# Model based Orchestration - Authoring Tool

# User Guide

Version: 1.1

26st Aug 2008

Kumaravel Ganesan

vel@hp.com

# Introduction

Authoring tool provides facility to develop SmarFrog’s Dependency based models (version 1.0). This tool is an Eclipse’s Rich Client platform product.

In this document, important features of the toll is listed down which may help the author to learn the tool and use it.

# How to Install

**Prerequisite:**

* Java 1.5

**Windows:**

Extract the given AuthoringTool.zip in to a folder say *authoringtool\_home* folder.

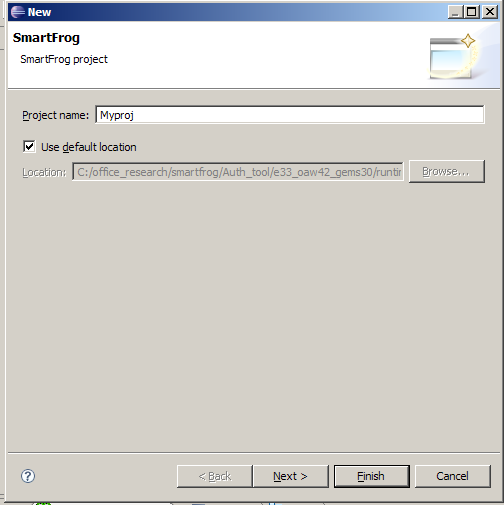
Just Click on *authoringtool\_home/ SFAuthoringTool.exe* to launch the tool

**Linux**

* *Working on the linux build*

# List of Features of the tool

# SmartFrog Project Creation



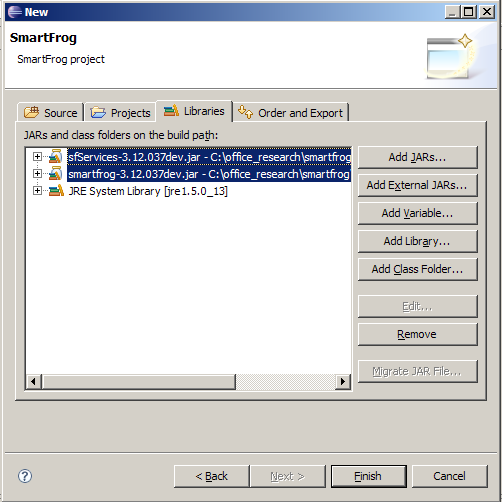
Figure

The general SmartFrog plug-in has been integrated in this tool. So the user straight away can create a new SmartFrog project as one do in normal SF plugin.

Create a new SF Project by selecting the following menu:

File 🡪 New 🡪 Java SmartFrog Project

Enter a new name for your project and create it and click next. Add SmartFrog core libraries in the build path as shown in figure 2.

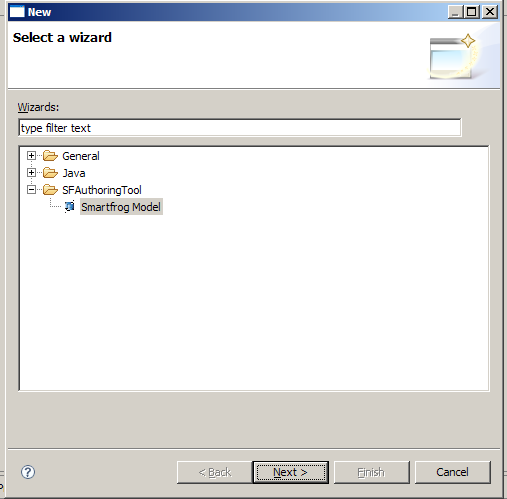


Figure

# New Model file Creation

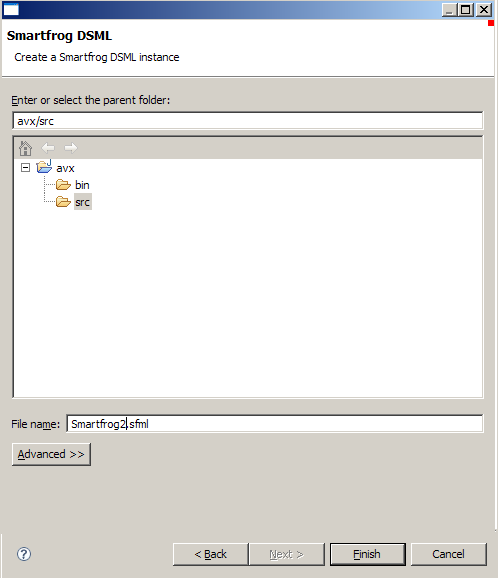
The following steps should be followed to create a SmartFrog Dependency Model.

