Selenium with Java
Lecture 3

WebDriver Interface:

```
package MYPackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LoginTest {
        public static void main(String[] args) {
                // TODO Auto-generated method stub

//      ChromeDriver driver =  new ChromeDriver();
//      FirefoxDriver driver = new FirefoxDriver();
    EdgeDriver driver =  new EdgeDriver();
    driver.get("https://www.saucedemo.com/v1/");
    driver.findElement(By.id("user-name")).sendKeys("standard_user");
    driver.findElement(By.id("password")).sendKeys("secret_sauce");
    driver.findElement(By.id("login-button")).click();


        }
}
```

We able to execute our script in different browser

If your browser driver coming from external configuration -XLS, CSV

```
package MYPackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LoginTest {

        public static String browser = "Firefox"; //external configuration -XLS, CSV

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                if(browser.equals("Firefox")) {

                        FirefoxDriver driver = new FirefoxDriver();
                }
                else if(browser.equals("Chrome")) {
              ChromeDriver driver =  new ChromeDriver();
                }
                else if(browser.equals("Edge")) {
                        EdgeDriver driver =  new EdgeDriver();
                }
    driver.get("https://www.saucedemo.com/v1/");
    driver.findElement(By.id("user-name")).sendKeys("standard_user");
    driver.findElement(By.id("password")).sendKeys("secret_sauce");
    driver.findElement(By.id("login-button")).click();
        }
}
```

We getting the error because of driver object scope so i can declare it as global

```
package MYPackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LoginTest {

        public static String browser = "Firefox"; //external configuration -XLS, CSV
        public static FirefoxDriver driver;
        public static void main(String[] args) {
                // TODO Auto-generated method stub

                if(browser.equals("Firefox")) {

                        FirefoxDriver driver = new FirefoxDriver();
                }
                else if(browser.equals("Chrome")) {
              ChromeDriver driver =  new ChromeDriver();
                }
                else if(browser.equals("Edge")) {
                        EdgeDriver driver =  new EdgeDriver();
                }
    driver.get("https://www.saucedemo.com/v1/");
    driver.findElement(By.id("user-name")).sendKeys("standard_user");
    driver.findElement(By.id("password")).sendKeys("secret_sauce");
    driver.findElement(By.id("login-button")).click();
        }
}
```

If we create an instance of a browser class so script executed on that browser only.

```
package MYPackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LoginTest {

	public static String browser = "Firefox"; //external configuration -XLS, CSV
	public static FirefoxDriver driver;
	public static ChromeDriver driver1;
	public static EdgeDriver driver2;
	public static void main(String[] args) {
		// TODO Auto-generated method stub

		if(browser.equals("Firefox")) {
			FirefoxDriver driver = new FirefoxDriver();
		}
		else if(browser.equals("Chrome")) {
	    ChromeDriver driver =  new ChromeDriver();
		}
		else if(browser.equals("Edge")) {
			EdgeDriver driver =  new EdgeDriver();
		}
    driver.get("https://www.saucedemo.com/v1/");
    driver.findElement(By.id("user-name")).sendKeys("standard_user");
    driver.findElement(By.id("password")).sendKeys("secret_sauce");
    driver.findElement(By.id("login-button")).click();
    driver.close();
	}
}
```

So i am creating an global variable of webdriver interface

```
package MYPackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LoginTest {

	public static String browser = "Firefox"; //external configuration -XLS, CSV
	public static WebDriver driver;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
```

```
                    if(browser.equals("Firefox")) {
                            driver = new FirefoxDriver();
                    }
                    else if(browser.equals("Chrome")) {
                driver =  new ChromeDriver();
                    }
                    else if(browser.equals("Edge")) {
                            driver =  new EdgeDriver();
                    }
        driver.get("https://www.saucedemo.com/v1/");
        driver.findElement(By.id("user-name")).sendKeys("standard_user");
        driver.findElement(By.id("password")).sendKeys("secret_sauce");
        driver.findElement(By.id("login-button")).click();
        driver.close();
            }
}
```

Locators:

Traditional Locators:

Locator	Description
class name	Locates elements whose class name contains the search value (compound class names are not permitted)
css selector	Locates elements matching a CSS selector
id	Locates elements whose ID attribute matches the search value
name	Locates elements whose NAME attribute matches the search value
link text	Locates anchor elements whose visible text matches the search value

partial link text	**Locates anchor elements whose visible text contains the search value. If multiple elements are matching, only the first one will be selected.**
tag name	**Locates elements whose tag name matches the search value**
xpath	**Locates elements matching an XPath expression**

```
package MYPackage;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Locators {

        public static String browser = "Chrome"; //external configuration -XLS, CSV
        public static WebDriver driver;

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                if(browser.equals("Firefox")) {
                        driver = new FirefoxDriver();
                }
                else if(browser.equals("Chrome")) {
              driver =  new ChromeDriver();
                }
                else if(browser.equals("Edge")) {
                        driver =  new EdgeDriver();
                }
//      driver.get("https://www.saucedemo.com/v1/");
     //By Id
//      driver.findElement(By.id("user-name")).sendKeys("Testing");
```

```
        //By ClassName
//        driver.findElement(By.className("form_input")).sendKeys("Testing");

        //By CSS- this is not advisable
//        driver.findElement(By.cssSelector("#user-name")).sendKeys("Testing");

      //By xpath
//        driver.findElement(By.xpath("//*[@id=\"user-name\"]")).sendKeys("Testing");

        //By tagName
//        driver.findElement(By.tagName("input")).sendKeys("Testing");

        driver.get("https://www.selenium.dev/documentation/overview/");
      //By linkText
//        driver.findElement(By.linkText("Documentation")).click();

      //By PartiallinkText
        driver.findElement(By.partialLinkText("Documen")).click();

           }

}
```

Utilizing Locators

By locator

```
Unset
    import org.openqa.selenium.By;
    WebDriver driver = new ChromeDriver();
      driver.findElement(By.className("information"));
```

ByChained

The ByChained class enables you to *chain* two By locators together. For example, instead of having to locate a parent element, and then a child element of that parent, you can instead combine those two FindElement() functions into one.

