Selenium

Class 6

Agenda

Switch To Commands:

Handle Alerts

Handle Frames

Handle Windows

Alerts/PopUps

Alerts are basically popup boxes that away focus from the current browser and forces to read the alert message.

We need to do some action such as accept or dismiss the alert box to resume our task on the browser.

Alert prevents the user from accessing other parts of the web page until the box is closed.

Alerts will not allow any action on the underlying webpage if they are present. So if an alert is present on the webpage and you try to access any of the element in the underlying page you will get following exception:

UnhandledAlertException: Modal dialog present

Alerts/PopUps

There are two types of alerts:

Windows Based Alerts
Web Based/Browser Based/JavaScript Alerts

To handle alert window in Selenium Webdriver we have predefined Interface Alert .

Since alert is separate window so before using methods from Alert Interface we have to switch to alert window using **switchTo()** method

Alert alert = driver.switchTo().alert();

Handle Alerts

Selenium provides us with an interface called Alert.

It is present in the org.openqa.selenium.Alert package.

Alert interface gives us following methods to deal with the alert:

- accept() To accept the alert
- dismiss() To dismiss the alert
- getText() To get the text of the alert
- sendKeys() To write some text to the alert

Simple Alert

Simple alerts just have a OK button on them. They are mainly used to display some information to the user.

Important point to note is that we need to switch from main window to an alert using the driver.switchTo().alert().

```
Alert simpleAlert = driver.switchTo().alert();
String alertText = simpleAlert.getText();
system.out.println("Alert text is " + alertText);
simpleAlert.accept();
```

Confirmation Alert

This alert comes with an option to accept or dismiss the alert.

To accept the alert you can use Alert.accept() and to dismiss you can use the Alert.dismiss(). Here is the code to dismiss a prompt alert.

```
Alert confirmationAlert = driver.switchTo().alert();
String alertText = confirmationAlert.getText();
system.out.println("Alert text is " + alertText);
confirmationAlert.dismiss();
```

Prompt Alert

In prompt alerts you get an option to add text to the alert box.

This is specifically used when some input is required from the user.

We will use the sendKeys() method to type something in the Prompt alert box.

```
Alert promptAlert  = driver.switchTo().alert();
String alertText = promptAlert .getText();
system.out.println("Alert text is " + alertText);
//Send some text to the alert
promptAlert .sendKeys("Accepting the alert");
promptAlert .accept();
```

Handle IFrames / Frames

In some application for better visibility developer use frame concept in web pages.

What is iFrame? An iFrame (Inline Frame) is an HTML document embedded inside the current HTML document on a website.

iFrame HTML element is used to insert content from another source, such as an advertisement, into a Web page.

A Web designer can change an iFrame's content without making them reload the complete website.

A website can have multiple frames on a single page. And a frame can also have inner frames (Frame in side a Frame)

We can handle frames/iframes present in the webpage using **driver.switchTo()** command in selenium webdriver.

Frame/iFrame is nothing but **another webelement in html page, which displays another part of webpage.**

How to Identify Frame ?

There are many ways to identify the Frame. We will see the easiest way to identify the frame. Let's see the step by step procedure-

Right click on the element. Check if “This Frame” option is available. If This frame option is available, it means that the element is inside a frame.

