TestNG Executor

In this section:

Introduction

The TestNG Executor tool enables you to execute TestNG tests in conjunction with the various other types of tests you can run with SOAtest. This enables you to design and execute a single test scenario that orchestrates TestNG tests as part of a broader sequence of events, such as setting up a test environment or test data, running unit tests via this tool, then executing mobile tests.

Requirements

- TestNG 6.9.3.10 is supported
- This tool requires SOAtest 9.9.0 or later

Installation

You can install this tool from the UI or from the command line.

UI Installation

1. Choose **Parasoft> Preferences** from the main menu.
2. Choose the **Syster Properties** tab and click **Add JARs**.
3. Brows for the testng*executor.jar* file and click **Apply**.
4. Restart SOAtest.

Command Line Installation

Add the *testngexecutor.jar* file to the `system.properties.classpath` property in your *settings.properties* file. For example:

```
system.properties.classpath=<path to jar>/testngexecutor.jar
```

Usage

You can add the TestNG Executor as a standalone tool using the wizard.

The test's class folder or jar file must contain the TestNG that you want executed. Other dependencies can be included in the jar/folder, added to the system properties, or both. The tool checks the jar/folder first, then the system properties if necessary.

Configuration

You can configure the following tool fields:

Field	Description	Required
Jar or Class Folder	Specify the test's class folder or jar file. If you provide a jar/folder without specifying the Class or Method, TestNG will first look for a Suite XML file with the default name (testng.xml) and attempt to use that file to configure which tests to run. If that Suite XML file is not available and no classes or methods are specified, then the tool will run all of the tests in the jar/folder.	Required
Class	Specify which TestNG class from the given jar or class folder to run.	Optional
Method	Specify which TestNG method to run. Leave this empty if you want to run all test methods in the specified class.	Optional

Groups	Specify the group(s) of tests you want to run. TestNG supports annotating test methods with groups, allowing you to partition test methods into different logical groupings. For example, TestNG unit tests could be partitioned into “unit” and “integration” tests, or “smoke” and “functional” tests. This field takes either a single group or a comma-separated list of groups. Refer to the TestNG documentation for more information: http://testng.org/doc/documentation-main.html#test-groups	Optional
Suite XML File	Specify the suite XML file(s) to run. Suite XML files (also called *testng.xml* files by default) specify which packages, classes, suites and tests to be included or excluded, define new groups, specify dependencies, provide data parameters, and more. Refer to the TestNG documentation for more information: http://testng.org/doc/documentation-main.html#testng-xml	Optional

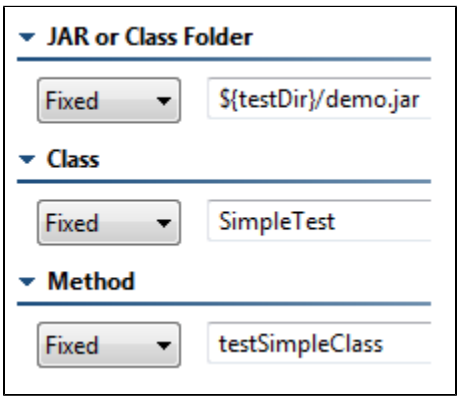
Parameters	Specify semicolon-separated key/value pairs for parameters that you want passed to parameterized test methods. This is an alternative to configuring parameters in a suite XML file. Refer to the TestNG documentation for more information:
	
	http://testng.org/doc/documentation-main.html#parameters
	
	You could specify the `first-name` and `last-name` parameters as follows if your test contained the snippet below:
	
	`first-name=John;last-name=Doe`
	
	`import static org.testng.AssertJUnit.*;`
	`import org.testng.annotations.*;`
	`public class DataTest {`
	`  @Parameters({ "first-name", "last-name" })`
	`  @Test public void testData(String firstName, String lastName) {`
	`  assertEquals("John", firstName);`
	`  assertEquals("Doe", lastName);`
	`  }`
	`}`
		Optional

Command Line Arguments	Specify the TestNG command line options and arguments for any other TestNG options you want to apply. Provide the command line arguments in the same format as you would provide on the TestNG command line. Refer to the TestNG documentation for more information: http://testng.org/doc/documentation-main.html#running-testng	Optional

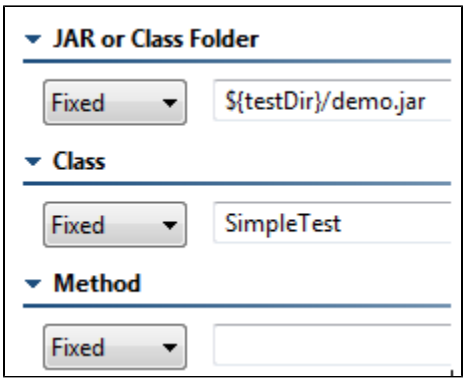
Examples

The following examples demonstrate how to configure the tool for different scenarios

