[**https://github.com/pavanoltraining/inetbankingV1**](https://github.com/pavanoltraining/inetbankingV1)

[**https://www.pavanonlinetrainings.com/p/selenium-with-java-course-contents.html**](https://www.pavanonlinetrainings.com/p/selenium-with-java-course-contents.html)

[**https://thetestingacademy.com/**](https://thetestingacademy.com/)

**Demo Web site :** [**https://automationintesting.online/**](https://automationintesting.online/)

**For Locator :**

[**https://rahulshettyacademy.com/locatorspractice/**](https://rahulshettyacademy.com/locatorspractice/)

[**https://rahulshettyacademy.com/AutomationPractice/**](https://rahulshettyacademy.com/AutomationPractice/)

**For Drop Down :**

[**https://rahulshettyacademy.com/dropdownsPractise/**](https://rahulshettyacademy.com/dropdownsPractise/)

**Assignments Practise :** [**https://rahulshettyacademy.com/angularpractice/**](https://rahulshettyacademy.com/angularpractice/)

**https://rahulshettyacademy.com/seleniumPractise/#/**

**ElementNotSelectableException**

**Cause: This exception occurs when trying to select an option in a dropdown that is not selectable. This typically happens in multi-select dropdowns or when using methods that are not appropriate for the element type.**

**Example:**

**java**

**Copy code**

**WebElement dropdownElement = driver.findElement(By.id("dropdown-id"));**

**dropdownElement.click(); // If dropdown is not actually a standard <select> tag**

**Solution:**

* **Ensure you are using the Select class for standard <select> dropdowns.**
* **For custom dropdowns, use appropriate methods to interact with options.**

**General Tips for Handling Dropdowns:**

* **Use Explicit Waits: Always use explicit waits when dealing with elements that may not be immediately available or visible.**
* **Verify Locators: Ensure your locators are correct and unique to avoid NoSuchElementException.**
* **Handle Dynamic Changes: Be prepared to re-locate elements if the page updates dynamically or use JavaScript for complex interactions.**

**XPath Tutorial :** [**https://www.softwaretestinghelp.com/xpath-axes-tutorial/**](https://www.softwaretestinghelp.com/xpath-axes-tutorial/)

**Gmail account :** [**sidpatiltesting@gmail.com**](mailto:sidpatiltesting@gmail.com)

**Gmail Password : selenium\_testing**

**Locator – XPath Axes :**

* Locating a Parent Element:

The parent axis contains the parent of the context node .

Every context element has only one parent element except root element(html).

**Syntax :**

**//<knownXpath>/parent::\*or**

**//<knownXpath>/parent::elementName**

**//<knownXpath>/..**

XPath of the known element : //input[@id=’textUsername’]

**Example :**

<

Selenium WebDriver exceptions are critical for diagnosing and handling issues during test automation. They indicate various problems that can arise while interacting with web elements or navigating through web pages. Below is a comprehensive list of common Selenium exceptions and their typical causes:

**1. NoSuchElementException**

* **Cause**: The WebDriver was unable to find an element using the provided selector.
* **Solution**: Check the selector, ensure the element is present on the page, and use waits to handle dynamic content.

java

Copy code

WebElement element = driver.findElement(By.id("nonExistentId")); // Throws NoSuchElementException

**2. NoSuchFrameException**

* **Cause**: The WebDriver was unable to switch to a frame that does not exist.
* **Solution**: Ensure the frame exists, and switch to it properly using driver.switchTo().frame().

java

Copy code

driver.switchTo().frame("frameId"); // Throws NoSuchFrameException if the frame does not exist

**3. NoSuchWindowException**

* **Cause**: The WebDriver was unable to switch to a window that does not exist.
* **Solution**: Ensure the window handle is valid and that the window is not closed before the switch.

java

Copy code

driver.switchTo().window("nonExistentWindowHandle"); // Throws NoSuchWindowException

**4. NoSuchAttributeException**

* **Cause**: The WebDriver is unable to find an attribute that does not exist on an element.
* **Solution**: Check if the attribute is correct and present on the element.

java

Copy code

String attribute = driver.findElement(By.id("elementId")).getAttribute("nonExistentAttribute"); // May cause NoSuchAttributeException

**5. StaleElementReferenceException**

* **Cause**: The WebDriver is trying to interact with an element that is no longer present in the DOM.
* **Solution**: Refresh the element reference or use explicit waits to ensure the element is available.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

