



Starting and Working with the
featherlite Model Editor

Author: Michael Gatto
Type: 5 minute tutorial
Date: 2011-02-18
Version: 0.1.0

Abstract

The goal of this 5 minute tutorial is to describe how to start and how to work with the model editor of the *featherlite* framework. We'll also see how changes performed in the modeling client are directly visible throughout the running framework implementation.

Contents

1 Overview	1
2 Prerequisites	1
3 Starting the Planning Server, the Model Editor and the Reference Planning Client	1
3.1 Shutting it all down	2
4 Working with the Model Editor	2
4.1 Orders	2
4.2 Resources	4
4.3 Script Editor and Policy Editor	5
5 Going Further	6

1 Overview

This short tutorial is meant to be completed in 5 minutes. Here, we show how to start the server and client sides of the reference planning implementation, and the Model Editor Client. We introduce several modifications in our model, and see how the changes are reflected in the reference planning client.

2 Prerequisites

To be able to complete this tutorial, you need to have downloaded three components of the *featherlite* framework from www.featherlite-framework.com:

1. the *featherlite* Planning Reference Server, which is also needed for the 5 Minute Planning Rich Client Reference Implementation tutorial;
2. the *featherlite* Planning Rich Client Implementation, which is also needed for the 5 Minute Planning Rich Client Reference Implementation tutorial;
3. the *featherlite* Model Editor.

Keep in mind that these components are executable applications, so you need to download the one specific to your platform (Windows, Linux or Apple, in the 32 or 64 bit version).

Note that in this tutorial we are using the infrastructure for Planning. However, this tutorial could also be described in terms of the Execution infrastructure (thus with the reference execution implementation). Our choice to use the planning server over the execution server is completely arbitrary.

3 Starting the Planning Server, the Model Editor and the Reference Planning Client

To start the planning server, simply double-click on the executable file called “PlanningServer”¹. Depending on the platform, you may see a terminal opening and displaying the logging information of the server. If you cannot see any logging information, you can expect the server to be running within 5 to 15 seconds on current consumer hardware.

To start the Modeling Client, simply run the executable file called “ModelEditor”, by double clicking, single clicking or otherwise executing it. Note that the server side (the planning server) needs to be running in order to ensure the correct communication between the model editor and the planning server. During startup, the model editor

¹ If you have access to a terminal, you may also start the server from the command line. Moreover, depending on your system configuration, it might be sufficient to single-click on an executable to run it.

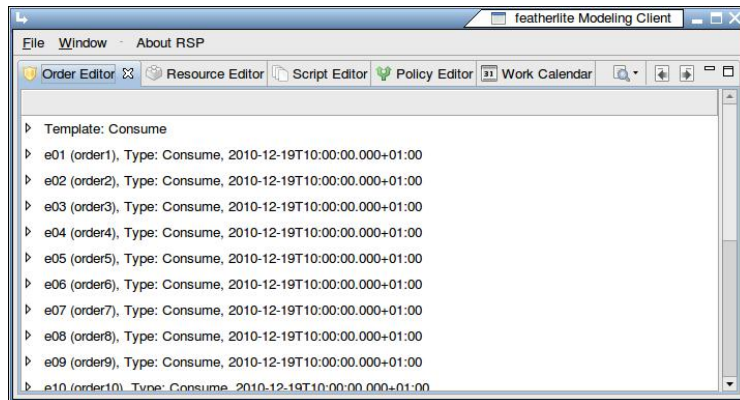


Figure 1: The Model Editor at startup, configured to start on the Order Editor view.

displays a splash screen, and, after it has finished loading, the view shown in Figure 1 appears.

We also start the planning client. To this aim, simply run the executable file called “PlanningClient”. Again, before starting the client, you must ensure that the server is running. During startup the client first displays a splash-screen and then the user interface you may have already seen in the 5 minute Planning Rich Client Reference Implementation Tutorial appears.

3.1 Shutting it all down

You may want to shut down the editor, the planning client and the planning server (hopefully after you completed this tutorial!). To close the modeling client, simply select the “File→Exit” menu entry. To close the client and shut down the server in one go, select the “File → Shutdown server and client ...” menu entry. This will first stop the client and close the user interface, and then remotely shut down the server as well. Keep in mind that if you do so, and have not saved your data in advance (either by exporting the model, or by enabling some persistence module) all changes you did to the model will be lost.

4 Working with the Model Editor

Since we have the application running, we proceed with analyzing the model, change it, and see the repercussions of these changes in the model.