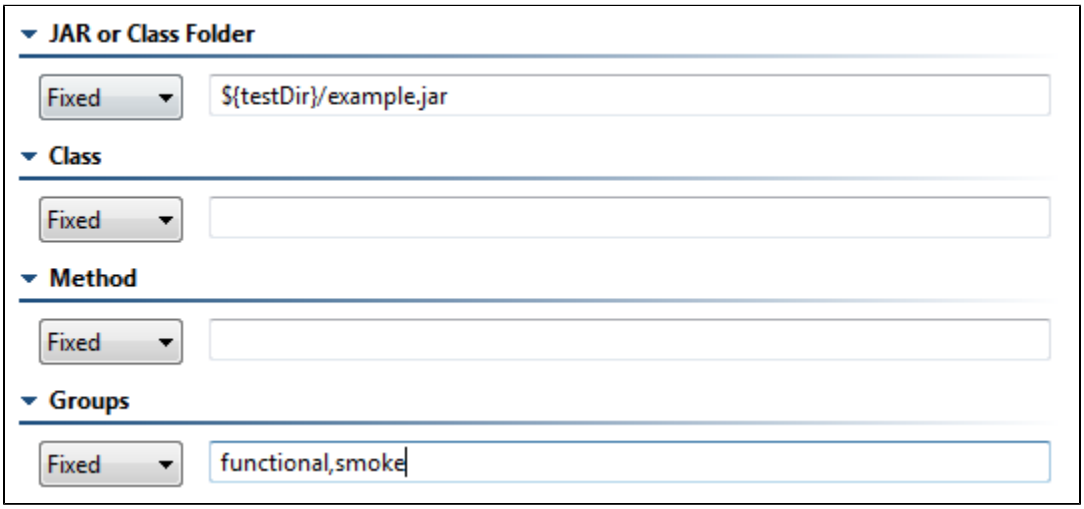
Running a Specific Test Method



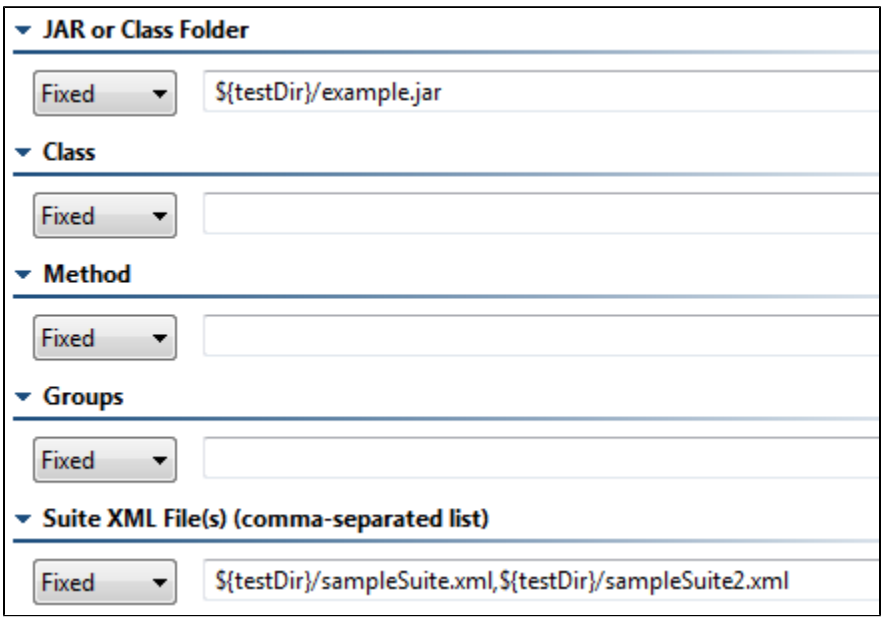
Running all Test Methods in a Class



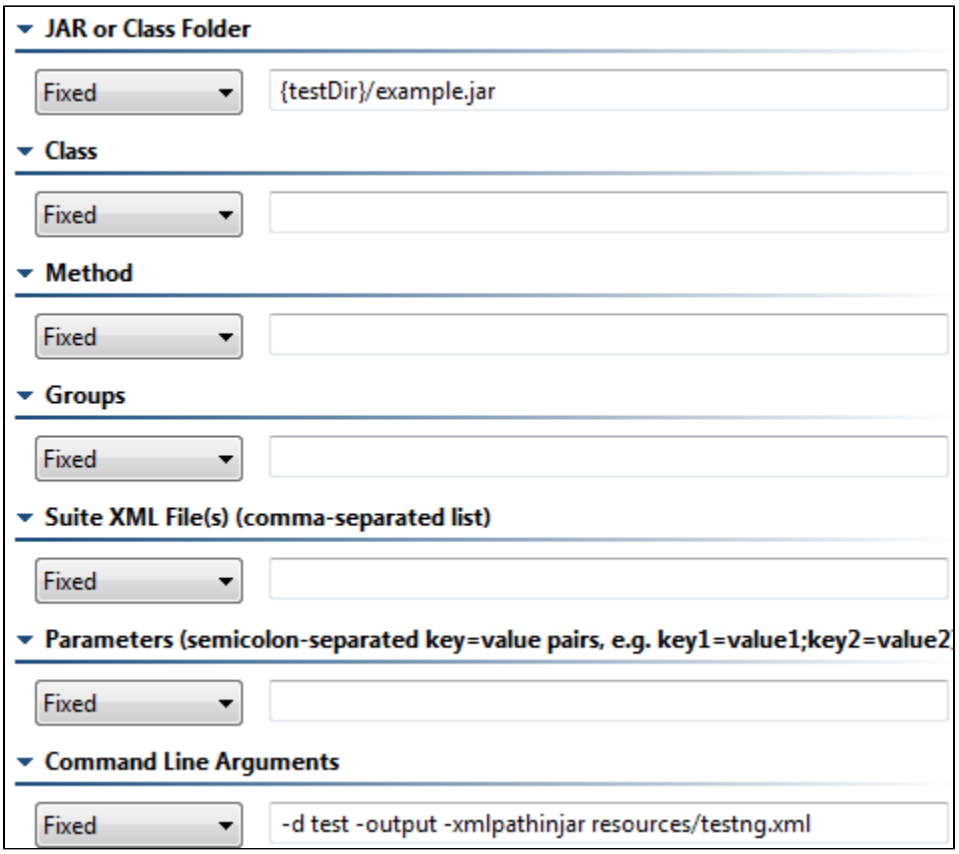
Running all Tests in Specific Groups



Running Suite XML Files

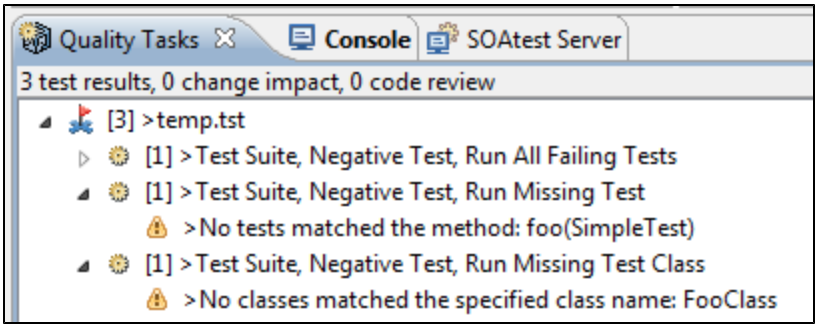


Specifying Command Line Arguments



Viewing Results

Execution details and results will be reported in the Console view. Additionally, any test failures detected will be reported in the Quality Tasks view:



Reporting Messages from TestNG Test to Console

Import the *com.parasoft.api* package into your project to access the Application Context, which enables SOAtest to report messages from your TestNG test execution to the console:

```
Application.showMessage("this displays in the console");
```

Add <SOATEST_INSTALL>/plugins/com.parasoft.ptest.libs.web_<version>/root/com.parasoft.api.jar to your Java project classpath to import the resource.

Retrieving Data from a TestNG Test

You can attach a SOAtest output to the TestNG Executor to retrieve data from your TestNG test output so that the data can be used in other tests. You need to configure the TestNG to indicate which values you want stored, and then attach an appropriate tool to the TestNG Output of the executor tool. The data will be passed to the attached tool in XML format. For example, you could send the data to a Diff tool, XML Assertor, or XML Data Bank tool.

Configuring the TestNG Output

Your TestNG must access the application context and store the desired values in a standard Java Map under the `custom_tool_TestNG_output` key. The TestNG Executor tool will check the application context for the stored map and convert it to XML that gets passed to the tool's "TestNG Output" output. The map's keys and values must be strings.

Import the *com.parasoft.api* package into your TestNG as described in Reporting Messages from TestNG Test to Console to access the Application Context.

