accenture
Latvia ATC

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Selenium WebDriver Basics

Selenium WebDriver basics

- **Introduction to Selenium WebDriver and set up**
- **Webpage source code**
- **Locators**
- **Asserts**
- **Actions on a page**
- **Alerts and pop-ups**

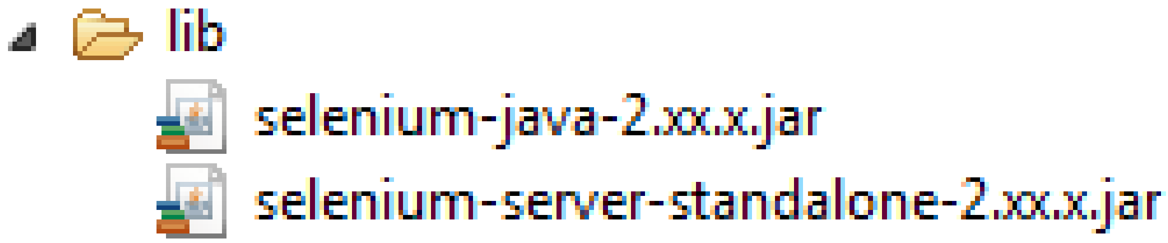
Introduction to Selenium WebDriver and set up

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Introduction to selenium and set up

Selenium WebDriver allows us to interact directly with browser (Firefox by default). In order to do that we must have 2 selenium jar libraries:

- **selenium-java**
- **selenium-server-standalone**



lib
selenium-java-2.xx.x.jar
selenium-server-standalone-2.xx.x.jar

Code example:

```
import org.openqa.selenium.firefox.FirefoxDriver;

public class SeleniumSample {
    public static void main(String arg[]) {
        WebDriver driver = new FirefoxDriver();
```

Introduction to selenium and set up

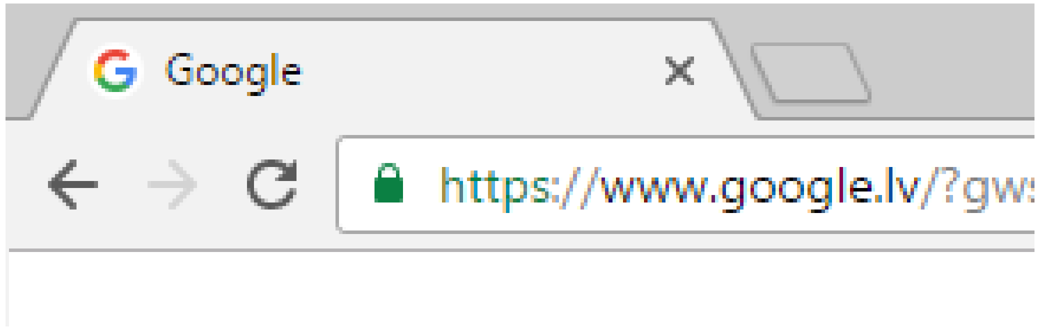
Then we can go to any specific page and close the driver.

```
WebDriver driver = new FirefoxDriver(); // start & open driver
driver.get("http://www.google.com"); //go to URL
System.out.println(driver.getCurrentUrl()); //get URL of page
driver.quit(); //stop & close driver
```

Or print out the page title first

```
System.out.println(driver.getTitle());
```

Title is what we see in the name of Tab:



Activity 1

Open in browser https://github.com/uljanovs/selenium_java_basic
(the link should have also sent to you via email)

Try:
1. **Running sample1 as JUnit and see output**
2. **Debug Sample1 with breakpoints**
3. **Change the URL, run and see how output changes**
4. **Commit our changes to git in our branch (<name_surname>)**

Webpage source code

accenture
Latvia ATC

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Webpage source code

When we are looking at page source we see something like:

```
<html>
    <head>
        <title>Page Title</title>
    </head>
        <body>
            <h1>Header</h2>
            <p>Text</p>
        </body>
</html>
```

Page source contains opening and closing tag, which page the same name (html, head, body etc.) difference is the slash at the beginning of ending tag.