4.1 Orders

In the “Order Editor” view, you can view and edit the characteristics of the orders. Here, we’ll manipulate order “e01” and increase the quantity to be produced from the initially defined 10.0 units to 20.0 units. To this aim, expand order “e01” by clicking on the triangle

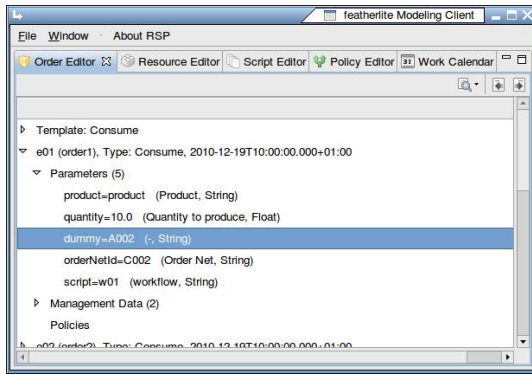


Figure 2: The tree-like view of the orders obtained by expanding the order “e01” and its “parameter” entry.

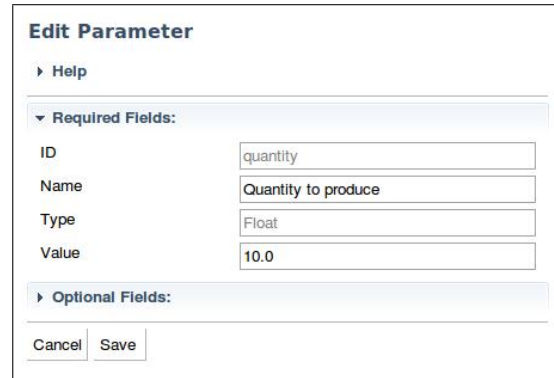


Figure 3: The dialog for editing the value of the parameter “quantity”.

marker at the left of the order; next, expand the parameter’s triangle marker. The tree-like representation shown in Figure 2 should have appeared. Now, double-click on the “quantity” entry. The dialog shown in Figure 3 appears. Edit the “Value” shown in the dialog and update the quantity from 10.0 to 20.0, for instance. Confirm your action by pressing the “Save” button.

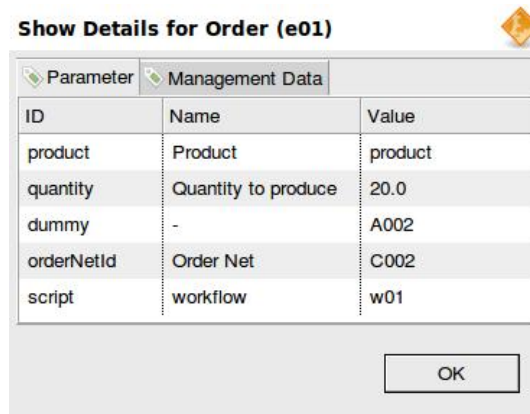


Figure 4: The “Order Detail” dialog of the planning client, reflecting the changes performed in the model editor for the order “e01”.

Now, we check that this change has propagated to the Reference Planning Client. Switch to the Reference Planning Client Window and click on the “Order” tab (if it is not already selected). Next, select the order “e01” by clicking on it, then open the context menu by clicking on the order with the right mouse button, and select the “Show order detail” menu entry of the context menu. In the dialog that appears, which is shown in Figure 4, you can now see that the quantity of this order is set to 20.0.

As a second change to the orders, we add a new order from a template. Templates are a special type of pre-configured orders that are used to create order instances with the pre-configured characteristics. To that aim, we switch back to the order view of the model editor. Select any order (be careful not to select one of the expanded entries of

Figure 5: The view of the dialog to add an order, with the id set to “myorder”, the name to “my new order”, and the date to midnight, central european time, of January first, 2100.

the order “e01”) and open the context menu by right-clicking on the order. Select the “Add Order” menu entry. In the dialog (see Figure 5) you can specify the order’s id, name and execution date. The id needs to be unique within the orders, so you can for instance set it to “myorder”. The order name has an informative character, and you can set it to “my new order”. The order date specifies (for *featherlite* configured in planning mode) the due date of this order. The date format required is ISO8601. To set it, e.g., to midnight of January 1st, 2100, we set the field to “2100-01-01T00:00:00.000+01:00”. Finally, we select the “Consume” template in the Template Selection list. In this way, the newly created order will have the standard configuration of so-called consume orders (more details of this can be found in *featherlite*’s Modeling Tutorial). Confirm by saving the order, which will now appear in the order list.

By expanding the details of the order through the marker at the left side of the order, we can see that it inherited the settings defined in the “Consume” template.

4.2 Resources

The resource view allows to edit and add new resource and resource templates. Here, we’ll update a resource template and propagate the change to all the instances derived from the template. To this aim, switch to the resource view by selecting the “Resource Editor” tab (see Figure 6). Select the entry “RawMaterial”. By opening its context menu, and selecting the “Edit Resource” menu entry, you can see that it has a “Type” value set to “Template”, as shown in Figure 7.

