EXCEPTIONS IN SOFTWARE TESTING

Document By Suraj G.

What are the Common Selenium Exceptions and How to Handle Them?

The benefits of automation testing are never-ending. However, errors and exceptions are something that a testing environment must deal with daily.

What are Errors and Exceptions?

To get a good idea of what errors and exceptions are, here is a small analogy.

Imagine you are in Pune, and you have an important meeting in Mumbai. Your friend suggests that you drive your way to the meeting. However, halfway through the journey, there is a weather alert. The weather department predicts a tornado and all roads leading Mumbai are closed. You have no choice but to cancel your meeting. Similarly, in programming errors occur during run-time, and it is impossible to recover from them.

Now imagine another scenario where there is no tornado, but your car is punctured. Thankfully you have a Stepney (a spare tire), and you replace it and attend your meeting on time. Exceptions are like small issues that can be overcome.

To sum up, errors are created by the testing environment and difficult to handle. Exceptions, on the other hand, are thrown by the application itself and can be handled.

Exception Handling in Selenium

Selenium is a web automation framework that allows testing applications against different browsers. During automation in Selenium, the testing team must handle multiple exceptions.

Exceptions are faults or disruptions that occur during the execution of a program/application. Exception handling is crucial for maintaining the natural or normal flow of the application.

Selenium exceptions can be broadly categorized into two types: Checked and Unchecked Exceptions.

Checked exceptions are handled during the coding process itself. Unchecked exceptions occur during run-time and can have a much greater impact on the application flow. We have compiled some of the most common selenium exceptions along with the various ways to handle them.

Most Common Selenium Exceptions

- NoSuchWindowException
- NoSuchFrameException
- NoSuchElementException
- NoAlertPresentException
- InvalidSelectorException
- TimeoutException
- ElementNotVisibleException
- ElementNotSelectableException
- NoSuchSessionException
- StaleElementReferenceException

1. NoSuchWindowException

One of the most frequent exceptions in Selenium Webdriver, NoSuchWindowException occurs if the current list of windows is not updated. The previous window does not exist, and you can't switch to it.

To handle this exception, use webdriver methods called "driver.getWindowHandles()."

2. NoSuchFrameException

Similar to NoSuchWindowException, the NoSuchFrameException occurs when switching between multiple frames is not possible.

The solution used for handling NoSuchWindowException must ideally work for this exception too.

3. NoSuchElementException

Happens when the locators are unable to find or access elements on the web page or application. Ideally, the exception occurs due to the use of incorrect element locators in the findElement(By, by) method.

To handle this exception, use the wait command. Use Try/Catch to ensure that the program flow is interrupted if the wait command doesn't help.

4. NoAlertPresentException

Happens when the user is trying to you switch to an alert which is not present. In simple terms, it means that the test is too quick and is trying to find an alert that has not yet been opened by the browser.

To handle or simply avoid this exception, use explicit or fluent wait in all events where an alert is expected.

5. InvalidSelectorException

This exception occurs due to an incorrect selector. More often than not, the selector syntax is wrong.

To avoid this exception, first, check the locator syntax. If it is incorrect, make sure the locator is syntactically correct.

6. TimeoutException

This exception is thrown if the command did not execute or complete within wait time. As already mentioned, waits are used to avoid NoSuchElementException. However, TimeoutException will be thrown after the page components fail to load within wait time.

Avoiding this exception is simple. All one needs to do is to calculate the average page load time and adjust the wait time accordingly.

7. ElementNotVisibleException

This exception occurs when the WebDriver tries to find an element that is hidden or invisible. To handle this exception, it is essential that the exact reason is identified, which can be due to nested web elements or overlapping web elements.

In the first case, you have to locate and correct the specific element. In the second case, use explicit wait.

8. ElementNotSelectableException

This exception belongs to the same class as InvalidElementStateException. In such exceptions, the element is present on the web page, but the element cannot be selected.

To handle this exception, the wait command must be used.

9. NoSuchSessionException

As the name suggests, the exception is thrown if a method is called post the browser is closed. Other reasons for this exception include a browser crash.

To avoid this handle, ensure that your browser is updated and stable.

10. StaleElementReferenceException

This exception occurs when the web element is no longer part of the web page. The element may have been part of the source code, but it is no longer part of the window. There can be multiple reasons for this exception. It can occur either from a switch to a different page, the element is no longer part of DOM or due to frame/window switch

To handle this exception, you can either use Dynamic Xpath for handling DOM operations or try to use the Page Factory Model or try to access the element in loops before throwing the exception.