1. Create a new SmartFrog model by selecting the manu : *New 🡪 Others 🡪SFAuthoringTool 🡪 SmartFrog Model* .



Figure

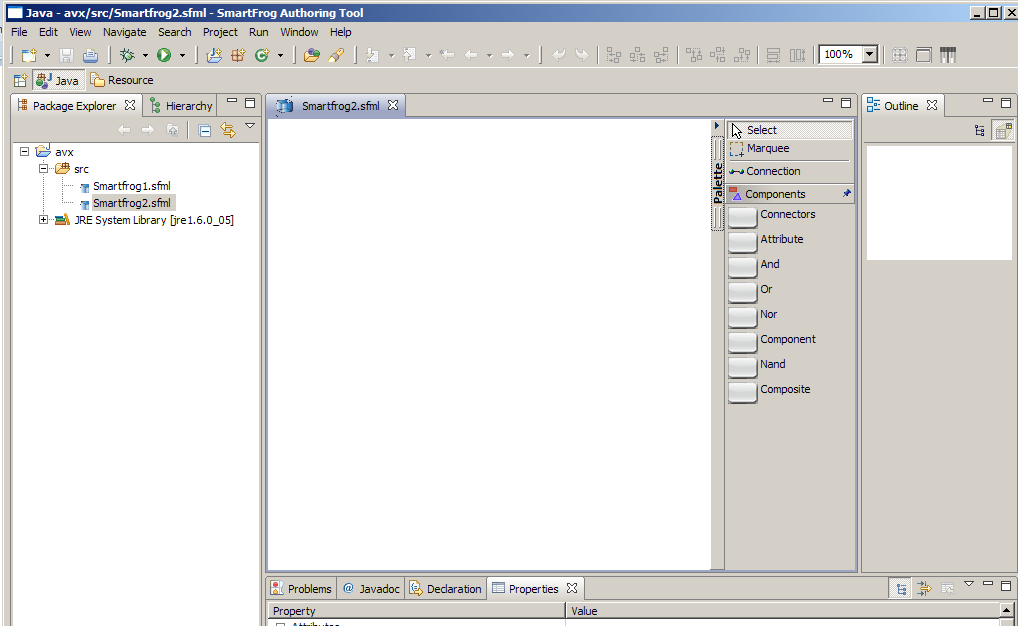
In the Next window, enter the file name for the model as shown in the figure- 4.



Figure

# Authoring Model

To start modeling, click the model file name in the left “package Explorer” view. Then the canvas will open in the editor as shown in the figure.



The palette in the editor has all the artifacts which can be used for authoring SmartFrog dependency. User can start modeling by clicking the artifacts in the palette and the dropping it in to the canvass. The properties for artifacts cane be specified in the property view.

Figure

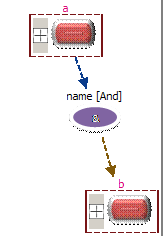
# Using artifacts in the Palette

The following list of components available in palette:

* Composite
* Component
* Attribute
* Set of Connectors (AND, OR, NOR & NAND)
* “Connection” element.

All components are state components and these components can be connected using connection element. Also two or more state components can be connected through Connecters provided in palette.

If the dependency connection is between two state components then it is simple connection. The required dependency properties like “on” and “by” are inferred automatically by the system when user creates a connection link. Other attribute values like “relevant” and “enabled” and the unique name for the dependency connection can be entered in the property view.



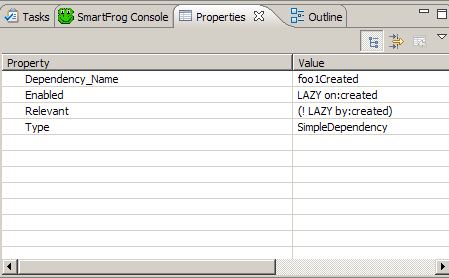
Figure

A connector can be used to link multiple dependences. The figure shows how it can be linked. When there is a connector, the tool will interpret the dependency link in different way compare with the way it understands in normal link between the state components. Here all the link towards the connector is type of “InputDependency” and the link goes away from connector is type of “OutputDependency”

An Attribute element can be placed as part of Composite or Component. And it’s value should be specified in property view.

