**Handle Exception in SoapUI Groovy Scripts**

### What is an exception?

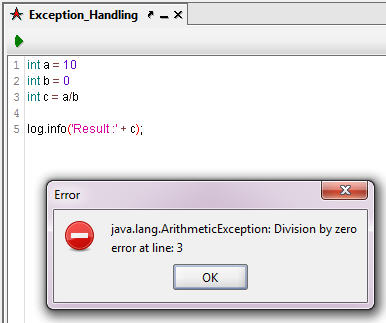
An[exception](http://en.wikipedia.org/wiki/Exception_handling) is an error that is encountered during the execution of a program. It can happen due to many reasons such as invalid data, network connection loss, trying open files that are unavailable, accessing invalid class, memory leakage i.e. forcing the system to work with huge amount of data, database server being unresponsive, etc. These errors may be because of users, developers or hardware resources.

Internally, when an exception is encountered during the execution, SoapUI will try to find the handler. Handler is the block that contains the code to catch the exception.

**Exceptions are categorized two types:**

1. [Runtime exception](http://docs.oracle.com/javase/7/docs/api/java/lang/RuntimeException.html?is-external=true)
2. Compile time exception – not applicable to SoapUI as it does not have an explicit compiler

Look at the following screenshot which shows us a runtime exception for invalid code. In the below script we tried to divide an integer by 0. In the error dialog it can be seen that the exception is raised from java libraries and the error message is **Division by zero**.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/06/Exception-Handling-in-Groovy-Scripts-1.jpg)

We can catch this exception during the execution and handle it programmatically. Before that we will see some of the important keywords that are used in the java exception concepts. Some of the keywords can be used in groovy script also. They are:

* **Throw** – This keyword help us to throw an exception manually i.e. to throw user defined exceptions
* **Throws** – It is used to call the pre-defined exceptions from the method definition. So it will catch the exception if there is any runtime error found during the execution.
* **Try** and **Catch** – “try” keyword is used with “catch” keyword. If we can predict the portion of the program where the exception can arise during the execution, we can use “try” block in that place. At end of the “try” block, “catch” block should start to catch exception. Inside the catch block, we have to write the handler to handle the exception.
* **Finally** – This is the default and optional block in the exception structure. If we need any statements to be executed at end of the program, like cleaning unused objects, closing connections etc. that can be done inside this block.

**The following is the general structure of an exception:**

try  
{  
<code to be executed>  
}  
catch <exception name>  
{  
<exception handler>  
}  
finally  
{  
<default statements>  
}

Now let us implement the exception handler in the sample code which we have already seen in the screenshot.

Add new test suite under the **GlobalWeather** project. Then add a test case and groovy script test step under the test step. In the script editor, enter the following script.

|  |  |  |
| --- | --- | --- |
| 1 | // initializing the variables | |
| 2 | int a = 10; |

|  |  |  |
| --- | --- | --- |
| 3 | int b = 0; | |
| 4 |  |

|  |  |  |
| --- | --- | --- |
| 5 | // try, catch block | |
| 6 | try |

|  |  |
| --- | --- |
| 7 | { |
| 8 | // Dividing a number by zero | |

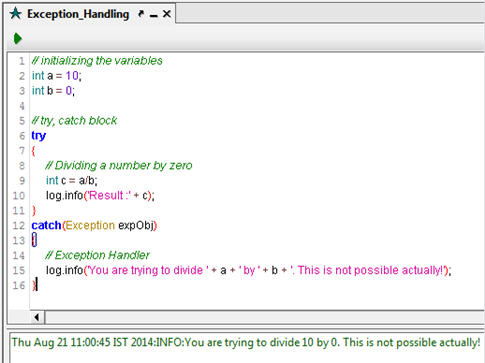
|  |  |  |
| --- | --- | --- |
| 9 | int c = a/b; | |
| 10 | | log.info('Result :' + c); | |

|  |  |
| --- | --- |
| 11 | } |
| 12 | catch(Exception expObj) | |

|  |  |
| --- | --- |
| 13 | { |
| 14 | // Exception Handler | |

|  |  |  |
| --- | --- | --- |
| 15 | log.info('You are trying to divide ' + a + ' by ' + b + '. This is not possible actually!'); | |
| 16 | } |

