Junit Sampler Tutorial

This tutorial attempts to explain the basic design, functionality and usage of the new Junit Sampler for Jmeter. The sampler was introduced in 2.1.2 release of Jmeter. Earlier releases do not have the sampler.

Design

The current implementation supports standard Junit convention and extensions, like oneTimeSetUp and oneTimeTearDown. Other features can be added on request. The sampler works like the JavaSampler with some differences.

1. rather than use Jmeter's test interface, it scans the jar files for classes extending junit's TestCase class. This means any class or subclass.
2. Junit test jar files are copied to jmeter/lib/junit instead of jmeter/lib
3. Junit sampler does not use name/value pairs for configuration. The sampler assumes setUp and tearDown will configure the test correctly.
4. The sampler measures the elapsed time only for the test method and does not include setUp and tearDown.
5. Each time the test method is called, Jmeter will pass the result to the listeners.
6. Support for oneTimeSetUp and oneTimeTearDown is done as a method. Since Jmeter is multi-threaded, we cannot call oneTimeSetUp/oneTimeTearDown the same way maven does it.
7. The sampler reports unexpected exceptions as errors.

Functionality

Here is a description of the functionalty.

Name – name for the sample. This is the same as all jmeter samplers.
Package Filter – provides a way to filter the classes by package name.
Classname – the name of the class to test. The sampler will scan the jar files in jmeter/lib/ext and jmeter/lib/junit for classes extending junit's TestCase.
Constructor String – a string to pass to the string constructor of the test class.
Test Method – the name of the method to test in the sampler.
Success message – a descriptive message indicating what success means.
Success code – an unique code indicating the test was successful.
Failure message – a descriptive message indicating what failure means.
Failure code – an unique code indicating the test failed
Error message – a description for errors
Error code – some code for errors. Does not need to be unique
Do not call setUp and tearDown – set the sampler not to call setUp and tearDown. By default, setUp and tearDown should be called. Not calling those methods could affect the test and make it inaccurate. This option should be used with caution. If the selected method is oneTimeSetUp or oneTimeTearDown, this option should be checked.
Append assertion error – By default, the sampler will not append the assert failures to the

failure message. To see the message in the result tree, check the option.
Append runtime exception – By default, the sampler will not append the exceptions to the failure message. To see the stacktrace, check the option

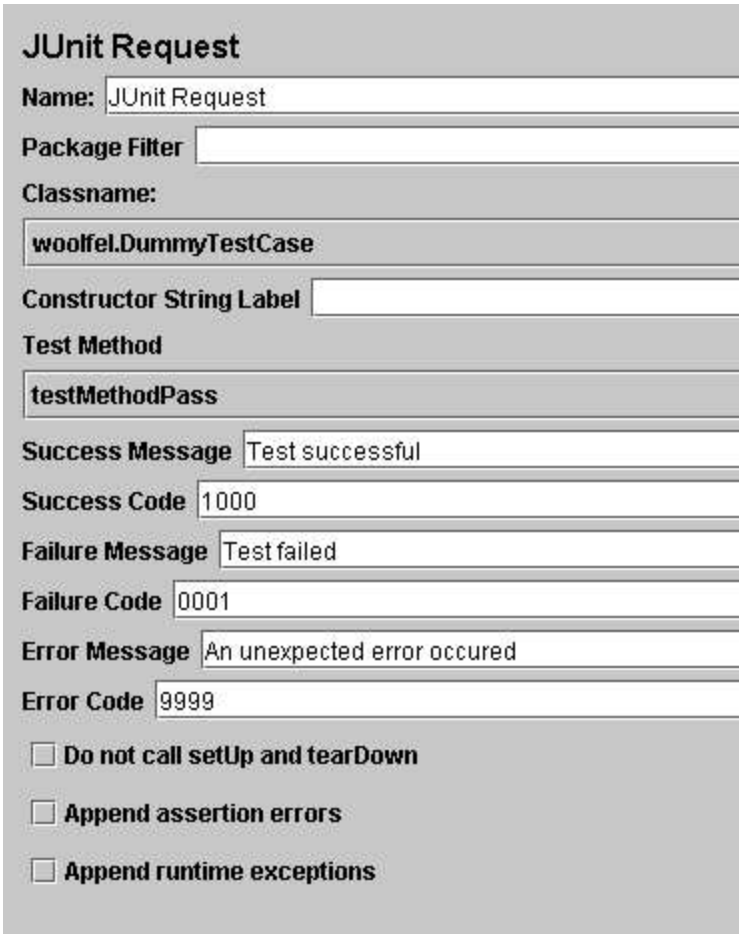
The current implementation of the sampler will try to create an instance using the string constructor first. If the test class does not declare a string constructor, the sampler will look for an empty constructor. Example below:

Empty Constructor:

```
public class myTestCase {
  public myTestCase() {}
}
```

String Constructor:

```
public class myTestCase {
  public myTestCase(String text) {
    super(text);
  }
}
```



By default, Jmeter will provide some default values for the success/failure code and message.

Users should define a set of unique success and failure codes and use them uniformly across all tests.

Usage

Here is a short step-by-step.

1. write your junit test and jar the classes
2. copy and paste the jar files into jmeter/lib/junit directory
3. start jmeter
4. select "test plan"
5. right click add -> thread group
6. select "thread group"
7. right click add -> sampler -> junit request
8. enter "my unit test" in the name
9. enter the package of your junit test
10. select the class you want to test
11. select a method to test
12. enter "test successful" in success message
13. enter "1000" in success code
14. enter "test failed" in failure message
15. enter "0001" in failure code
16. select the thread group
17. right click add -> listener ->view results tree

One benefit of the Junit sampler is it allows the user to select any method from a variety of unit tests to create a test plan. This should reduce the amount of code an user needs to write to create a variety of test scenarios. From a basic set of test methods, different sequences and tests can be created using Jmeter's GUI.

For example:

Test Plan1
- TestCase1.testImportCustomer
- TestCase2.testUpdateRandomCustomer
- TestCase1.testSelect100
- TestCase2.testUpdateOrder
- TestCase1.testSelect1000

TestPlan2
- TestCase1.testImportCustomer
- TestCase1.testSelect100
- TestCase1.testSelect1000
- TestCase2.testAdd100Customers

General Guidelines

Here are some general guidelines for writing Junit tests so they work well with Jmeter. Since

Jmeter runs multi-threaded, it is important to keep certain things in mind.

1. Write the setUp and tearDown methods so they are thread safe. This generally means avoid using static memebers.
2. Make the test methods discrete units of work and not long sequences of actions. By keeping the test method to a descrete operation, it makes it easier to combine test methods to create new test plans.
3. Avoid making test methods depend on each other. Since Jmeter allows arbitrary sequencing of test methods, the runtime behavior is different than the default Junit behavior.
4. If a test method is configurable, be careful about where the properties are stored. Reading the properties from the Jar file is recommended.
5. Each sampler creates an instance of the test class, so write your test so the setup happens in oneTimeSetUp and oneTimeTearDown.
6. If you select a class and no methods show up, it means the sampler had a problem creating an instance of the test class. The best way to debug this is to add some System.out to your class constructor and see what is happening.