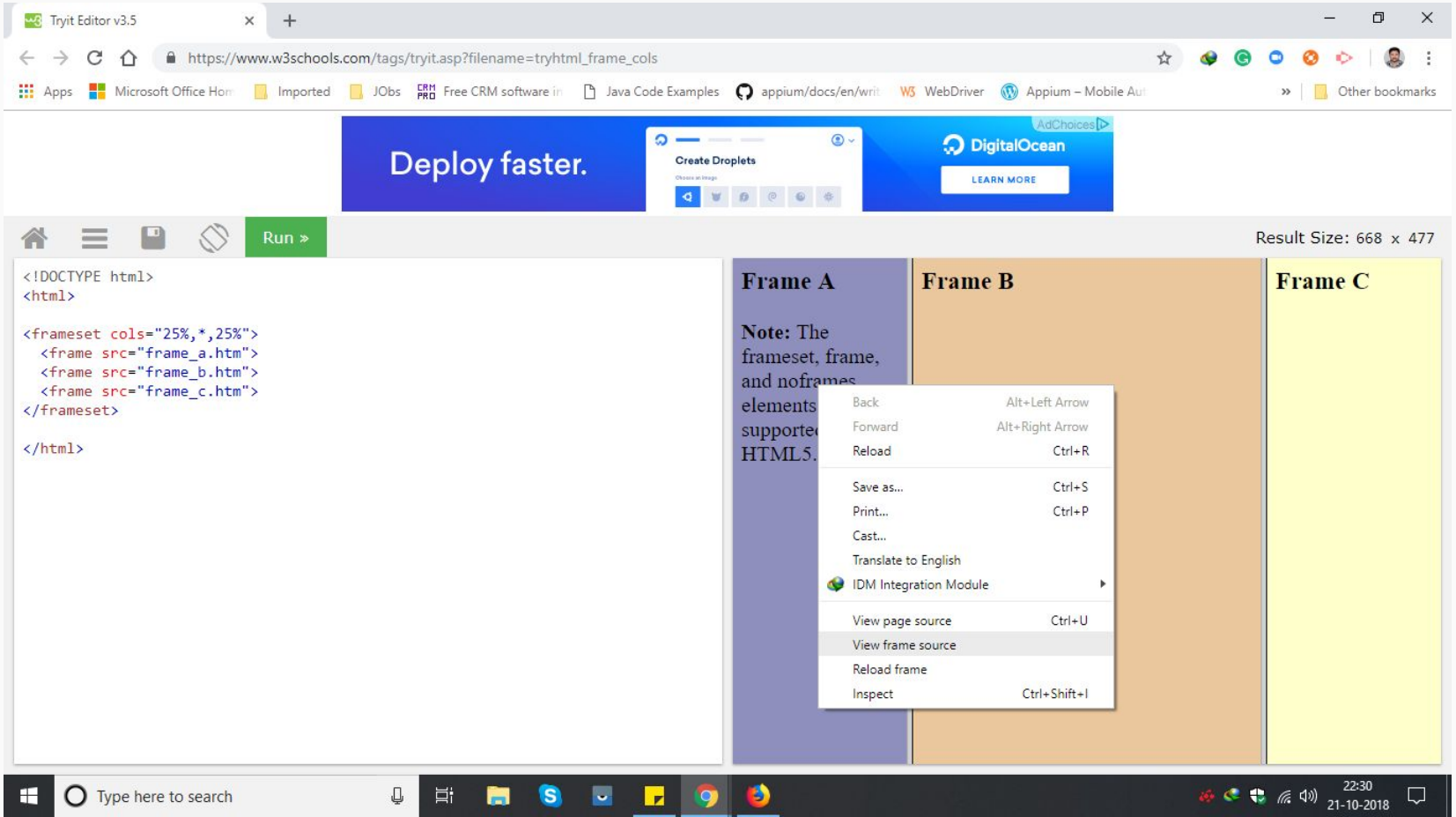
View page source of the web page and check if any tag is available for ‘iframe’.

It will show the iFrame tag for the frame inside which the webelement is present.

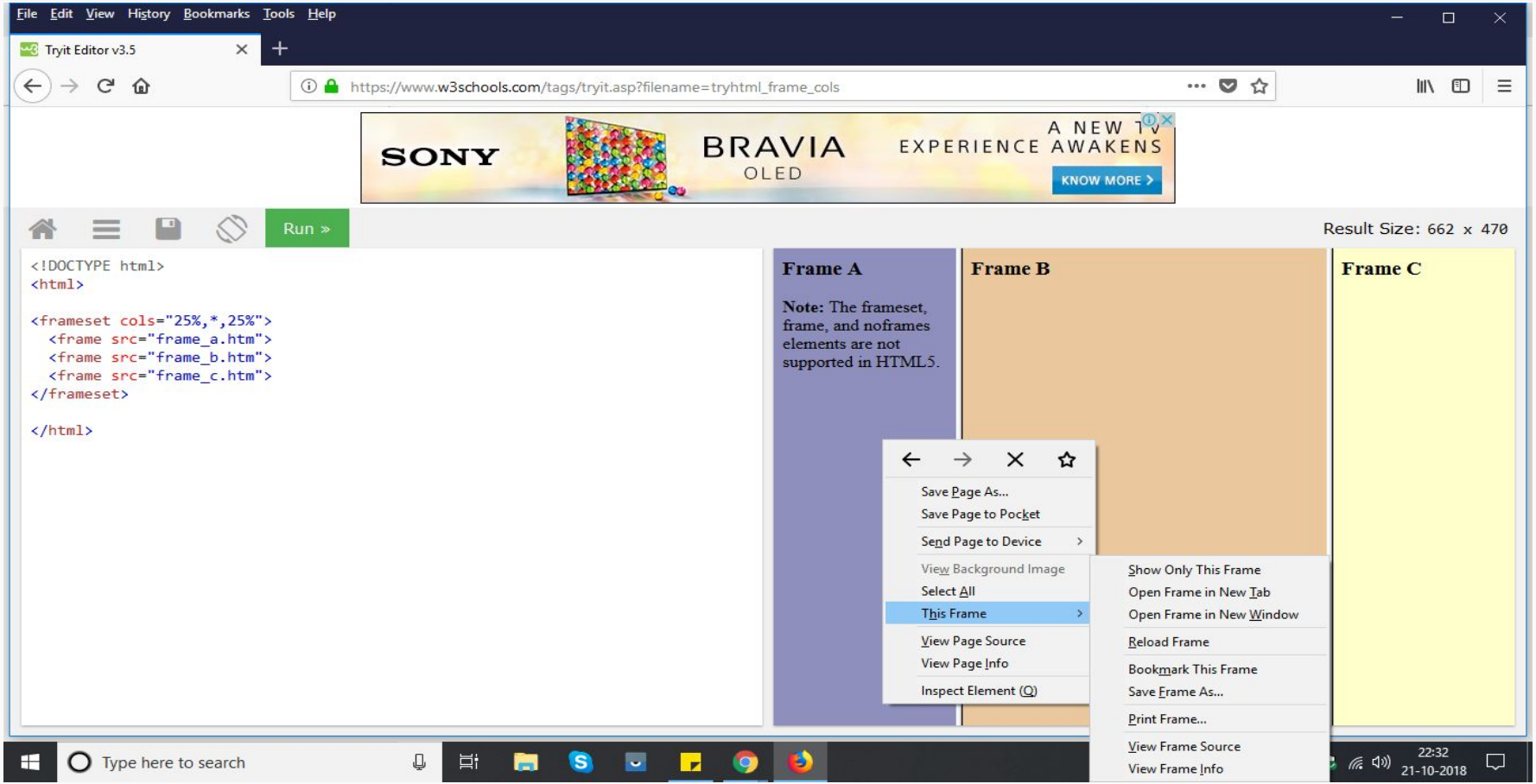
To work with the frames you need to switch to frame first.

