Day 1 of Automation

With Selenium

End Goal of this Session

- Learn to automate few actions on a web page
  - Launching a browser
  - Opening a demo e-commerce website / Getting the URL
  - Create a new user account
  - Sign in to the page.

Selenium Intro

What is Selenium?

- A range of tools and libraries to automate browser applications. That's it. Not a testing tool.

Range of tools ???

- Selenium WebDriver
- Selenium IDE
- Selenium Grid

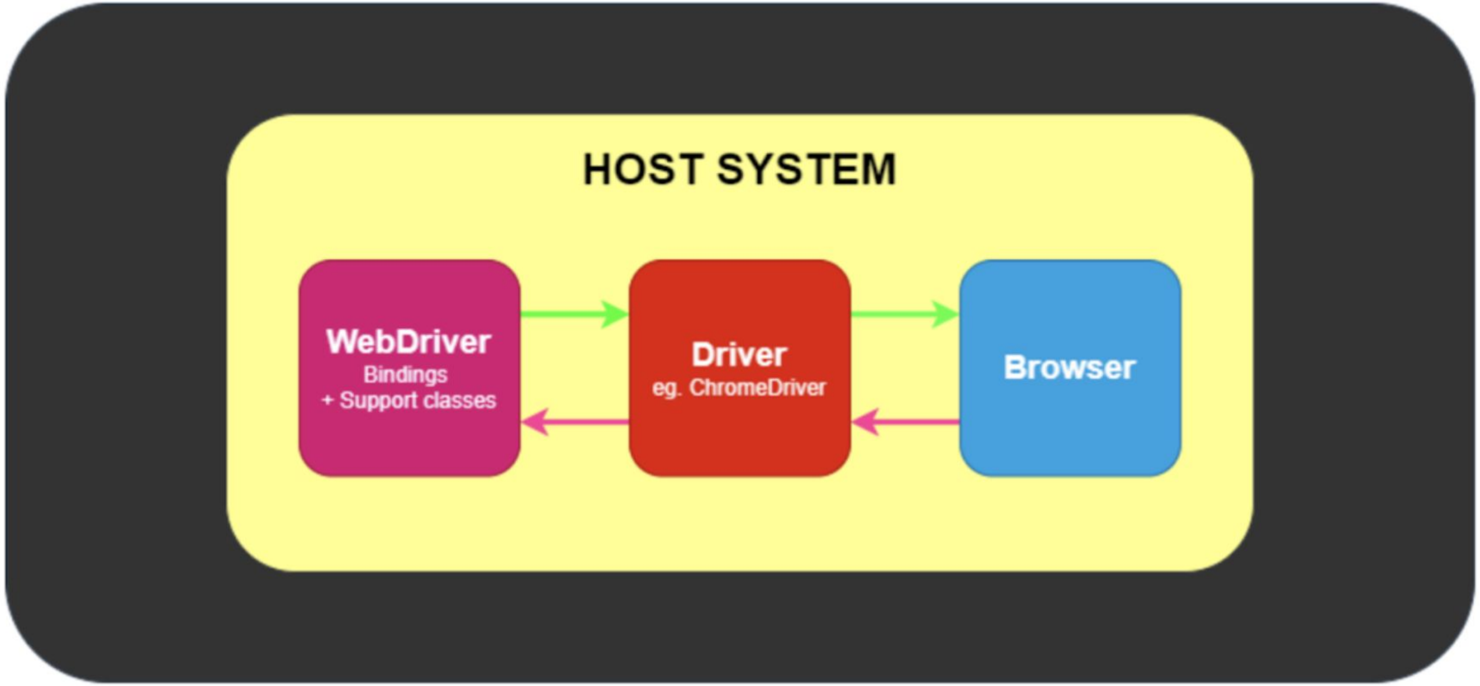
Selenium WebDriver

- Selenium WebDriver is a library to automate web applications.
- Supports Multiple Programming Languages including Java, Python, JS etc
- Helps to perform Effective Cross Browser Testing on Multiple OS

Supported Browsers

- Chrome
- FireFox
- Safari
- Internet Explorer
- Edge
- Opera

Selenium WebDriver Architecture



Create Java Project

- Create a Java Project
- Add Chrome driver [https://chromedriver.chromium.org/downloads]
- Add Selenium jar[https://selenium-release.storage.googleapis.com/index.html?path=3.141/]