// Element is removed or updated in the DOM

element.click(); // Throws StaleElementReferenceException

**6. ElementNotInteractableException**

* **Cause**: The WebDriver finds the element but is unable to interact with it because it is not in a state that allows interaction.
* **Solution**: Ensure the element is visible, enabled, and not covered by other elements.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

element.click(); // Throws ElementNotInteractableException if the element cannot be interacted with

**7. ElementNotSelectableException**

* **Cause**: The WebDriver tries to select an element that cannot be selected.
* **Solution**: Verify that the element can be selected, such as by checking if it is a selectable option.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

element.click(); // Throws ElementNotSelectableException if the element cannot be selected

**8. TimeoutException**

* **Cause**: A command did not complete in the specified time.
* **Solution**: Increase the wait time or improve the code to handle the timing issues.

java

Copy code

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(5));

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("elementId"))); // May throw TimeoutException

**9. InvalidElementStateException**

* **Cause**: The WebDriver attempts to interact with an element in an invalid state (e.g., trying to send keys to a disabled input field).
* **Solution**: Ensure the element is in the correct state to perform the desired action.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

element.sendKeys("text"); // Throws InvalidElementStateException if the element is not in a valid state

**10. WebDriverException**

* **Cause**: A generic error that can occur in various situations, often related to WebDriver issues.
* **Solution**: Check the specific message and stack trace for more details about the problem.

java

Copy code

driver.quit(); // May throw WebDriverException if the driver is not properly initialized or has issues

**11. JavascriptException**

* **Cause**: An error occurred while executing JavaScript code through the WebDriver.
* **Solution**: Check the JavaScript code being executed and ensure it is correct.

java

Copy code

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("return someInvalidJavaScript();"); // Throws JavascriptException if the script is invalid

**12. NoSuchSessionException**

* **Cause**: The WebDriver session is no longer valid (e.g., the browser was closed or the session was terminated).
* **Solution**: Ensure the WebDriver session is properly managed and the browser remains open during interactions.

java

Copy code

driver.get("https://www.example.com");

driver.quit(); // Subsequent operations on this driver instance may throw NoSuchSessionException

**13. RemoteDriverException**

* **Cause**: An issue occurred when interacting with a remote WebDriver instance.
* **Solution**: Ensure the remote WebDriver server is running and properly configured.

java

Copy code

WebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), new ChromeOptions());

driver.get("https://www.example.com"); // May throw RemoteDriverException if there's a problem with the remote server

**14. ElementNotVisibleException**

* **Cause**: The WebDriver is unable to interact with an element that is not visible (e.g., hidden by CSS).
* **Solution**: Ensure the element is visible and can be interacted with.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

element.click(); // Throws ElementNotVisibleException if the element is not visible

**15. ElementClickInterceptedException**

* **Cause**: The WebDriver tries to click on an element that is obstructed by another element.
* **Solution**: Use JavaScript to click on the element if necessary, or handle the obstruction.

java

Copy code

WebElement element = driver.findElement(By.id("elementId"));

element.click(); // Throws ElementClickInterceptedException if the element is covered by another element

By understanding these exception types and their causes, you can better diagnose and resolve issues that arise during your Selenium test automation.

The ElementClickInterceptedException in Selenium occurs when an element that you are trying to click is obstructed by another element. This can happen for several reasons, such as:

* **Overlapping elements:** Another element (like a modal, tooltip, or loading spinner) might be covering the element you want to click.
* **Dynamic content:** The page may be updating or loading new content, which can cause elements to move or overlap.

Here's how you might encounter this exception in Java:

java

Copy code

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.ElementClickInterceptedException;

public class ClickExample {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

WebDriver driver = new ChromeDriver();

try {

driver.get("http://example.com");

WebElement elementToClick = driver.findElement(By.id("clickableElement"));

// Attempt to click on the element

elementToClick.click();

} catch (ElementClickInterceptedException e) {

System.out.println("ElementClickInterceptedException caught: " + e.getMessage());

} finally {

driver.quit();

}

}

}

**Handling ElementClickInterceptedException**

Here are some strategies to handle this exception:

1. **Wait for the Element to be Clickable:** Use WebDriverWait to wait until the element is clickable. This ensures that any dynamic content or overlays have had time to disappear or become interactive.