This is the template for raw resources (such as the lowest level components which are needed to build an item). We’ll extend this template by adding a cost factor per kilogram to the resource. This cost factor could then be used to compute the total costs of building an item.

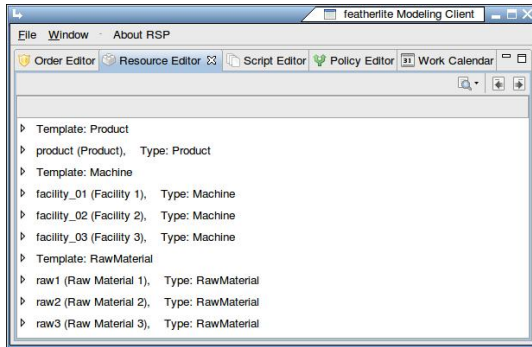


Figure 6: The “Resource Editor” view.

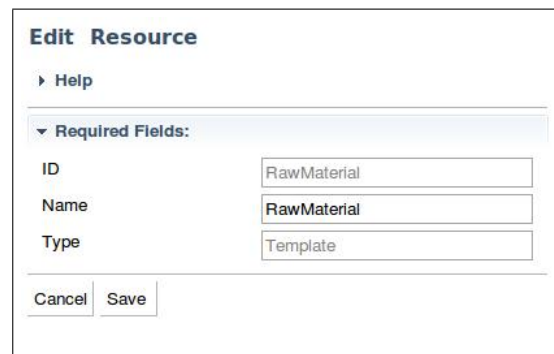


Figure 7: The “Edit Resource” view. Here, we can see that the type of the RawMaterial resource is “Template”

To this aim, expand the entries of the template “RawMaterial”, and select the “Parameter” entry. Open its context menu, and select the “Add Parameter” menu entry. The dialog shown in Figure 8 appears. Let’s fill in the data for the costs of the raw material. We set the ID of the parameter (that is how the parameter is referenced within the resources) to “costPerKg”, the name (which again has just an informative function) to “Cost Per Kilogram”. From the “Parameter Type Selection” list select the entry “FloatParameter”. As the description appearing on the right of the list says, this type of parameter wraps a numeric type “double” of Java, that is, a “real” (floating point) number. We initialize the “Value” to 0.0, as this is the value that will be propagated to any new and updated resource. Expand the “Optional Fields” view by selecting the triangle marker at the left. Here, we can set additional information for this parameter. For instance, we can set the unit of measure (UOM) field to Euro per Kilogram. This field has an informative value, but can be used in customized policies to get the units right between different resources. The fully configured parameter is shown in Figure 9.

After saving this parameter, we propagate the change to the resources which were created from this template. These resources are “raw1”, “raw2” and “raw3”, and the template they were derived from can be discovered in the “edit resource view” (the same we opened for the template): indeed, they show to have type “RawMaterial”. You may want to check that these resources do not already come with the parameter we just added: to do so, expand the resources and look if they have some parameters added! To update the resources, open the context menu of the “RawMaterial” template, and select the menu entry “Update Model”. A confirmation dialog saying that the model has been updated should appear. You can now see that the resources “raw1-3” now feature this parameter as well (see Figure 10).

4.3 Script Editor and Policy Editor

There are two views left in the Model Editor, which we omit to manipulate in this tutorial. It shall suffice to say that the Script Editor shown in Figure 11 allows to implement and configure the business logic of the orders to be processed, and the actions which shall

Add Parameter

▶ Help

▼ Required Fields:

ID

Name

Type

Value

▶ Optional Fields:

UOM

Interpretation

Sort Index

Is Hidden

▼ Parameter Type Selection

StringParameter

IntegerParameter

FloatParameter

BooleanParameter

DateParameter

Cancel Save

Figure 8: The dialog for adding a parameter to the resource template.

Add Parameter

▶ Help

▼ Required Fields:

ID

Name

Type

Value

▼ Optional Fields:

UOM

Interpretation

Sort Index

Is Hidden

▼ Parameter Type Selection

StringParameter

IntegerParameter

FloatParameter

BooleanParameter

DateParameter

A float parameter represents a floating point number accessible by 'ID', internally represented as a Java Double.

Cancel Save

Figure 9: The dialog for adding a parameter, with the values filled in.

occur as a consequence of planning (or executing) an order. The policy editor shown in Figure 12 has just an informative view, and allows to see what type of instances are configured for the different possible policies which exist within the model.

5 Going Further

In this tutorial, we showed how to perform some simple tasks to configure the model we are working with in a comfortable way. Even though we performed just some simple actions, this plug-in can be used extensively to configure the basic components of the business problem that is being model, from the orders to the resources involved and continuing with the business logic that links the orders with the resources. This user interface serves as an alternative to writing the model directly into the XML-serialized form, and takes care of producing a syntactically correct model (who never forgot to close an xml tag?). Moreover, this editor may be extended to fit your needs, for instance by automatically looking for feasible references for specific fields (like we do, e.g., for the possible parameter types in the “addParameter” view) and to make the extension of your model easier.