Launch Chrome Browser

- Set the property and create the instance of Chrome Driver to launch the browser
  - System.setProperty(“webdriver.chrome.driver”,”path of exe file”)
  - WebDriver driver = new ChromeDriver();
- Navigate to url
  - driver.get(“https://spree-vapasi-prod.herokuapp.com/”)
- How to find the element - DOM
- Quit the browser
  - driver.close()
- JavaDoc link:
  - https://www.javadoc.io/doc/org.seleniumhq.selenium/selenium-api/3.141.59/org/openqa/selenium/WebElement.html

Breakout Room Activity

Time: 20 mins

Set up your machine with the .jar file and chromedriver.exe and launch your chrome browser and navigate to Spree Site

FindElement

- Web Elements(Buttons,text box) that are available in the webpage are used for performing various operations on the application.
- To interact with WebElements, we first have to identify these elements on the web page by specifying the attributes such as Id of the element or class name of the element and such other parameters.
- These alternatives using which we can find elements on a webpage are called locator strategies.
- Id,name, class,css,xpath
- WebDriver - https://www.javadoc.io/doc/org.seleniumhq.selenium/selenium-api/3.141.59/org/openqa/selenium/WebDriver.html

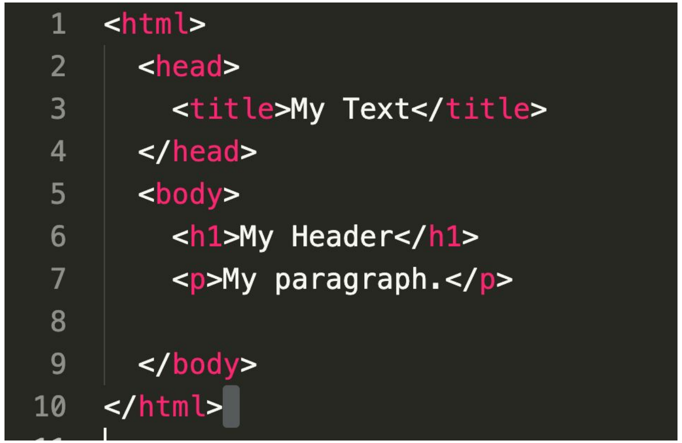
HTML/ HyperText Markup Language

- HTML is the language to tell the browser how to structure your webpage.
- Syntax of an element: <tag name attribute_name= " attr_value"> content </ tag name>

- Tags: keywords used to wrap an element
  - <html>
  - <body>
  - <button>
  - <input>
- Attributes: provide additional information about elements
  - Examples of attributes:
    - id
    - class
    - href
    - width,height

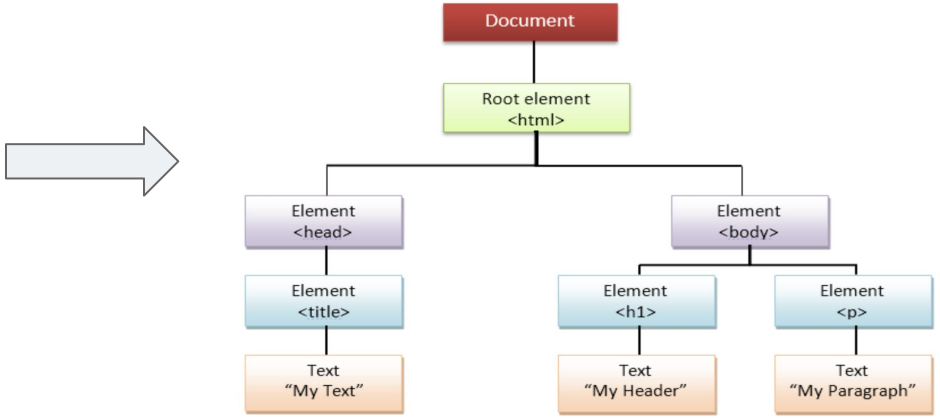
DOM (Document Object Model)

- Browser interacts with the HTML using the DOM API
- The DOM is a tree-like representation of the web page that gets loaded into the browser.



```html
<html>
  <head>
    <title>My Text</title>
  </head>
  <body>
    <h1>My Header</h1>
    <p>My paragraph.</p>

  </body>
</html>
```



How to locate an element in DOM (DEMO)

