Tutorial for Writing Scenarios

OSMO Tester

MBT tool

v3.3

Teemu Kanstrén

Table of Contents

[Introduction 3](#_Toc372289091)

[Startup Sequence 3](#_Toc372289092)

[Slicing Steps 4](#_Toc372289093)

[Forbidden Steps 6](#_Toc372289094)

[Conclusions 7](#_Toc372289095)

[References 7](#_Toc372289096)

# Introduction

This tutorial describes using scenarios with OSMO Tester using simple examples. The reader should be familiar with the information presented in the OSMO Tester basic and data tutorials. This tutorial continues with the model presented in those tutorials.

The reader is expected to have basic knowledge of Java programming and ability to use their own favourite IDE such as Eclipse, IntelliJ, or Netbeans. The code shown in this tutorial is available in the OSMO Tester examples package.

# Startup Sequence

Previously in the basic tutorial we created a model that prints “HELLO” and “WORLD” in that order. In the data tutorial we extended this to add string text names and numerical data values to these items. Surprisingly, this time I will not copy paste the same stuff here for the 5th time. Instead, let us assume that we have the model from those examples, and it has the steps “hello” and “world” in it. Let us also assume for this scenario example that these are allowed in any order. OK, so Listing 1 shows the model program.

public class HelloModel {

private ValueSet<String> names = new ValueSet<>("teemu", "bob");

private ValueSet<String> worlds = new ValueSet<>("mars", "venus");

@BeforeTest

public void startTest() {

System.out.println("TEST START");

}

@AfterTest

public void endTest() {

System.out.println("TEST END");

}

@TestStep("hello")

public void sayHello() {

System.out.println("HELLO "+names.next());

}

@TestStep("world")

public void sayWorld() {

System.out.println("WORLD "+worlds.next());

}

}

Listing 1. The model program.

There are two main parts to a scenario. A scenario can have a startup script, and a scenario can have one or more slices defined. If a startup script is defined, it will be execute at the beginning of each test case. So let us create a simple scenario with just a startup script:

public class Main1 {

public static void main(String[] args) {

OSMOTester osmo = new OSMOTester();

osmo.addModelObject(new HelloModel());

Scenario scenario = new Scenario(false);

scenario.addStartup("hello", "world", "hello");

osmo.getConfig().setScenario(scenario);

osmo.setTestEndCondition(new Length(5));

osmo.setSuiteEndCondition(new Length(2));

osmo.generate(55);

}

}

Listing 2. Running the model program.

Now, running this program gives the following result:

TEST START

HELLO teemu

WORLD mars

HELLO teemu

HELLO bob

HELLO teemu

TEST END

TEST START

HELLO teemu

WORLD mars

HELLO teemu

HELLO teemu

WORLD venus

TEST END

generated 2 tests.

Listing 3. Results for running scenario.

Here we see how all generated tests start with the given startup sequence “hello”, “world”, “hello”. After that, the generator is free to pick whatever it likes as there are no more constraints given.

Notice the parameter to the scenario constructor. This is the “strict” parameter, and is set to false here. The startup script is always executed as given. Unless of course the rules (guards) in the model forbid the given ordering, at which point the generation fails. After executing the startup sequence, the scenario can define a set of slices to control what steps can be taken and how many times. If the “strict” parameter is set to true, only the steps defined in the slices are allowed after the startup script. Since in this example we do not define any slices, this would mean there would be no step to take after the initial script.

# Slicing Steps

To illustrate slicing, let us set up another scenario for the same model:

public class Main2 {

public static void main(String[] args) {

OSMOTester osmo = new OSMOTester();

osmo.addModelObject(new HelloModel());

Scenario scenario = new Scenario(false);

scenario.addStartup("hello", "world", "hello");

scenario.addSlice("world", 0, 1);

osmo.getConfig().setScenario(scenario);

osmo.setTestEndCondition(new Length(8));

osmo.setSuiteEndCondition(new Length(2));

osmo.generate(55);

}

}

Listing 4. Scenario with maximum slice.

This one has the same startup script as the one before but also defines a slice for the “world” step. If basically says the step “world” should appear at minimum 0 times and at maximum 1 time after the startup sequence is finished.

TEST START

HELLO teemu

WORLD mars

HELLO teemu

HELLO bob

HELLO teemu

HELLO teemu

HELLO teemu

WORLD mars

TEST END

TEST START

HELLO teemu

WORLD venus

HELLO bob

WORLD mars

HELLO teemu

HELLO bob

HELLO teemu

HELLO bob

TEST END

generated 2 tests.

Listing 5. Results for running scenario.

Notice that in both tests, the startup sequence is there and there is one “world” step. We can also define the minimum number of times a step has to appear:

public class Main3 {

public static void main(String[] args) {

OSMOTester osmo = new OSMOTester();

osmo.addModelObject(new HelloModel());

Scenario scenario = new Scenario(false);

scenario.addStartup("hello", "world", "hello");

scenario.addSlice("world", 3, 0);

osmo.getConfig().setScenario(scenario);

osmo.setTestEndCondition(new Length(1));

osmo.setSuiteEndCondition(new Length(2));

osmo.generate(55);

}

}

Listing 6. Scenario with minimum slice.

Notice that we set the end condition for a test case to length of 1. This means it should stop after taking a single step. A startup sequence definition given alone would still stop after the first step (all tests would be just one “hello”). However, since we have a scenario minimum length defined for a step (3 for “world”), it goes on even after the end condition is met until the scenario is satisfied. Running this then produces the following:

TEST START

HELLO teemu

WORLD mars

HELLO teemu

HELLO bob

HELLO teemu

HELLO teemu

HELLO teemu

WORLD mars

HELLO teemu

WORLD venus

HELLO bob

WORLD mars

TEST END

TEST START

HELLO teemu

WORLD mars

HELLO bob

WORLD mars

HELLO teemu

HELLO bob

WORLD mars

HELLO bob

HELLO teemu

HELLO bob

HELLO bob

HELLO teemu

HELLO teemu

WORLD venus

TEST END

generated 2 tests.

Listing 7. Results for running scenario.

Notice how every test case again starts with the given startup sequence. After that, the choice is up to the test generation algorithm. However, each test case only ends after the “world” step has been taken three times as defined in the scenario.

# Forbidden Steps

Defining a slice with minimum and maximum of 0 is the same as not defining a slice at all. This is because only values greater than zero are taken into account to allow to define only one of them if desired. To completely forbid a step do the following:

public class Main4 {

public static void main(String[] args) {

OSMOTester osmo = new OSMOTester();

osmo.addModelObject(new HelloModel());

Scenario scenario = new Scenario(false);

scenario.forbid("world");

osmo.getConfig().setScenario(scenario);

osmo.setTestEndCondition(new Length(5));

osmo.setSuiteEndCondition(new Length(2));

osmo.generate(55);

}

}

Listing 8. Scenario with forbidden “world” (not forest).

Running this will now produce the following:

TEST START

HELLO teemu

HELLO teemu

HELLO bob

HELLO teemu

HELLO teemu

TEST END

TEST START

HELLO teemu

HELLO teemu

HELLO bob

HELLO teemu

HELLO bob

TEST END

generated 2 tests.

Listing 9. Results for running scenario.

Here we have five steps as requested and none of them “world” since we forbid it.

That is pretty much all there is to scenarios..

# Conclusions

This tutorial showed how to create scenarios to slice test models using OSMO Tester. Have fun..

# References

OSMO Tester home page: <http://code.google.com/p/osmo/>