Webpage source code

Or sometimes like:

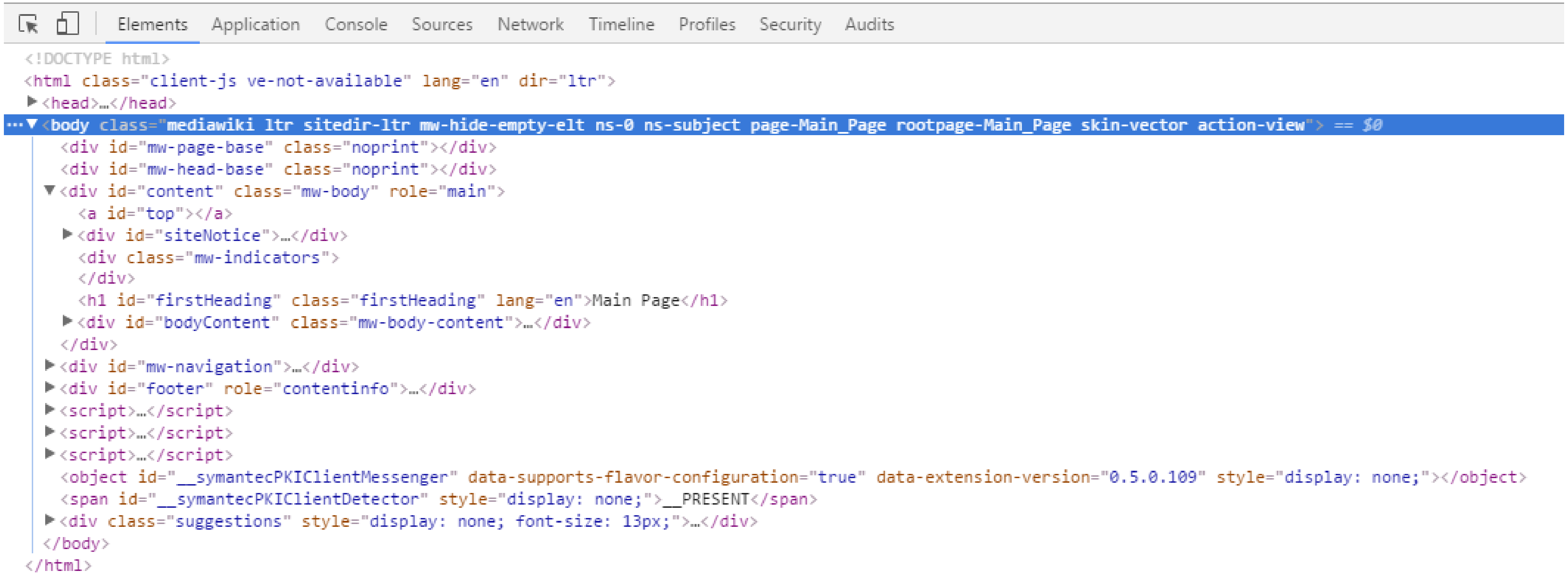
```
<html>
        <head>
                <title>Page Title</title>
        </head>
        <body>
                <input type=“text” value=“Search” />
                <input type=“submit” value=“Go” />
        </body>
</html>
```

In there the `input` **is closing and opening at the same time, because it has slash before closing angle bracket.**

Webpage source code

accenture
Latvia ATC

After opening any page we need to be able to get any element on page. For that Locators are used. To find any locators open browser and click F12. E.g. in Chrome on Wiki page:



```
<!DOCTYPE html>
<html class="client-js ve-not-available" lang="en" dir="ltr">
 ▶<head>…</head>
…▼<body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Main_Page rootpage-Main_Page skin-vector action-view"> == $0
     <div id="mw-page-base" class="noprint"></div>
     <div id="mw-head-base" class="noprint"></div>
    ▼<div id="content" class="mw-body" role="main">
       <a id="top"></a>
      ▶<div id="siteNotice">…</div>
       <div class="mw-indicators">
       </div>
       <h1 id="firstHeading" class="firstHeading" lang="en">Main Page</h1>
      ▶<div id="bodyContent" class="mw-body-content">…</div>
     </div>
    ▶<div id="mw-navigation">…</div>
    ▶<div id="footer" role="contentinfo">…</div>
    ▶<script>…</script>
    ▶<script>…</script>
    ▶<script>…</script>
     <object id="__symantecPKIClientMessenger" data-supports-flavor-configuration="true" data-extension-version="0.5.0.109" style="display: none;"></object>
     <span id="__symantecPKIClientDetector" style="display: none;">__PRESENT</span>
    ▶<div class="suggestions" style="display: none; font-size: 13px;">…</div>
   </body>
</html>
```

