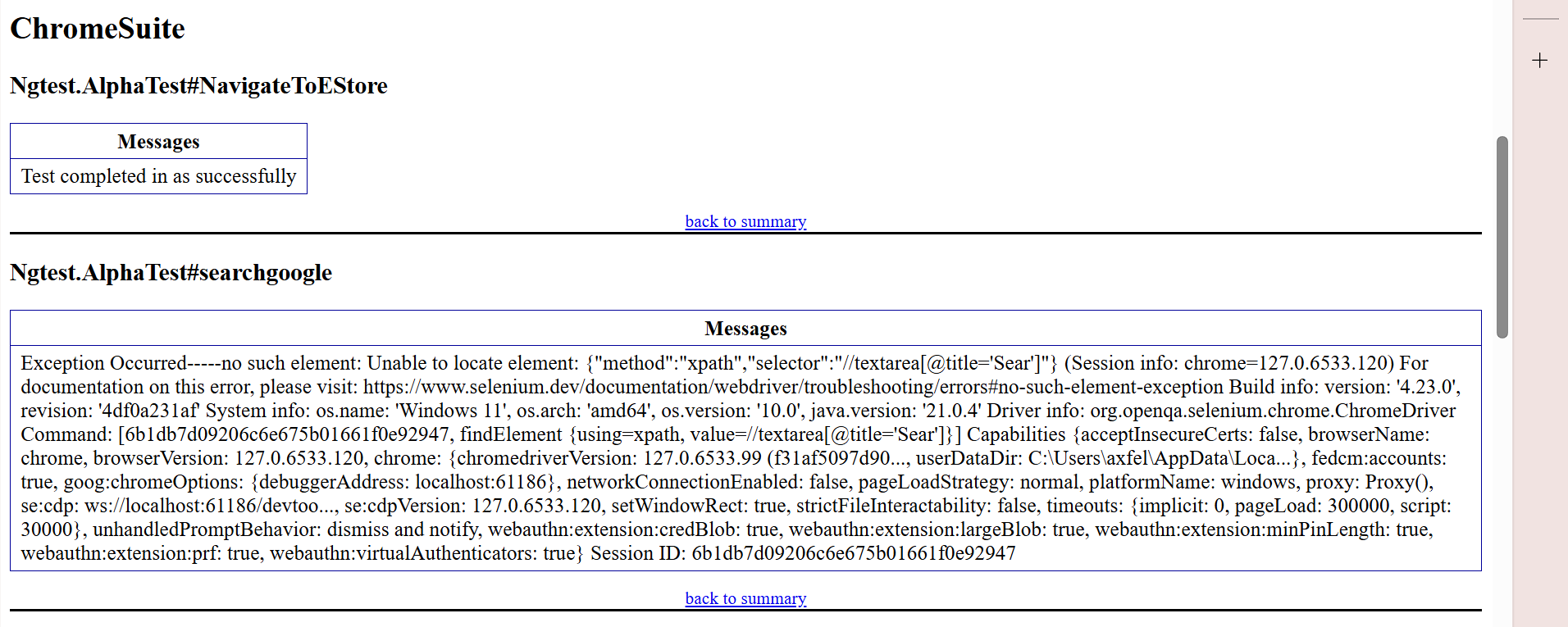




Implementation of No such element exception

Added try and Catch block in a @Test method with catch block having the reporter.log statement. This prints the text in the TestNG report with the stack trace.





This way we need to implement all other exceptions in the same way.

Advance Selenium topics

**11. Selenium with Cloud Testing Platforms**

* **Cloud-Based Test Execution**: Running Selenium tests on cloud-based platforms like BrowserStack, Sauce Labs, or LambdaTest for broader device and browser coverage.
* **Parallel Testing in the Cloud**: Configuring and optimizing tests to run in parallel on cloud platforms, reducing overall test execution time.

**Logging and Reporting**

* **Advanced Logging**: Implementing structured logging to capture detailed test execution logs, including screenshots, browser console logs, and network activity.
* **Custom Reporting**: Generating custom test reports with detailed insights using tools like Allure, ExtentReports, or custom HTML/PDF reports.

