Automation Testing

Working With External Elements

A Day in the Life of an Automation Test Engineer

Joel, an Automation Test Engineer, wants to automate web applications.

For a Test Automation Engineer, there is an interaction with many external elements in a web application like iFrames, alerts, windows, pop-ups, etc.

To know about it, let us go through this lesson.

Learning Objectives

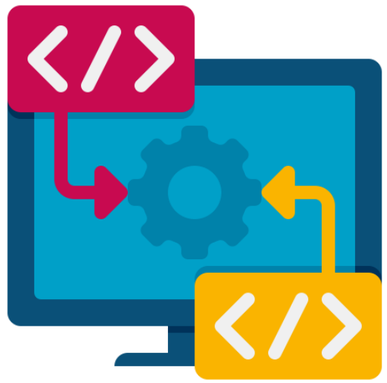
By the end of this lesson, you will be able to:

- Comprehend various handling elements
- Apply iFrames for HTML
- Comprehend alerts and their types
- Identify how to handle multiple windows
- Use Key-Press operations

Handling iFrames

Handling iFrames

The <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document.



Handling iFrames

Example:

```
<html>
<body>
 <div class="box">
        <iframe name="iframe1" id="IF1" height="50%"
width="50%" src="url"> </iframe>
  </div>
   <div class="box">
       <iframe name="iframe2" id="IF2" height="50%"
width="50%"  align="left" src="url"></iframe>
   </div>
    </body>
</html>
```

Handling iFrames

- Selenium cannot identify elements inside an iFrame directly.
- To interact with an iFrame, one must switch to the frame and then interact with it.
- Just like Selenium cannot switch to a frame, it also cannot come back to the main page or window from a frame.

To switch back from a frame to the main page Selenium WebDriver provides a method called **driver.switchToDefaultContent();**

Handling iFrames

Selenium WebDriver provides us following methods to switch to a frame:

Using Name or Id Tag (driver.switchToFrame(String nameOrId))

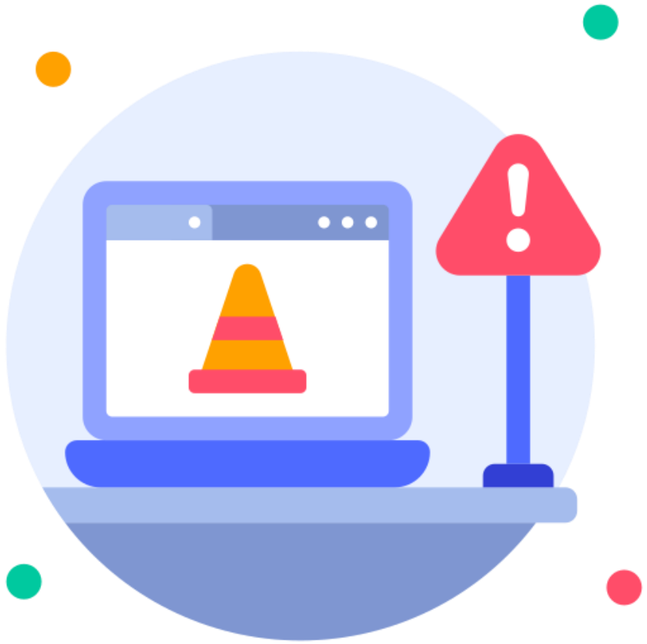
Using Index (driver.switchToFrame(int frameIndex))

Using WebElement (driver.switchToFrame(WebElement element))

Handling Alerts

Handling Alerts

Alerts are messages or notifications box, that notifies the user about some information or asks for permission to perform a certain kind of operation.

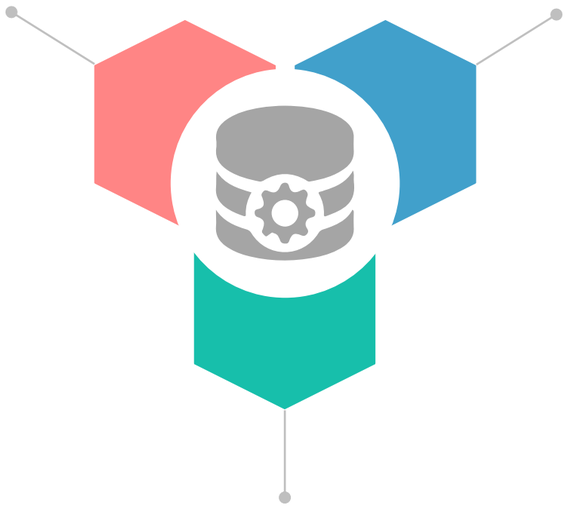


Handling Alerts

There are three types of Alerts:

Simple Alert

It is used to notify a simple warning message with an ‘OK’ button.