Sending the Data to Another Tool

To send the output data from your TestNG test to a tool that processes XML data:

1. Right-click the TestNG Executor and choose **Add Output**.
2. Choose a tool for receiving the output and click **Finish.**

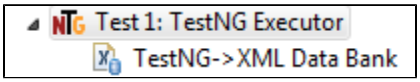
Example

In this example, we will store the values of the `multiplyResult` and `sumResult` TestNG operations in an XML Data Bank so that the values can be used in a subsequent test. The following procedure will occur:

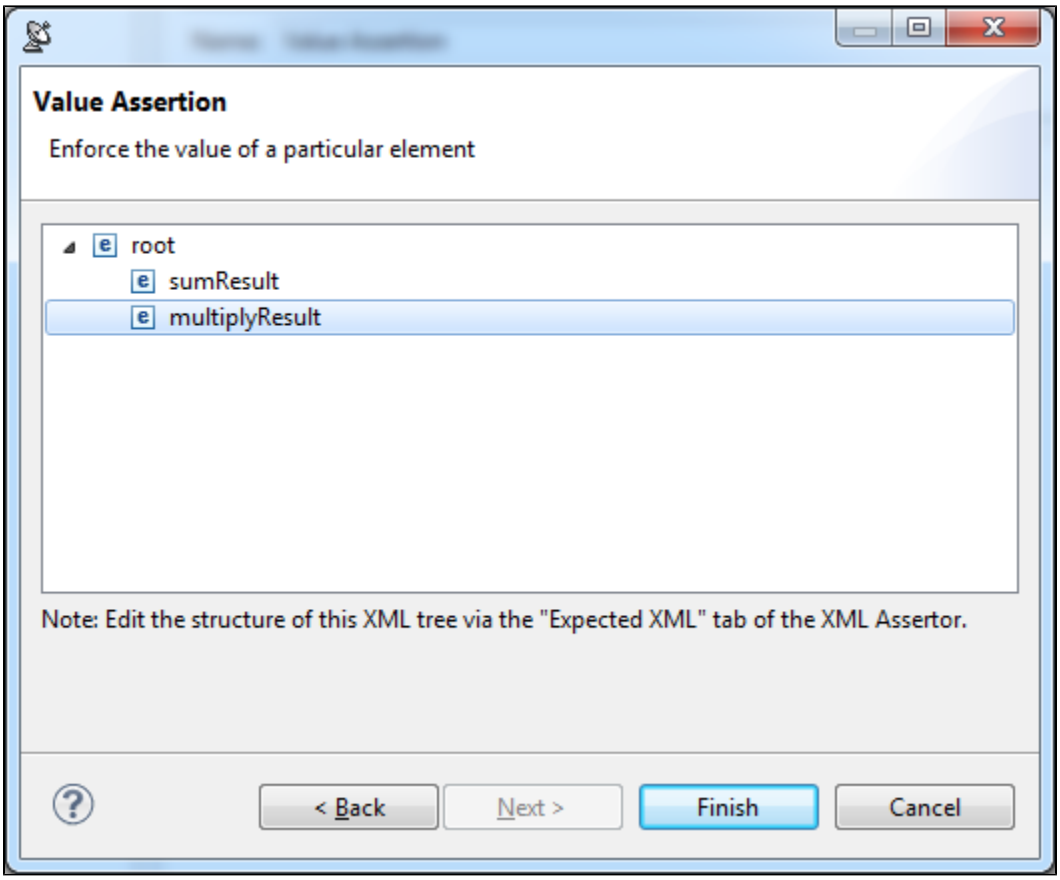
1. The TestNG test constructs a map containing the values
2. The map is placed into the application context.
3. The values in the map will be converted into an XML document that is passed to any tools attached to the TestNG Output

```
@Test
public void testExample() throws Exception {
  int multiplyResult = 8 * 8;
  int sumResult = 8 + 8;

// Create map representing name-value pairs for XML
  Map<String, String> map = new HashMap<String, String>();
  map.put("multiplyResult", String.valueOf(multiplyResult));
  map.put("sumResult", String.valueOf(sumResult));

// Get the Parasoft Scripting Context
  ScriptingContext context = Application.getContext();

// Place the map in the context with the custom_tool_TestNG_output
  context.put("custom_tool_TestNG_output", map);
  }
```

Add a TestNG Executor tool configured to run this test and add an XML Data Bank tool to the TestNG output.



Specify which values to extract and use in other tools. The content of the map appears below the `<root>` element within the XML document that is passed to the tools attached to the TestNG output. In this example, a value assertion tool was added.



Third-party Content

This plug-in includes items that have been sourced from third parties as outlined below.

- TestNG (Apache Version 2.0 License)

Additional license details are available in this plugin's licenses folder.