**17. Advanced Error Handling and Recovery**

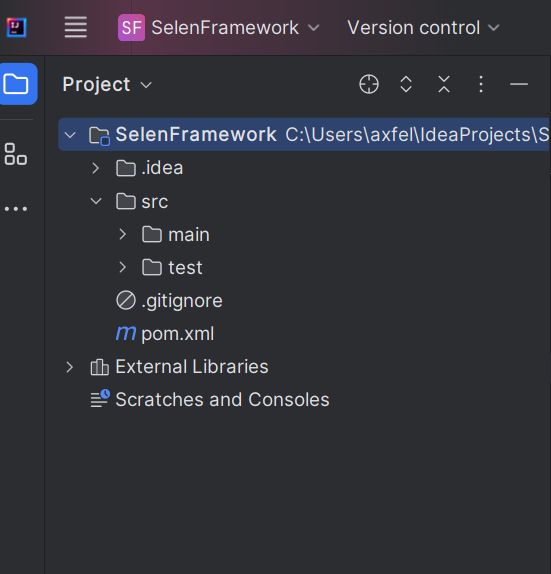
* **Exception Handling**: Implementing robust error handling mechanisms to deal with unexpected issues, such as stale elements, timeouts, or network failures.
* **Test Recovery**: Writing recovery scenarios to continue execution or retry tests in case of failure.

**18. Visual Testing**

* **Visual Regression Testing**: Integrating Selenium with visual testing tools like Applitools or using image comparison techniques to detect visual changes in the UI.
* **Screenshot Management**: Capturing and managing screenshots for visual verification and debugging.

Framework Setup in Selenium Using Maven & TestNG

Create a new Project with Maven Archetype as “QuickStart”.



Above screenshot shows a Structure of a Maven Project, which consists of src/main, src/test folders and pom.xml file.

Source files, properties file, configurations and other required files should be placed according to the folders explained below.

What is main and test folders under the src.

**“SelFrameworkStudy”** is the root directory as the maven setup.

**“.idea”** folder contains information specific to settings, project level condifurations and so on.. [IDE specific Folder]

Under **“src”**,

**“main/java”** contains the java files generic for your project.

**“main/resources”** contains the configuration files, property files common for the project.

**“test/java”** contains the java files related to test cases.

**“test/resources”** contains the test cases related config files, test data.

Dependencies are the important ones in the maven project.

For a simple selenium project, we need the below dependencies to be added in the POM file.

1. Selenium,
2. TestNG,
3. Maven sure fire plugin – for running tests, it supports Junit & TestNG.
4. Maven Jar Plugin – Creates Default Jar with Java files and resources available in src/main folder
5. Maven Assembly Plugin – Creates Jar with External Dependencies.

Maven POM file.

This POM file has all the dependencies except the Maven jar and assembly plugin which is not required.

Link for TestNG xml file is mentioned in the maven sure fire plugin dependency.

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>SelFramework</groupId>  
 <artifactId>SelenFramework</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>SelenFramework</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>4.23.0</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>1.7.36</version>  
 </dependency>  
 <dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>7.7.1</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.3.1</version>  
 <configuration>  
 <suiteXmlFiles>  
 <suiteXmlFile>src/test/resources/testng.xml</suiteXmlFile>  
 </suiteXmlFiles>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

Basic TestNG.xml file

This xml file has three suites named as Edgesuite, chromesuite, firefox suite.

All the three suites will be executed in Parallel and hence three thread count is given specific.

All three suites has different URL mentioned and same will be opened in the mentioned browser types on executing it.

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >  
<suite name="MySuite" parallel = "tests" thread-count="3">  
 <test name="EdgeSuite">  
 <parameter name="Browser" value="edge"/>  
 <parameter name="url" value="https://www.amazon.com"/>  
 <classes>  
 <class name="Ngtest.AlphaTest"/>  
 </classes>  
 </test>  
 <test name="ChromeSuite">  
 <parameter name="Browser" value="chrome"/>  
 <parameter name="url" value="https://www.flipkart.com/"/>  
 <classes>  
 <class name="Ngtest.AlphaTest"/>  
 </classes>  
 </test>  
 <test name="FirefoxSuite">  
 <parameter name="Browser" value="firefox"/>  
 <parameter name="url" value="https://www.nykaa.com/"/>  
 <classes>  
 <class name="Ngtest.AlphaTest"/>  
 </classes>  
 </test>  
</suite>