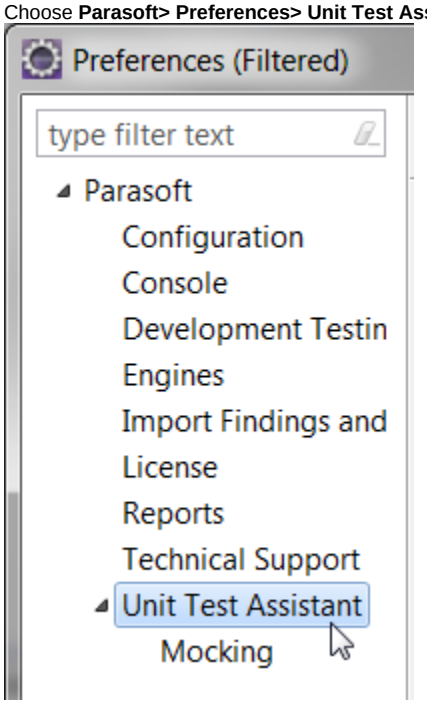
Configuring Preferences

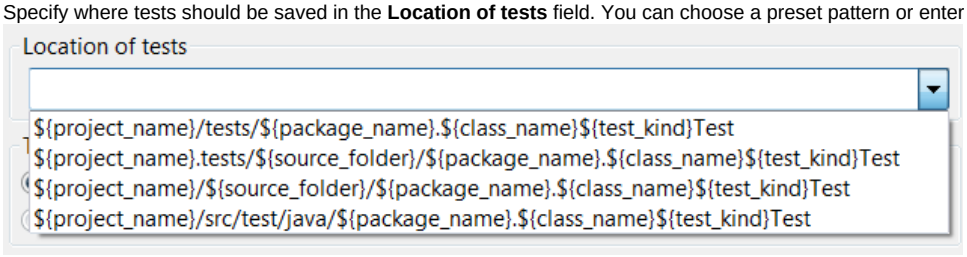
In this section:

Configuring General Options

1. Choose **Parasoft> Preferences> Unit Test Assistant** in the menu bar.

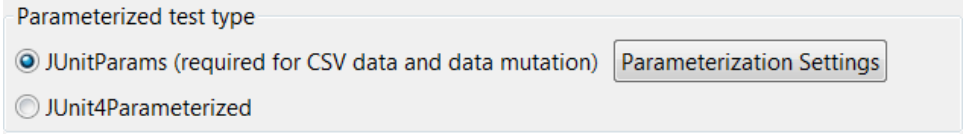


2. Specify where tests should be saved in the **Location of tests** field. You can choose a preset pattern or enter a custom pattern.



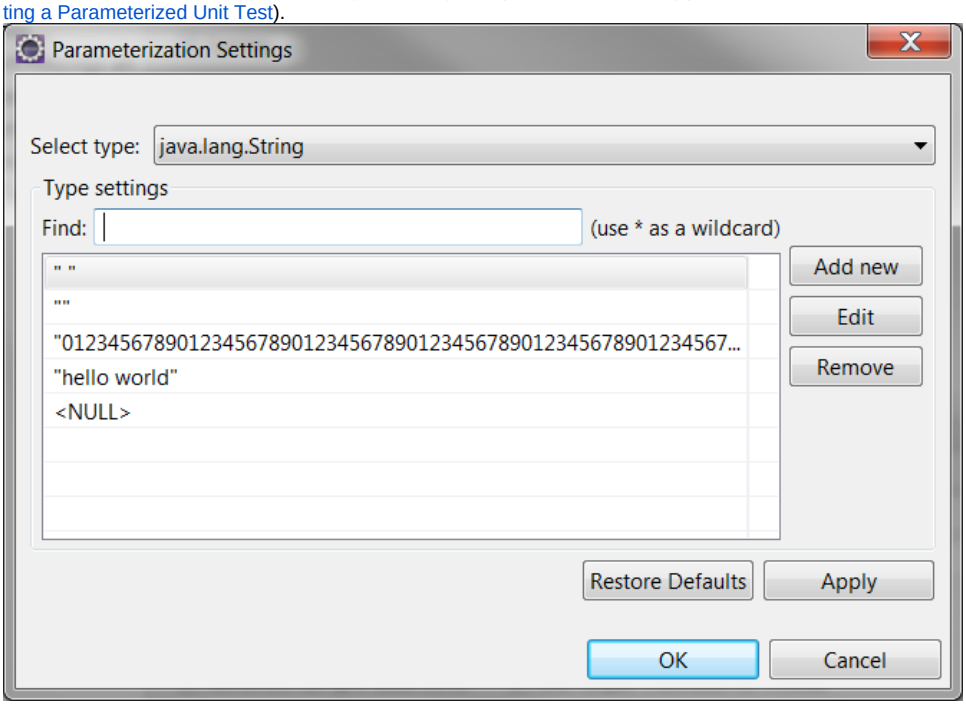
   You can select an option from a drop-down menu or customize the pattern with the following predefined Eclipse variables:
   `${class_name}` - uses the name of the tested class
   `${package_name}` - uses the name of the package that includes the tested class
   `${source_folder}` - uses the name of the folder that contains the source files of the project
   `${project_name}` - uses the name of the project that includes the tested class
   `${test_kind}` - uses the name of the test type, which allows you to separate regular tests from parameterized test cases

3. Select the framework for creating parameterized unit tests. JUnitParams is selected by default.



   If you use the JUnitParams framework, ensure that the JUnitParams library is added to your project; Adding Required Dependencies see for details.