Unset

```
        By example = new ByChained(By.id("login-form"),
By.id("username-field"));

            WebElement username_input =
driver.findElement(example);
```

Copy

ByAll

The ByAll class enables you to utilize two By locators at once, finding elements that mach *either* of your By locators. For example, instead of having to utilize two FindElement() functions to find the username and password input fields seperately, you can instead find them together in one clean FindElements()

Unset

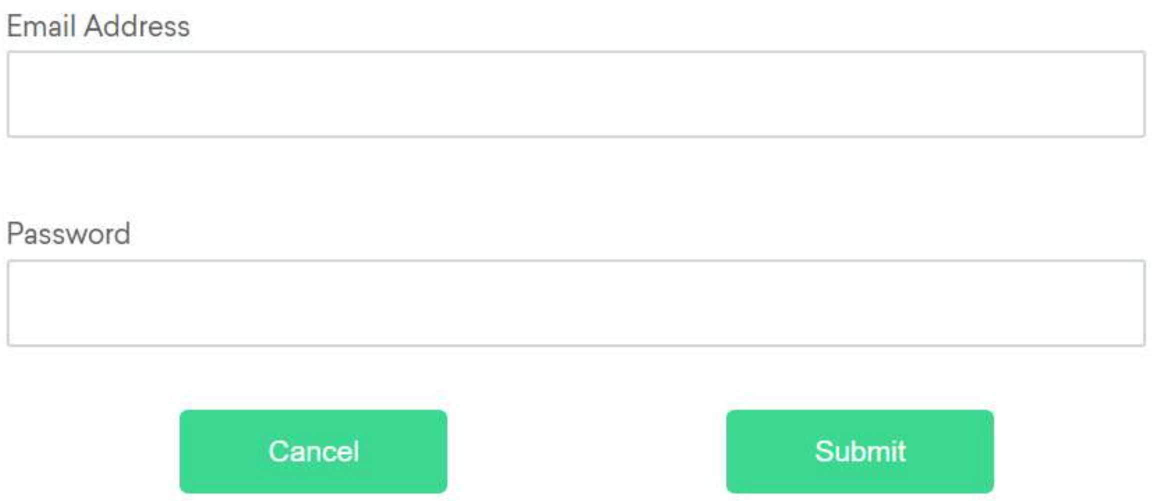
```
        By example = new ByAll(By.id("password-field"),
By.id("username-field"));

            List<WebElement> login_inputs =
driver.findElements(example);
```

Copy

Relative Locators

Selenium 4 introduces Relative Locators (previously called *Friendly Locators*). These locators are helpful when it is not easy to construct a locator for the desired element, but easy to describe spatially where the element is in relation to an element that does have an easily constructed locator.

Let us consider the below example for understanding the relative locators.



Available relative locators

Above

If the email text field element is not easily identifiable for some reason, but the password text field element is, we can locate the text field element using the fact that it is an "input" element "above" the password element.

Unset
```
By emailLocator =
RelativeLocator.with(By.tagName("input")).above(By.id(
"password"));
```
Copy

Below

If the password text field element is not easily identifiable for some reason, but the email text field element is, we can locate the text field element using the fact that it is an "input" element "below" the email element.

Unset

```
By passwordLocator =
RelativeLocator.with(By.tagName("input")).below(By.id(
"email"));
```

Copy

Left of

If the cancel button is not easily identifiable for some reason, but the submit button element is, we can locate the cancel button element using the fact that it is a "button" element to the "left of" the submit element.

Unset

```
By cancelLocator =
RelativeLocator.with(By.tagName("button")).toLeftOf(By.id("
submit"));
```

Copy

Right of

If the submit button is not easily identifiable for some reason, but the cancel button element is, we can locate the submit button element using the fact that it is a "button" element "to the right of" the cancel element.

Unset

```
By submitLocator =
RelativeLocator.with(By.tagName("button")).toRightOf(By.id(
"cancel"));
```

Copy

Near

If the relative positioning is not obvious, or it varies based on window size, you can use the near method to identify an element that is at most 50px away from the provided locator. One great use case for this is to work with a form element that doesn't have an easily constructed locator, but its associated input label element does.

Unset
```
By emailLocator =
RelativeLocator.with(By.tagName("input")).near(By.id("lbl-email"));
```
Copy

Chaining relative locators

You can also chain locators if needed. Sometimes the element is most easily identified as being both above/below one element and right/left of another.

Unset
```
By submitLocator =
RelativeLocator.with(By.tagName("button")).below(By.id("email")).toRightOf(By.id("cancel"));
```
Copy