- Id
- name
- Class
- XPATHs:
  - Absolute XPATH
  - Relative XPATH
- Css Selectors : Syntax: htmltag[attribute = ‘attribute value’]
- Locating them and showing different attributes in each tag - Console

Time - 20
Mins

Breakout Room for Activity

Interacting with Console and understanding DOM

- Find the Logo and the Login link using id
- Find the Search and All departments dropdown using name
- Locate the sidebar using class selector
- Locate the name of the product ‘Ruby on Rails Bag’ using CSS Selectors
- Locate xpath for Login button

Find Element By Id

- Usually elements are associated with the ids in any webpage, because id is always unique
- This is by far the easiest and quickest way to identify elements because of its simplicity

- Syntax:

```
WebElement element = driver.findElement(By.id("Element_Id"));
```

- Example:

```
WebElement emailTextBox =
driver.findElement(By.id("spree_user_email"));
emailTextBox.sendKeys("te@gm.com");
```

Find Element By Name

- Name will return the first matching element only, if there multiple elements in the form matching the name, it will ignore others
- same as id except that the locator locates an element using the "name" instead of "id "
- The value of the NAME attribute accepted is of type String
- Syntax:
  WebElement element = driver.findElement(By.name("Element_NAME"));
- Example:

```
WebElement passwordTextBox =
driver.findElement(By.name("spree_user[password]"));

passwordTextBox.sendKeys("Welcome");
```

Find Element By Class

- Find the elements using class
- Not a good practice to use class as it will be changing
- CSS class applies to a group of DOM elements
- Class: Checkbox and Login button, Image.
    - Sidebar item
    - Shop By Categories & Shop By Brand
        - .taxonomy-root.h5

Find Element By CssSelector

- Identifies elements based on HTML structure
- one of the most efficient strategies to locate dynamic elements that don’t have consistent HTML attributes
- Syntax:

```
tag_name[attribute_name = ‘attribute_value’]
```

Find Element By Xpath

- navigates through the structure of the HTML or XML documents
- Syntax: `WebElement elem = driver.findElement(By.xpath(“ElementXPathExpression”));`
- Example: `WebElement buttonLogin = driver.findElement(By.xpath("//button[@id = 'submit']"));`

- **Project creation recap**
- **Locators recap**
- **IDE help (arguments)**
- **Findelement**
  - Id
  - Name
  - Class
  - Css
  - Xpath
- **Signup hands-on coding by trainer**

BO

API Documentation

- WebElement -- https://www.javadoc.io/doc/org.seleniumhq.selenium/selenium-api/3.141.59/org/openqa/selenium/WebElement.html
- WebDriver --

  https://www.javadoc.io/doc/org.seleniumhq.selenium/selenium-api/3.141.59/org/openqa/selenium/WebDriver.html

API code and show

- Login
  - checkbox
- In Homepage
  - Select (dropdown)
  - Search

    BO

- Add item to cart
- Get cart items (get text)

    BO

Hardcoded sleeps

Implicit Wait

- Implicit Wait tells the WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.
- Using implicit wait makes the test to execute slow, as it waits for each DOM to load for a certain amount of time.

  Example:

  ```
  driver.manage().timeouts().implicitlywait(10,TimeUnit.SECONDS);
  ```

BO - to remove hard sleeps

Breakout Room for Activity

Find element and interact with them

Time : 30 mins

Activity: Create an Account flow in Spree website

Breakout Room for Activity

Find element and interact using XPATH

Activity: Verify the email id in My Account Tab

FindElements

- used for finding the bulk of WebElements and its return type is List<WebElement>.
- If findelements method is not able to find even a single WebElement, It will return an empty list
- Example:

```
Find the Number of Products
System.setProperty("webdriver.chrome.driver","path of exe file");
WebDriver driver = new ChromeDriver();
driver.get("https://spree-vapasi-prod.herokuapp.com/");
List<WebElement> productsInFirstPage =
driver.findElements(By.cssSelector("div[id='products']>div"));
int noOfProductsInFirstPage=productsInFirstPage.size();
WebElement firstProduct= productsInFirstPage.get(0);
Boolean firstProductIsPresent = firstProduct.isDisplayed();
String productName = firstProduct.getText();
```

Breakout Room Activity

- Find the number of titles of the product and Print the all of the titles

Explicit Wait

- Sometimes page loading is done but the web element loading is not done. Here implicit wait will not help as it will look for the page loading
- Explicit wait tells WebDriver to wait for a specified element for a certain amount of time. Polling time for explicit wait is 500ms.
- Example:

```
WebDriverWait wait =new WebDriverWait(driver,10);
wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath());
```

FluentWait

- AJAX
    - Asynchronous JavaScript Xml
    - Loading will not be in a specific order (Example: Facebook posts loaded in a page)
    - Does not require page to refresh every time to load the WebElement
    - Fluent Wait class implements Wait interface.