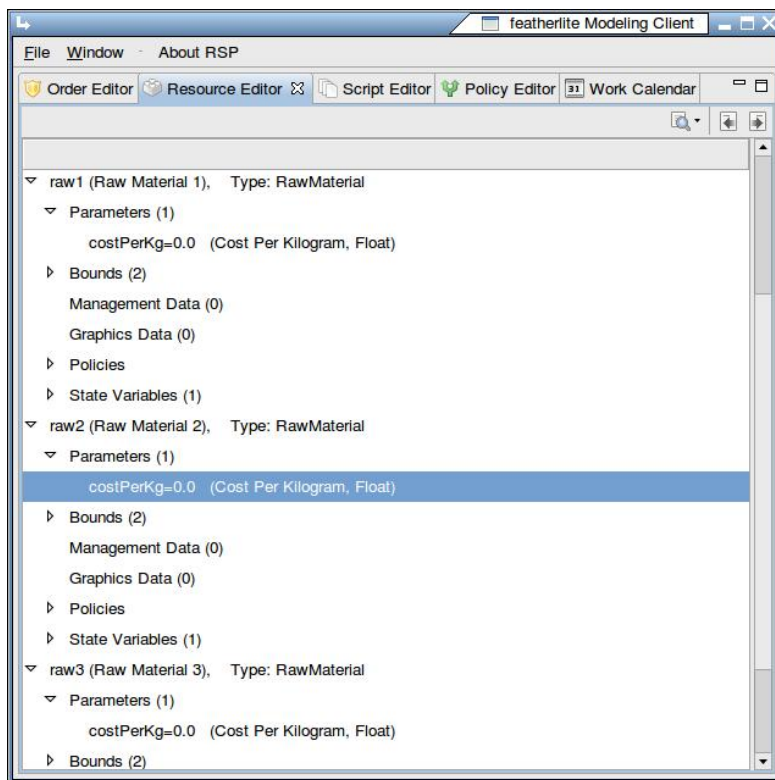


Figure 10: The updated raw resources, which now feature the parameter “costPerKg”.

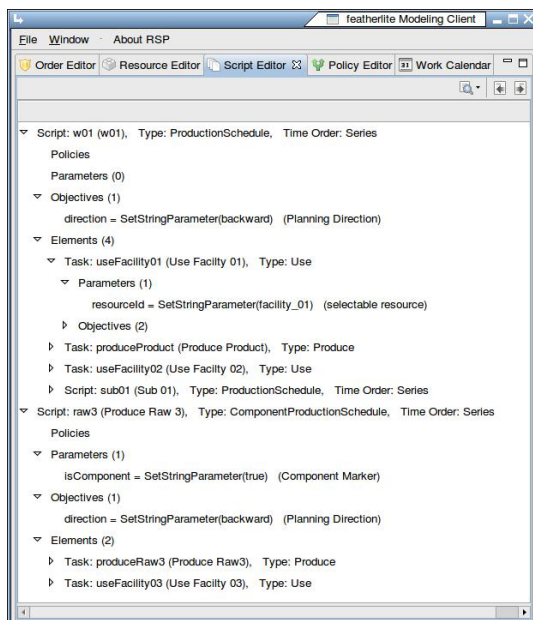


Figure 11: The “Script Editor” view.

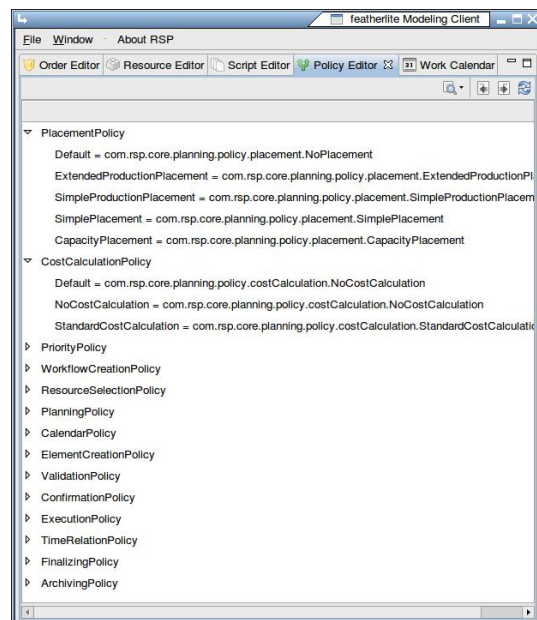


Figure 12: The “Policy Editor” view