Do...While

Sometimes it is not possible to determine in advance how many times a loop should be repeated. Here is an example:

A user queries a Check Status page multiple times until the server completes a slow, asynchronous transaction and redirects to a Result page. When recording such a test case, it is unknown how many times during the load test, each virtual user should query the Check Status page. It is necessary to use a **Do...While** (Conditional Loop) to configure such a scenario. Conditional loops have a condition that is checked at the end of the loop to determine if the loop should continue or exit. You can configure a loop to exit when the condition returns true or false.
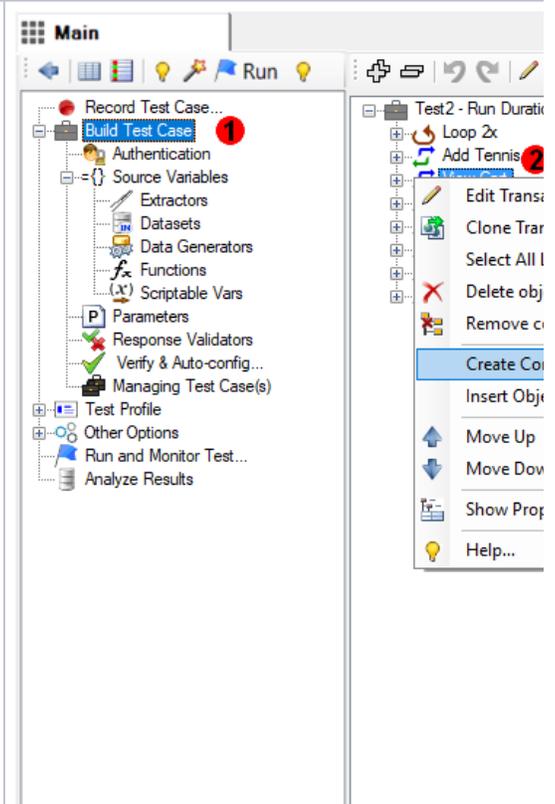
Two types of conditions are supported.

- A **Search response** condition depends on finding a specified text or regular expressions in response.
- A **Match extractor** condition depends on matching an extractor value with a specified text.

A **Search response** exit condition should be defined to make the provided example work. This condition has to verify if the response contains "Location: http://www.website.com /result.aspx". In this case, exit the loop.

To create a conditional loop, follow these steps:

1. Go to the Build Test Case node.

2. Select the starting object.

3. Right-click to open the context menu and click **Create Container -> Create Do... While**.

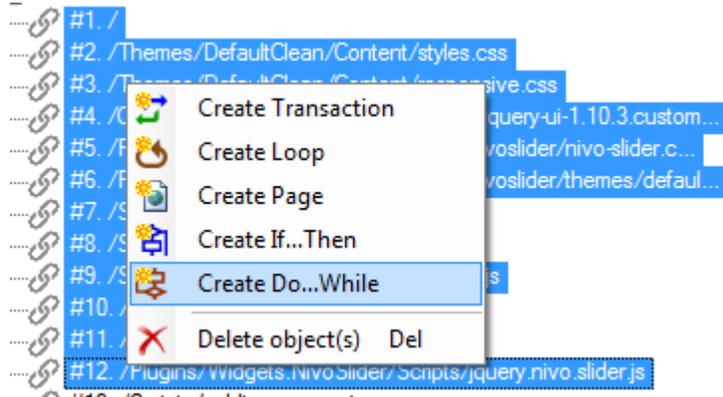4. A new object dialog will appear.

5. Select the last object in the loop

6. Click Create **Do...While**.

Quick Create:

Another way to create a loop is to select multiple objects the test case tree -> right-click -> select Create Do...While. This will generate a Do...While around the selected objects.

7. Set the **Number of Repeats (Max)** property to the maximum number of allowed loop iterations to avoid endless loops.

8. Optionally specify in the **Delay before next Loop**.

9. Select condition type as **Search response**, **Match extractor**, or **Match scriptable variable**

10. The search response condition is based on finding a specified text or regular expressions in an HTTP response. For this condition type, specify the following:

   a. From the drop-down, select **Response to search**, where the specified text or regular expressions will be searched.

   b. In the **Search string** property, specify a character string that will be searched in the HTTP response.

   c. In the **Search type** property, select **Text** if the search string is a text, or select **RegEx** if the search string is a regular expression.

   d. In the **Exit While Loop if match found?** property select **Yes** to repeat the loop when the search text is not found, and exit the loop when the search text is found. Or select **No** to repeat the loop, when the search text is found and exit the loop when the search text is not found.