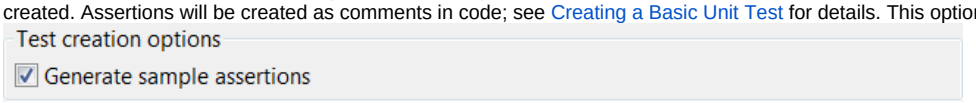
You can click **Parameterization Settings** to configure input data for creating parameterized test cases with the **Add test case(s)** option (see Creating a Parameterized Unit Test).



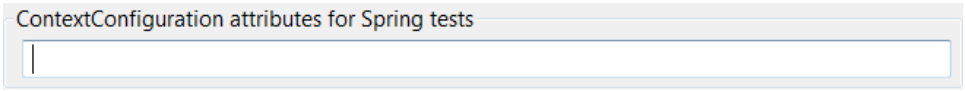
You can customize the default list of values by selecting a data type from the **Select type** drop-down menu and adding, editing or removing the values in the list. The **Find** field allows you to conveniently search for a particular value.

When you add new values, the characters must match the character encoding that will be used to save the files to ensure that the files are properly generated.

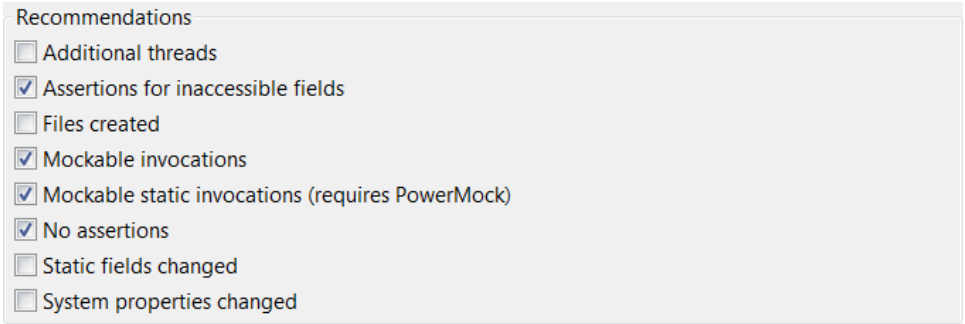
4. Enable or disable the **Generate sample assertions** option. If enabled, UTA will automatically generate assertion templates when a test case is created. Assertions will be created as comments in code; see Creating a Basic Unit Test for details. This option is enabled by default.



5. (optional) Specify the attributes that will be included in the ContextConfiguration annotation when a Spring test is created; see Creating a Spring Unit Test.



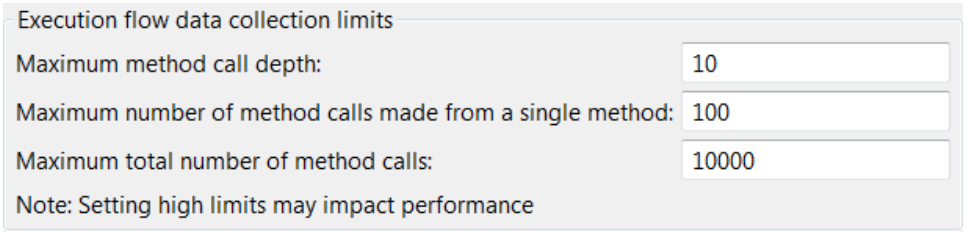
6. Specify which recommendations you want UTA to display after test execution.



Additional threads - detects side threads, which may impact the state of your test.
Assertions for inaccessible fields - detects inaccessible fields that have been modified during execution and generates assertion templates.
Files created - detects files that were created during the test run, but were not removed after execution.
Mockable invocations - detects calls to mock objects that can be modified to ensure proper test isolation.
No assertions - detects when no assertions have been made.
Static fields changed - detects when static fields have been modified during test execution.
Mockable static invocations (requires PowerMock) - detects calls to the static methods that are configured to be mocked with PowerMock (see Configuring Mocking Options)
System properties changed - detects system properties that were modified during the test run, but not restored after execution.

See Executing Unit Tests with Unit Test Assistant for examples of recommendations displayed by UTA.