Webpage source code

What you see on page are elements:
- `div` **(division of a page)**
- `p` **(paragraph)**
- `h1` **(header 1)**
- `input` **(field were user can enter data)**
- **etc**

And there attributes:
- `id` **(**`<div id="navigation">…</div>`**)**
- `class` **(**`<p class="text'>…</p>)`
- `name` **(**`<input name=        "Submit" value=""Submit />`**)**

Searching elements on page

In order to find an element on page “`driver.findElement(By locator)`**” function is used and for multiple elements** `driver.findElements(By locator)`**”**

- **“**`driver.findElement(By locator)`**” function returns 1 webElement, if no element was found a NoSuchElementException is thrown, if more then one element is found the first one is returned.**
- **“**`driver.findElements(By locator)`**” function returns a list of webElement, no error is thrown if search returns empty list.**
  - ***If specific element is needed - “***`get(number)`***” is used (the numbering starts from 0), e.g. “***`driver.findElements(By locator).get(0)`***” for first element.***
  - ***If number is too big is IndexOutOfBoundsException thrown.***

Locators

accenture
Latvia ATC

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Locators

There are different type of locators:

- **id;**
- **class;**
- **name;**
- **tagName;**
- **xPath;**
- **linkText;**
- **cssSelector.**

Locators by id

Lets say you want to find element. We left click the element and click “Inspect” or “inspect Element” in Firefox. E.g.:

```
<p id=“id1">Text</p>
```

We can find this element by id, e.g.:

```
driver.findElement(By.id(“id1”));
```

Would return a WebElement:

```
[[FirefoxDriver: firefox on WINDOWS (cf310a15-92b0-4910-ac30-9578ce72a0d6)] -> id: id1]
```

In order to get text inside of element “`.getText()`**” method is used. E.g.:**

```
driver.findElement(By.id(“id1")).getText();
```

Would return a String:

```
“Text”
```

Locators by name

By name:
`<input type=“text” name=“FirstName” value=“Mickey”>`
e.g.
`driver.findElement(By.name("FirstName"));`

Input doesn’t contain any text, so instead we can look at other attribute of an element with method “`.getAttribute(attribute_name)`**”, e.g.:**
`driver.findElement(By.name("FirstName")).getAttribute(“value”);`
Would return a String:
`“Mickey”`
or
`driver.findElement(By.name("FirstName")).getAttribute(“type”);`
Would return a String:
`“text”`

Locators by class and tagName

Lets say you want to find element. We left click the element and click “Inspect” or “Inspect Element” in Firefox. E.g.:

```
<p class=“class1">Text</p>
```

Or by class:

```
driver.findElement(By.className(“class1"));
```

Or same can be done by tagName:

```
driver.findElement(By.tagName(“p"));
```

Locators by class and tagName

Usually we have more than 1 element with same class or tag, this is why "`driver.findElements(By)`" is used more often.

```
<ul>
    <li>Homepage</li>
    <li>Contact us</li>
    <li>Login</li>
</ul>

List<WebElement> menu = driver.findElements(By.tagName(“li”));
System.out.print(menu.size()); // 3
for(WebElement menuItem : menu) {
    System.out.println(menuItem.getText());
}
System.out.print(menu.get(1)); // “Contact us”
```

Locators (xpath, css)

accenture
Latvia ATC

```
<div id="navigation">
    <h2>Navigation menu</h2>
</div>
```

In xpath:

```
driver.findElement(By.xpath("//div[@id='navigation']/h2"));
```

In css:

```
driver.findElement(By.cssSelector("div#navigation > h2"));
```

```
<p class="headline" id="main">Text</p>
```

In xpath:

```
driver.findElement(By.xpath("//span[@class='headline' and @id='main ']"));
```

In css:

```
driver.findElement(By.cssSelector(".headline#main"));
```

CSS vs xPath

Goal	CSS	xPath
All Elements	*	//*
All P Elements	p	//p
All Child Elements	p > *	//p/*
Direct child	div > a	//div/a
Child or sub child	div a	//div//a
Element By ID	#foo	//*[@id='foo']
Element By Class	.foo	//*[contains(@class,'foo')]
Element With Attribute	a[href='url']	//a[@href='url']
… (name="continue" and type="button")	input[name='continue'][type='button']	//input[@name='continue' and @type=' button']
A link with an "id" that contains the text "id_pattern"	a[id*='id_pattern']	//a[contains(@id, 'id_prefix_')]
Matching by inner text (exact)		//a[.='Find me I have nothing in me!!'] //a[text()='Find me I have nothing in me!!']