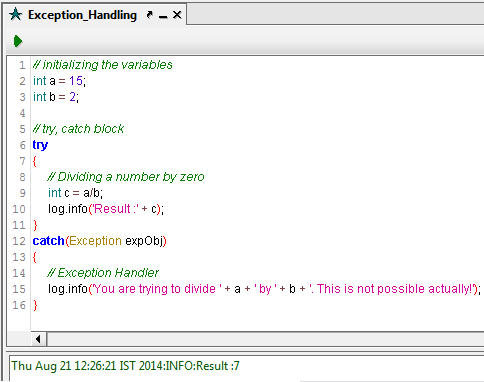
The above script produces the following result as shown in the screenshot.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/06/Exception-Handling-in-Groovy-Scripts-2.jpg)

As we discussed earlier, we tried to divide “A” “B” which is zero. So the ‘catch’ block gets executed and shows the user defined message in the log. See that in the “catch” statement, we have used **Exception** class which is the super class in Java for all the built-in exceptions. All pre-defined exception classes are inherited from **Exception** class. To handle unpredictable runtime exceptions, we can use **Exception** class in the “catch” block.

------------

Let us now modify the above script to get the required result. See the following screenshot:

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/06/Exception-Handling-in-Groovy-Scripts-3.jpg)

Let us now try this in our regular web services testing. In the following script, we did not use try-catch block so we will get runtime exception.

|  |  |
| --- | --- |
| 1 | // Initializing array with 5 elements |
| 2 | String[] countryNames = new String[5]; | |

|  |  |
| --- | --- |
| 3 |  |
| 4 | // Assigning values to the array | |

|  |  |
| --- | --- |
| 5 | countryNames[0] = 'India'; |
| 6 | countryNames[1] = 'Cyprus'; | |

|  |  |
| --- | --- |
| 7 | countryNames[2] = 'Canada'; |
| 8 | countryNames[3] = 'Austria'; | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | countryNames[4] = 'Mauritius'; | | |
| 10 | |  |

|  |  |  |
| --- | --- | --- |
| 11 | // Iterate the array elements and assign value to the global property | |
| 12 | for(int idx=0; idx<=5; idx++) |

|  |  |
| --- | --- |
| 13 | { |
| 14 | com.eviware.soapui.SoapUI.globalProperties.setPropertyValue( "CountryName", countryNames[idx]); | |

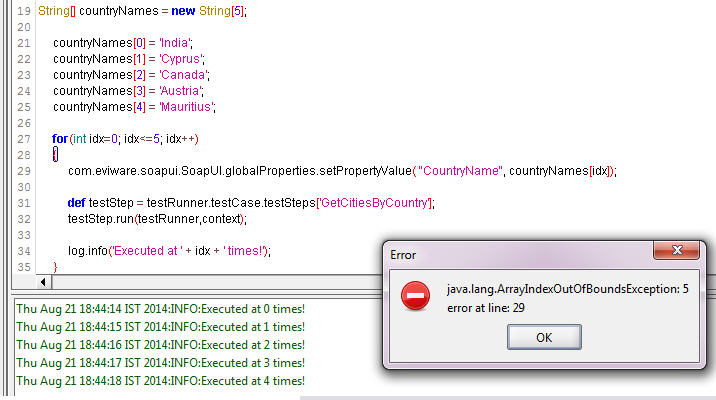
|  |  |
| --- | --- |
| 15 |  |
| 16 | def testStep = testRunner.testCase.testSteps['GetCitiesByCountry']; | |

|  |  |  |
| --- | --- | --- |
| 17 | testStep.run(testRunner,context); | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | log.info('Executed at ' + idx + ' times!'); | |
| 20 | } |

The above script will throw an exception called **ArrayIndexOutOfBoundsException** because the script is trying to access invalid array index i.e. 5 which is not available.

(Click image for enlarged view)

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/06/Exception-Handling-in-Groovy-Scripts-4.jpg)

As you can see in the above script, we have initialized “countryNames” array with the size of five. It accepts only five string value i.e. country names. Inside the iterative statements, we have checked as **idx < = 5**. So the loop will be iterating up to 6 times and it will try to search 6th element in the array. Since the value will not be there it throws a runtime exception.