In Selenium to work with Frames, we have different ways to handle frame depending on the need.

On Chrome : View frame source option get enable when you mouse hover on element



On Firefox : This frame option get enable



```
<html>
<head><title>Welcome to         Selenium iframes Tutorial</title>
</head>
<body>
<div>
       <iframe id="frame1">
            <iframe id="frame2">
              <body>
               <input type="text" id="input2">UserName</input>
             </body >
            </iframe>

            <body>
               <input type="text" id="input1">Password</input>
            </body >
       </iframe>

            <body>
               <button name="btnG">OK<button>
            </body >
</div>
</body>
</html>
```

Switch to Frames by Name or ID

Select a frame by its name or ID. Frames located by matching name attributes are always given precedence over those matched by ID.

Syntax - driver.switchTo().frame(String name/id);

```
driver.switchTo().frame("iframe1");
driver.quit();
```

Throws: NoSuchFrameException - If the frame is not found

Switch to Frame by WebElement

we can switch to an iFrame by simply passing the iFrame WebElement to the driver.switchTo().frame() command.

First find the iFrame element using any of the locator strategies and then passing it to switchTo command.

```
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(iframeElement);
driver.quit();
```

Throws: NoSuchFrameException - If the given element is neither an iframe nor a frame element.

Switch to Frames by Index

Index of an iFrame is the position at which it occurs in the HTML page.

Select a frame by its (zero-based) index. That is, if a page has multiple frames (more than 1), the first frame would be at index "0", the second at index "1" and so on.

Once the frame is selected or navigated , all subsequent calls on the WebDriver interface are made to that frame. i.e the driver focus will be now on the frame.

What ever operations we try to perform on pages will not work and throws element not found as we navigated / switched to Frame.

Syntax - driver.switchTo().frame(int arg0);

```
driver.switchTo().frame(0);
driver.quit();
```

Throws: NoSuchFrameException - If the frame is not found.

Switch to Frames by Index

Sometimes when there are multiple Frames (Frame in side a frame), we need to first switch to the parent frame and then we need to switch to the child frame.

In the below HTML, you can see we have frame2 inside frame1 and web elements inside the frame2.

```
//Switch to child frame
driver.switchTo().frame("frame1").switchTo().frame("frame2");
driver.findElement(By.id("input2")).sendKeys("Username");
driver.findElement(By.id("input1")).sendKeys("Password");

driver.findElement(By.cssSelector("button[name='btnG']")).click();

//switch to parent frame
driver.switchTo().frame("frame1");
```

Switching back to Main page from Frame

After working with the frames, main important is to come back to the web page.

if we don't switch back to the default page, driver will throw an exception.

Once you are done with all the task in a particular iFrame we need to switch back to the main page using the switchTo().defaultContent().

```
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(0);

//Do all the required tasks in the frame 0
//Switch back to the main window
driver.switchTo().defaultContent();
driver.quit();
```