Activity 2

Fill in https://goo.gl/forms/6AZsHBnvBPkfjuOp2

(the link should have also sent to you via email)

Activity 2

1. **Create new “Sample2.java”**
   Note: page to use “Page Examples” -> “Locators”
2. **Create a new test method:**
   1. **findElementByID** - where you find element by id “heading_1” and then print out the text of this element;
   2. **findElementByName** - where you find element by name “randomButton1” and then print out “value” attribute of this element;
   3. **findElementByTagName** - where you find element by tagName “h2” and then print out “id” attribute of this element;
   4. **findElementByClassFirst** - where you find element by className “text” and then print out the text of this element.

Push all your changes to repository

Activity 2 (continue in Sample 2)

5. **findElementByClassAll - where you find elements by className “text” and then print out**
   - number of elements
   - the text of this elements
   - 3rd element
6. **findElementByXPath - where you find element by xpath:**
   - “//div[@id='nonStandartText']/*[contains(@class, 'amazing')]” and then print out the text of this element
   - “//p[@class='text' and @id='dummy']” and then print out the text of this element
7. **findElementByCssName - where you find element by css:**
   - “div#nonStandartText > .amazing” and then print out the text of this element
   - “.text#dummy” and then print out the text of this element

Push all your changes to repository

Asserts

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Asserts

accenture
Latvia ATC

Assert is used in java to verify or fail something. Assert types (default):

- `assertTrue(Boolean condition);`
- `assertTrue(String errorMessage, Boolean condition);`
- `assertFalse(Boolean condition);`
- `assertFalse(String errorMessage, Boolean condition);`
- `assertEquals(String expected, String actual);`
- `assertEquals(String errorMessage, String expected, String actual);`
- `fail();`
- `fail(String errorMessage);`
- **etc.**

For more assert see http://junit.sourceforge.net/javadoc/org/junit/Assert.html

Activity 3

- In this activity, you will:
  - Open the file ‘Sample3Task.java’
  - Read the instructions and create the code to complete this program.

Push all your changes to repository

Actions and checks on a page

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

What kind of actions can you name?

Actions on page - clicking

accenture
Latvia ATC

For clicking “`element.click()`” **method is used (usually) on element:**

- **link (**`<a href="#">Link</a>`**)**
- **button:**
  - ➢ `<button>Default Button</button>`
  - ➢ `<a href="#" class="button">Link Button</a>`
  - ➢ `<button class="button">Button</button>`
  - ➢ `<input type="button" class="button" value="Input Button">`
- **checkbox**
  - ➢ `<input type="checkbox" name="vehicle" value="Bike" />`
  - ➢ `<input type="checkbox" name="vehicle" value="Car" checked/>`
- **radio**
  - ➢ `<input type="radio" name="gender" value="male" checked>`
  - ➢ `<input type="radio" name="gender" value="female">`

Note: can be used on any elements if needed.

Actions on page – removing and entering text

For removing text “`element.clear()`**” method is used and for entering text “**`element.sendKeys()`**” (usually) on element:**

- **text input (one line)**

```
<input type="text" name="firstname" />
```

- **text area (many lines)**

```
<textarea rows="4" cols="50">
    This is a textarea.
</textarea>
```

Actions on page – selecting

For selecting from select dropdown - 3 methods can be used:

- `element.selectByVisibleText(“Text”)`
- `element.selectByIndex(0)`
- `element.selectByValue(“value")`

Select dropdown code example:

```
<select>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
</select>
```

Checks on page

For verifying on page is something selected/enabled/displayed following methods are used:

- `element.isSelected()`
- `element.isEnabled()`
- `element.isDisplayed()`

Activity 4

- In this activity, you will:
  - Open the file ‘Sample4Task.java’
  - Read the instructions and create the code to complete this program.

Push all your changes to repository

Alerts and pop-ups

Test Automation Engineering Fundamentals: Java + Selenium WebDriver

Alerts and pop-ups

Decision making is one of the critical facilities which a programming language has to provide. To understand Java better we have to understand this chapter and practice by writing some decision making code.

There are two types of decision making statements in Java. One is very commonly used which is *If statement* and you will find it almost in every piece of code. Second is *Switch statement*.

Activity 5

- In this activity, you will:
  - Open the file ‘Sample4Task.java’
  - Read the instructions and create the code to complete this program.

Push all your changes to repository

QUESTIONS