To handle this scenario, let us modify the above script as below:

|  |  |  |
| --- | --- | --- |
| 1 | String[] countryNames = new String[5]; | |
| 2 | // Try block |

|  |  |  |
| --- | --- | --- |
| 3 | try | |
| 4 | { |

|  |  |
| --- | --- |
| 5 | countryNames[0] = 'India'; |
| 6 | countryNames[1] = 'Cyprus'; | |

|  |  |
| --- | --- |
| 7 | countryNames[2] = 'Canada'; |
| 8 | countryNames[3] = 'Austria'; | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | countryNames[4] = 'Mauritius'; | | |
| 10 | |  |

|  |  |  |
| --- | --- | --- |
| 11 | for(int idx=0; idx<=5; idx++) | |
| 12 | { |

|  |  |  |
| --- | --- | --- |
| 13 | com.eviware.soapui.SoapUI.globalProperties.setPropertyValue | |
| 14 | ( "CountryName", countryNames[idx]); |

|  |  |
| --- | --- |
| 15 |  |
| 16 | def testStep = testRunner.testCase.testSteps['GetCitiesByCountry']; | |

|  |  |  |
| --- | --- | --- |
| 17 | testStep.run(testRunner,context); | |
| 18 |  |

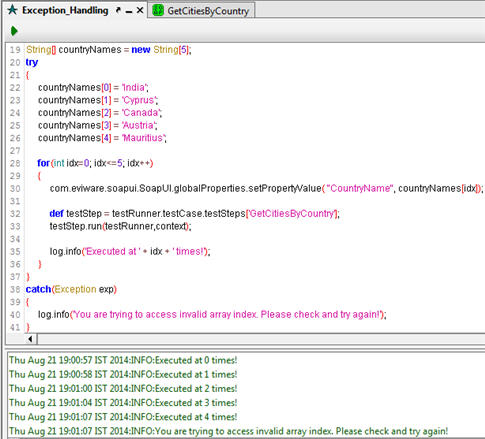
|  |  |  |
| --- | --- | --- |
| 19 | log.info('Executed at ' + idx + ' times!'); | |
| 20 | } |

|  |  |
| --- | --- |
| 21 | } |
| 22 | catch(Exception exp) // Catch the exception and displaying the message in the log | |

|  |  |
| --- | --- |
| 23 | { |
| 24 | log.info('You are trying to access invalid array index. Please check and try again!'); | |

|  |  |
| --- | --- |
| 25 | } |

Here is the result for the above script.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2015/06/Exception-Handling-in-Groovy-Scripts-5.jpg)

This is how we can handle runtime exception during our program execution.

**Note:** we can use **ArrayIndexOutOfBoundsException** in the “catch” block directly instead of using **Exception** class. If we put the exact exception name in the “catch” block”, it will catch only when the particular exception is thrown. If any other pre-defined exceptions thrown, catch block will be failed.

A good automation script should have proper exception handlers. Otherwise it will be difficult to monitor every moment of the execution.

As I have mentioned earlier, groovy script supports “throws” keyword to throw pre-defined exception to the caller.

See the below sample script to understand this concept:

<return-type> <method name> [ arguments / parameters ]***throws***<exception name>  
{  
                <statements to be executed>  
}

Here’s the sample code for the above skeleton.

|  |  |
| --- | --- |
| 1 | // Invoke Method |
| 2 | MethodWithThrowKeyword(); | |

|  |  |
| --- | --- |
| 3 |  |
| 4 | void MethodWithThrowKeyword() throwsArrayIndexOutOfBoundsException | |

|  |  |
| --- | --- |
| 5 | { |
| 6 | String[] countryNames = new String[5]; | |

|  |  |
| --- | --- |
| 7 |  |
| 8 | countryNames[0] = 'India'; | |

|  |  |  |
| --- | --- | --- |
| 9 | countryNames[1] = 'Cyprus'; | |
| 10 | | countryNames[2] = 'Canada'; | |

|  |  |
| --- | --- |
| 11 | countryNames[3] = 'Austria'; |
| 12 | countryNames[4] = 'Mauritius'; | |

|  |  |
| --- | --- |
| 13 |  |
| 14 | for(int idx=0; idx<=5; idx++) | |

|  |  |
| --- | --- |
| 15 | { |
| 16 | log.info('Country Names: ' + countryNames[idx]); | |

|  |  |  |
| --- | --- | --- |
| 17 | } | |
| 18 | } |

In the above script, the **ArrayIndexOutOfBoundsException** will be thrown to the called function. There we need to handle properly with try-catch block. Otherwise, an exception will be thrown by SoapUI.

### Conclusion:

Implementing exception handling in our regular web services related testing scripts, will be helpful for us to maintain the code and reduce manual intervention/monitoring by testers. We can use multiple try-catch blocks when required in the script.