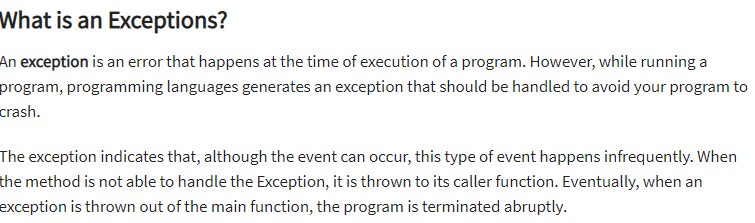
**Selenium Exception Handling (Common Exceptions List)**



## Common Exceptions in Selenium Web driver

|  |  |
| --- | --- |
| **Exception name** | **Description** |
| ElementNotVisibleException | This type of Selenium exception occurs when an existing element in  DOM has a feature set as hidden. |
| ElementNotSelectableException | This Selenium exception occurs when an element is presented in the DOM,  but you can be able to select. Therefore, it is not possible to interact. |
| NoSuchElementException | This Exception occurs if an element could not be found. |
| NoSuchFrameException | This Exception occurs if the frame target to be switched to does not exist. |
| NoAlertPresentException | This Exception occurs when you switch to no presented alert. |
| NoSuchWindowException | This Exception occurs if the window target to be switch does not exist. |
| StaleElementReferenceException | This Selenium exception occurs happens when the web element is detached  from the current DOM. |
| SessionNotFoundException | The WebDriver is acting after you quit the browser. |
| TimeoutException | Thrown when there is not enough time for a command to be completed.  For Example, the element searched wasn't found in the specified time. |
| WebDriverException | This Exception takes place when the WebDriver is acting right after you  close the browser. |
| ConnectionClosedException | This type of Exception takes place when there is a disconnection in the driver. |
| ElementClickInterceptedException | The command may not be completed as the element receiving the events is  concealing the element which was requested clicked. |
| ElementNotInteractableException | This Selenium exception is thrown when any element is presented in the DOM.  However, it is impossible to interact with such an element. |
| ErrorInResponseException | This happens while interacting with the Firefox extension or the remote driver  server. |
| ErrorHandler.UnknownServerException | Exception is used as a placeholder in case if the server returns an error without  a stack trace. |
| ImeActivationFailedException | This expectation will occur when IME engine activation has failed. |
| ImeNotAvailableException | It takes place when IME support is unavailable. |
| InsecureCertificateException | Navigation made the user agent to hit a certificate warning.  This can cause by an invalid or expired TLS certificate. |
| InvalidArgumentException | It occurs when an argument does not belong to the expected type. |
| InvalidCookieDomainException | This happens when you try to add a cookie under a different domain  instead of current URL. |
| InvalidCoordinatesException | This type of Exception matches an interacting operation that is not valid. |
| InvalidElementStateExceptio | It occurs when command can't be finished when the element is invalid. |
| InvalidSessionIdException | This Exception took place when the given session ID is not included in the  list of active sessions. It means the session does not exist or is inactive either. |
| InvalidSwitchToTargetException | This occurs when the frame or window target to be switched does not exist. |
| JavascriptException | This issue occurs while executing JavaScript given by the user. |
| JsonException | It occurs when you afford to get the session when the session is not created. |
| NoSuchAttributeException | This kind of Exception occurs when the attribute of an element could not be found. |
| MoveTargetOutOfBoundsException | It takes place if the target provided to the ActionChains move() methodology  is not valid. For Example, out of the document. |
| NoSuchContextException | ContextAware does mobile device testing. |
| NoSuchCookieException | This Exception occurs when no cookie matching with the given pathname found  for all the associated cookies of the currently browsing document. |
| NotFoundException | This Exception is a subclass of WebDriverException. This will occur when an  element on the DOM does not exist. |
| RemoteDriverServerException | This Selenium exception is thrown when the server is not responding because  of the problem that the capabilities described are not proper. |
| ScreenshotException | It is not possible to capture a screen. |
| SessionNotCreatedException | It happens when a new session could not be successfully created. |
| UnableToSetCookieException | This occurs if a driver is unable to set a cookie. |
| UnexpectedTagNameException | Happens if a support class did not get a web element as expected. |
| UnhandledAlertException | This expectation occurs when there is an alert, but WebDriver is not able to  perform Alert operation. |
| UnexpectedAlertPresentException | It occurs when there is the appearance of an unexpected alert. |
| UnknownMethodException | This Exception happens when the requested command matches with a known  URL but and not matching with a methodology for a specific URL. |
| UnreachableBrowserException | This Exception occurs only when the browser is not able to be opened or crashed  because of some reason. |
| UnsupportedCommandException | This occurs when remote WebDriver does n't send valid commands as expected. |

## Handling Selenium Exceptions

**Try-catch**: This method can catch Exceptions, which uses a combination of the try and catch keywords. Try command indicates the start of the block, and **Catch** is placed at the end of the try block, which helps to resolve the Exception.

try

{

// Code

} catch (Exception e) {

// Code for Handling exception

}

**Multiple catch blocks**: There are various types of Exceptions, and you can expect more than one exception from a single block of code. Multiple catches help you to handle every type of Exception separately with a separate block of code. It can be used for more than two catch blocks, and there is no limitation on the number of catch blocks.

try

{

//Code

} catch (ExceptionType1 e1) {

//Code for Handling Exception 1

} catch (ExceptionType2 e2) {

//Code for Handling Exception 2

}

**Throw**: When you want to generate an Exception, the Throw keyword is used to throw Exception to handle it in the run time. When you are throwing an Exception without handling it, then they need to use Throw keyword.

public static void anyFunction() throws Exception{

try {

// write your code here

} Catch (Exception b) {

// Do whatever you want to perform

// Throw the Exception back to the system

throw(b);

}

}

**Multiple Exceptions**: You can mention various Exceptions in the throws clause.

public static void anyFunction() throws ExceptionType1, ExceptionType2{

try {

// write your code here

} catch (ExceptionType1 e1) {

// Code to handle exception 1

} catch (ExceptionType1 e2) {

// Code to handle exception 2

}

**Finally**: The Final keyword is used to create a block of code under the try block. It is the final code that helps you to executes irrespective of the occurrence of an exception

try {

//Code

} catch (ExceptionType1 e1) {

//Catch block

} catch (ExceptionType2 e2) {

//Catch block

} catch (ExceptionType3 e3) {

//Catch block

} finally {

//The finally block always executes.

}