FluentWait continue..

Example:

```
FluentWait<WebDriver> wait = new FluentWait<WebDriver>(driver);
wait.withTimeout(Duration.ofSeconds(20));
wait.pollingEvery(Duration.ofMillis(100));
wait.ignoring(NoSuchElementException.class);

WebElement test = wait.until(new Function<WebDriver,WebElement>(){
        public WebElement apply (WebDriver driver){
        return driver.findElement(By.id(“”));
}
});
test.click();
```

Breakout Room Activity

Use waits wherever NoSuchElementException Comes In
Time: 20 mins

Actions

- Intro to Mouse Interactions
- Actions Class
- Mouse over -> moveTo Element()+perform()
- Click -> click()+perform()
- Enter Data -> sendKeys()[present in WebElement Interface as well as Actions class]+perform()
- Right click ->contextClick()+perform()
- Range Slider -> moveByOffSet()+perform()
- Drag and Drop -> dragAndDrop()+perform(); dragAndDropBy()+perform()
- Login to the Account using Mouse Interactions

Actions

- Intro to Mouse Interactions
- Actions Class
- Mouse over -> moveTo Element()+perform()
- Click -> click()+perform()
- Enter Data -> sendKeys()[present in WebElement Interface as well as Actions class]+perform()
- Right click ->contextClick()+perform()
- Range Slider -> moveByOffSet()+perform()
- Drag and Drop -> dragAndDrop()+perform(); dragAndDropBy()+perform()
- Login to the Account using Mouse Interactions

Breakout Room Activity

Activity: Login to the Account and Click on Bag under Shop By Categories using Actions

FireFox Driver

- Download FireFox driver and add it to the Drivers folder
- FireFox Driver [https://github.com/mozilla/geckodriver/releases/tag/v0.30.0]
- System.*setProperty*("webdriver.gecko.driver","./Drivers/geckodriver");

```
WebDriver driver = new FirefoxDriver();
```

Take Screenshot

- To take screenshot of the webpage, we have a getScreenShotAs(OutputType.FILE) method in TakesScreenshot.

```
Class Screenshot
{
public static void main(String[] args)
          {
                              System.setProperty("webdriver.chrome.driver","path of exe file");
                              WebDriver driver = new ChromeDriver();
                              driver.manage().window().maximize();
                              driver.get("https://spree-vapasi-prod.herokuapp.com/");
                              TakesScreenshot ts = (TakesScreenshot) driver;
                              File srcFile = ts.getScreenshotAs(OutputType.FILE);
                              File destFIle = new File("./ScreenShots/1/screenshot.png");
                              try{
                              Files.copy(srcFile,destFile);
                              }
                              catch(IoException e)
                              // throwing the exception is not a good practice. Exceptions should be handled
                              {
                              }

          }
}
```

Breakout Room Activity

Launch Firefox Driver, Navigate to url and TakeScreenshot

JavaScript Executor

```
Class ScrollIntoView{
Public static void main(String[] args){
        System.setProperty("webdriver.chrome.driver","path of exe file");
        ChromeDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://spree-vapasi-prod.herokuapp.com/");
        JavascriptExecutor je = (JavascriptExecutor) driver;
        String jsCode = "arguments[0].scrollIntoView(true)";
        je.executeScript(jsCode,driver.findElement(By.className(")));

}
}
```

What are the challenges with building code manually?

- Need to download dependent libraries manually.
- Set classpath and maintain classpath
- Manual overhead of compiling changes code

Solution

Maven™

Gradle

Maven

- Maven is a build tool
- It helps in taking care of
  - Building application code (Compile)
  - Package into a single executable file (Jar)
  - Maintains the libraries needed for the application and includes in the output

Maven

- Create maven project
- Compile
    - mvn compile
- Run
    - mvn exec:java “-Dexec.mainclass=org.example.App”

Breakout Room Activity

- Create maven project
- Compile and Run