7. Specify the limits for collecting data during execution.

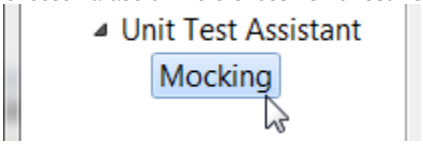


Maximum method call depth - Specifies the maximum depth of method calls during analysis.
Maximum number of method calls made from a single method - Specifies the maximum number of sub-method calls from a single method.
Maximum total number of method calls - Specifies the maximum number of all method calls during analysis.

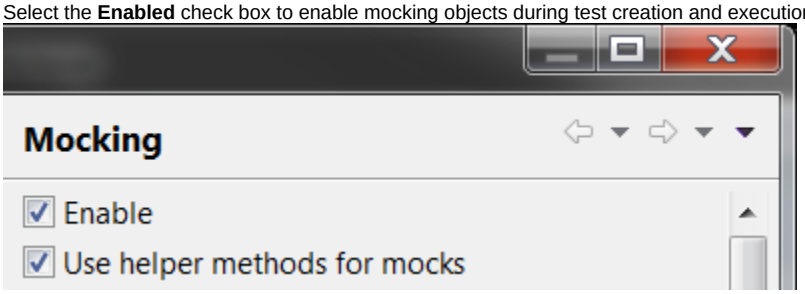
8. Click **Apply**.

Configuring Mocking Options

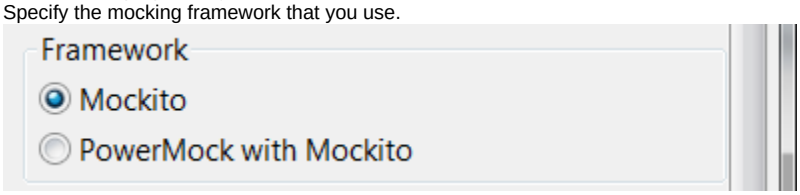
1. Choose **Parasoft> Preferences> Unit Test Assistant> Mocking** to configure mocking options.



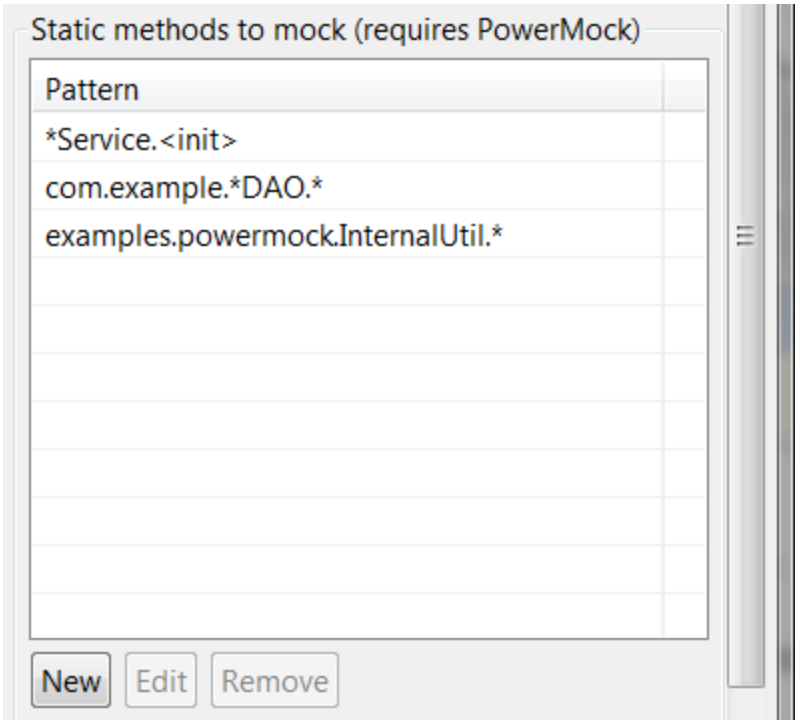
2. Select the **Enabled** check box to enable mocking objects during test creation and execution.



3. Enable or disable the **Use helper methods** for mocks option. If enabled, generated tests classes will separate regular test methods from helper methods that prepare objects but do not make assertions.
4. Specify the mocking framework that you use.



5. If you select PowerMock, specify a list of static methods and constructors to be mocked. Click **New** to add a new method or pattern. Use qualified method names and wildcards (*) to match patterns. The patterns that end with `.*` or `<init>` will be matched with constructors. For example, the following configuration will mock:
- all constructors in all classes whose names end with "Service",
- all static methods and constructors in all `DAO` classes in all sub-packages of "com.example",
- all methods of the `InternalUtil` class.



If the specified pattern matches all methods of a class, UTA will use the `mockStatic()` method which mocks all static methods of a class. Otherwise, the `spy()` method will be used to mock individual methods that are specified. For details about the `mockStatic()` and `spy()` methods, see https://github.com/powermock/powermock/wiki/Mockito#usage.

Only the static methods and constructors specified in the table will be mocked. UTA will automatically add mocks to test templates during test creation, or display recommendations that will help you add mocks after your tests are run (see Creating Mocks).

6. Click **Apply.**