java

Copy code

import org.openqa.selenium.support.ui.WebDriverWait;

import org.openqa.selenium.support.ui.ExpectedConditions;

WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement elementToClick = wait.until(ExpectedConditions.elementToBeClickable(By.id("clickableElement")));

elementToClick.click();

1. **Scroll to the Element:** Sometimes scrolling to the element can help if the element is not fully in view.

java

Copy code

import org.openqa.selenium.JavascriptExecutor;

JavascriptExecutor js = (JavascriptExecutor) driver;

WebElement elementToClick = driver.findElement(By.id("clickableElement"));

js.executeScript("arguments[0].scrollIntoView(true);", elementToClick);

elementToClick.click();

1. **Handle Overlapping Elements:** Check for any overlapping elements or modals and handle them. For example, if a modal is blocking the element, you might need to close the modal first.

java

Copy code

// Check for a modal and close it if present

try {

WebElement modalCloseButton = driver.findElement(By.className("modal-close"));

if (modalCloseButton.isDisplayed()) {

modalCloseButton.click();

}

} catch (NoSuchElementException e) {

// Modal is not present, continue

}

1. **Retry Clicking:** Implement a retry mechanism to attempt clicking the element multiple times if the issue is intermittent.

java

Copy code

import java.util.concurrent.TimeUnit;

for (int i = 0; i < 3; i++) {

try {

WebElement elementToClick = driver.findElement(By.id("clickableElement"));

elementToClick.click();

break; // Exit loop if successful

} catch (ElementClickInterceptedException e) {

// Wait for a moment before retrying

TimeUnit.SECONDS.sleep(1);

}

}

By using these techniques, you can often resolve the issue of ElementClickInterceptedException and make your Selenium tests more robust.

//div[@class='r-1awozwy r-19m6qjp r-y47klf r-1loqt21 r-eu3ka r-1777fci r-1otgn73 r-1i6wzkk r-lrvibr r-1aockid css-1dbjc4n']//\*[name()='svg'] Top of Form

div[class='css-1dbjc4n r-k8qxaj r-d9fdf6'] div:nth-child(1) div:nth-child(2) div:nth-child(3) svg explain this cssSelectors in selenium java

**ChatGPT said:**

ChatGPT

The CSS selector div[class='css-1dbjc4n r-k8qxaj r-d9fdf6'] div:nth-child(1) div:nth-child(2) div:nth-child(3) svg is used to target a specific SVG element within a nested structure of <div> elements based on their class names and child positions. Let's break down this selector step-by-step:

**Breakdown of the CSS Selector**

1. **div[class='css-1dbjc4n r-k8qxaj r-d9fdf6']**:
   * This part selects a <div> element with a class attribute value that exactly matches 'css-1dbjc4n r-k8qxaj r-d9fdf6'.
   * Note that the class attribute value must match exactly, including spaces.
2. **div:nth-child(1)**:
   * This selects the first child <div> element of the previously selected <div>.
   * :nth-child(1) targets the very first child, regardless of its type.
3. **div:nth-child(2)**:
   * This selects the second child <div> element within the first <div> selected by the previous step.
   * :nth-child(2) specifically targets the second child.
4. **div:nth-child(3)**:
   * This selects the third child <div> element within the second <div> selected by the previous step.
   * :nth-child(3) targets the third child.
5. **svg**:
   * Finally, this selects the <svg> element that is a descendant of the third <div> element selected by the previous steps.

**Example Structure**

To visualize this selector, imagine the following HTML structure:

html

Copy code

<div class="css-1dbjc4n r-k8qxaj r-d9fdf6">

<div> <!-- First child div -->

<div> <!-- Second child div -->

<div> <!-- Third child div -->

<svg> <!-- Target SVG -->

<!-- SVG content -->

</svg>

</div>

</div>

</div>

</div>

**Explanation**

* **div[class='css-1dbjc4n r-k8qxaj r-d9fdf6']**: Start by selecting the outermost <div> with the specific class.
* **div:nth-child(1)**: From that <div>, select its first child, which is another <div>.
* **div:nth-child(2)**: Within that <div>, select its second child <div>.
* **div:nth-child(3)**: Within that second child <div>, select its third child <div>.
* **svg**: Finally, select the <svg> element within that third <div>.

Bottom of Form