# Enter the attributes in property view

User can enter the appropriate property values in the “property” view. It will show the relevant list of property for which user selected on the canvass.



Figure

# Outline view Facility

Figure

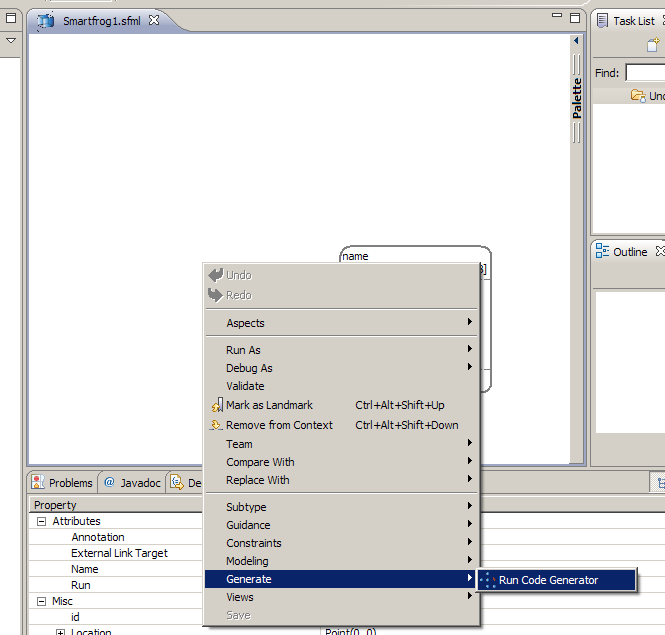
This facility helps to focus/zoom on the certain part of the model. This view can be opened from windows🡪show view🡪outline

# Importing existing .sfml file from outside of the tool

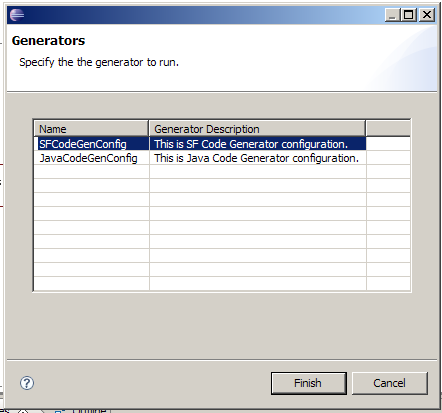
The tool facilitates to import the model file from outside of it. To perform this, user should go to File🡪 import menu and select the file.

# SmartFrog Description Generation

Write click on the model and select *Generate🡪 Run Code Generator*



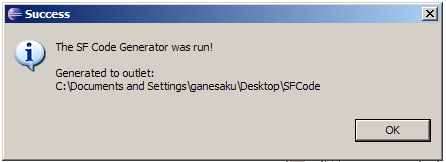
Figure



Figure

It will pop up a wizard to select the SF code generator.

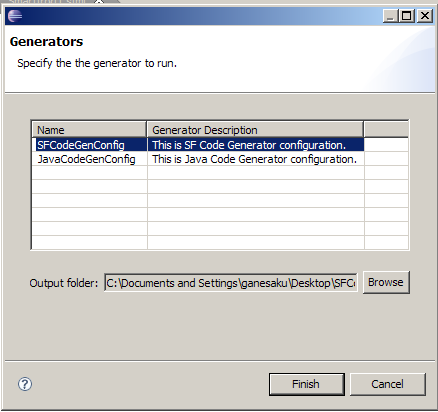
At the successful completion of code generation, it will show the success message as shown Figure 11. The generated code is placed by default in ‘src’ folder of the project.



Figure

# Java Template Generation

To generate the java template for the selected model, user should select “JavaCodeGenConfig” as shown in the figure. The generated code is placed by default in ‘src’ folder of the project.



Figure

# Other Features:

* The model can be zoom in and zoom out.
* It Integrates with SmartFrog Plugin - so that it can be parsed and deployed. (Few bugs need to be fixed.)