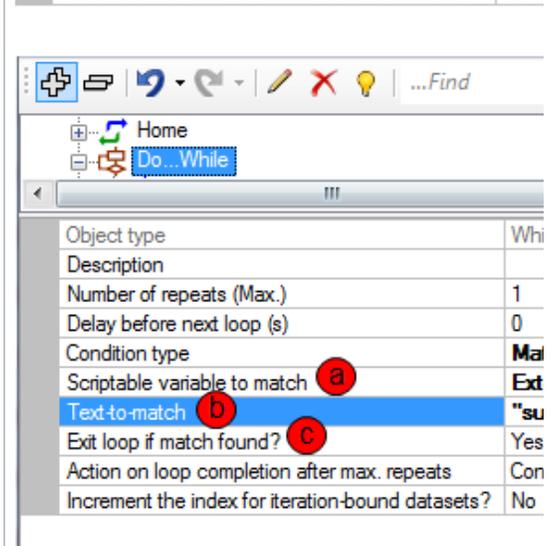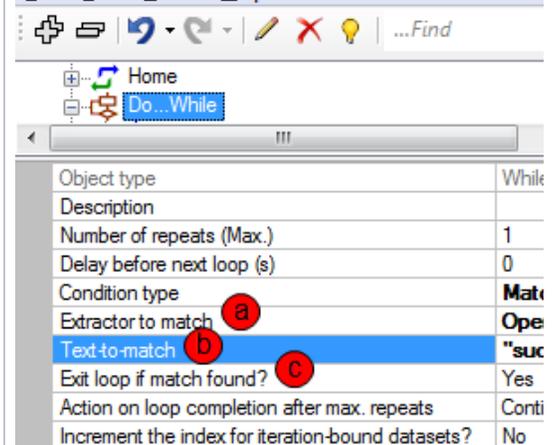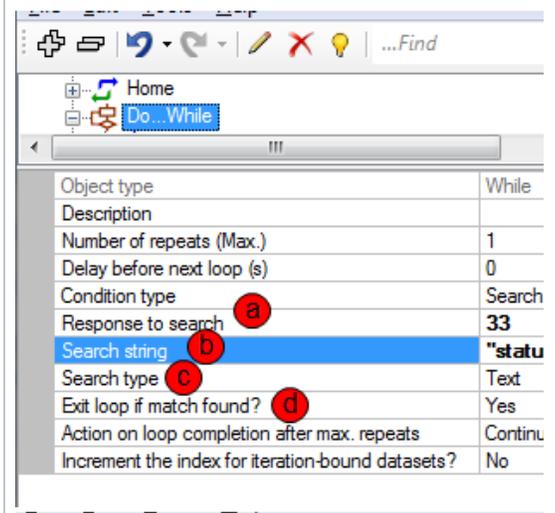11. Match extractor condition is based on evaluating an extractor. For this condition type, specify the following:

   a. From the **Extractor to match** drop-down, select the name of the extractor that will be used for the match.

   b. In the **Text to match** property, specify a text to match the extractor's value with.

   c. In the **Exit While Loop if match found?** the property, select **Yes** to repeat the loop when the extractor's value does not match value to match and exit the loop when the extractor value matches. Or select **No** to repeat the loop when the extractor value matches, and exit the loop when the extractor value does not match.

12. Match scriptable variable condition is based on evaluating the result of a scriptable variable that is returned by IExternalVariable.GetValue() method. For this condition type, specify the following:

   a. From the **Scriptable variable to match** drop-down, select the name of the scriptable variable that will be used for the match.

   b. In the **Text to match** property, specify a text to match the scriptable variable's value.

   c. In the **Exit While Loop if match found?** the property, select **Yes** to repeat the loop when the scriptable variable's value does not match value to match and exit the loop when the scriptable variable's value matches. Or select **No** to repeat the loop when the scriptable variable's value matches, and exit the loop when the scriptable variable's value does not match.

13. In some cases, if the condition is not met after maximum attempts, it is essential to stop executing the test case. For example, if the test case fails to log-in after a few attempts, it doesn't make sense to complete post-login transactions. Use the **Action on loop completion after max. repeats** property to control the outcome of failed conditions.

   a. **Continue**: To continue the iteration

   b. **Abort iteration**: To stop the current iteration and start a new one

   c. **Abort vu**: To stop sending requests from this VU.

Iteration-bound datasets

To increment the index for iteration-bound datasets on every loop iteration set the **Increment the index for iteration-bound datasets** property to Yes. For more information on databinding, see Databinding.