Confirmation Alert

It is used for the confirmation of some tasks.

Prompt Alert

It is used to ask the user to input the required information to complete the task.

Handling Alerts

Selenium WebDriver provides us following methods to handle alerts:

Accept an alert (driver.switchToAlert().accept()):
This method is used to accept an alert.

Reject/Dismiss an alert (driver.switchToAlert().dismiss()):
This method is used to reject/dismiss an alert.

Get Text from an alert (driver.switchToAlert().getText()):
This method is used to get text from an alert. This method returns a string value.

Handling Pop-Ups

Pop-Ups in Selenium

Pop-up is a window that displays or pops up on the screen due to interaction activity.

Web Elements on a web pop-up can be handled as normal web elements. Selenium can easily interact with these pop-ups.

Pop-Ups in Selenium

Some of the most common pop-ups are as follows :

Web Pop-ups

These pop-ups show up within a separate browser window.

JS Alerts

These are the native browser alerts.

Handling Multiple Windows

Handling Multiple Windows

An example:

When you are working on a webpage, and there is a link on the page, or when you click on the link, a new window or a new tab opens.

Unlike your browser, Selenium cannot handle this situation automatically, but you can programmatically write the code to switch to a new window, interact with the new elements, and switch back to the main page.

The main page can be referred to as the “Parent Window”, and the second window is referred to as the "Child Window".

Handling Multiple Windows

Selenium WebDriver provides us following method to handle or work with multiple windows:

Get the Window handle (driver.getWindowHandle()):

To get the window handle of the currently active window

Get all Window handles (driver.getAllWindowHandles()):

To get the window handles of all the opened windows

Switch to Windows (driver.switchToWindows(String windowHandle):

To switch to the window whose identifier is passed as an argument

Key-Press Operations

Key Press Operation

Selenium WebDriver provides methods like sendKey(), click() to perform user actions.
Some advanced keyboard events like:

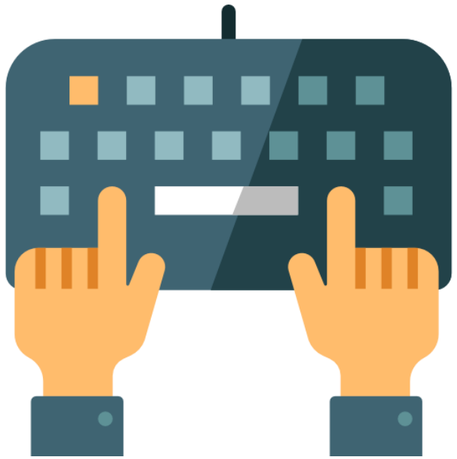
Invoke keyboard interactions by passing key combinations, e.g., CTRL + SHIFT, CTRL + A, etc.

Invoke typical keyboard-related interactions, e.g., Key Up, Key Down, etc.

Invoke actions on the browser instance using Function (F) keys, e.g., F5 to refresh the current browser page, etc.

Commonly Used Keyboard Events

Keyboard events can appropriately be used when Selenium testing is performed on the test web page or web application



Commonly Used Keyboard Events

Shown below are some of the commonly used keyboard events provided by the ActionChains class:

Action	Arguments	Description
send_keys(**keys_to_send)*	• *keys_to_send* – The keys to send. • Modifier keys constants can be found in the *Keys* class.	It sends keys to the element that is currently in focus.
key_down(*value*, *element=None*)	• *value* – Modifier key to send. • *element* – It is an optional argument. It represents the element on which key need to be sent. If it is not specified, i.e., None, the key is sent to the currently focused element.	It sends a key press without performing the release. It should only be used with modifier keys like Control, Alt, and Shift.

Commonly Used Keyboard Events

Action	Arguments	Description
key_up(*value*, *element=None*)	• *value* – Modifier key to send. • *element* – It is an optional argument. It represents the element on which keys need to be sent. If it is not specified, i.e., None, the key is sent to the currently focused element.	It releases a key. It should only be used with modifier keys like Control, Alt, and Shift.
perform	None	It performs the chain of actions stored in the ActionChains object.

Key Takeaways

- iFrames or inline frames are used for embedding other pages inside main web page.
- Selenium provides various switch methods to interact with elements inside a frame, switch to an alert, or switch to multiple windows.
- Selenium provides an Actions class to handle Keyword interactions.
- Selenium WebDriver provides methods like sendKey(), click() to perform user actions.

Thank You