Generator Demos

Generator Demos

The Generator Demos provide task-oriented description of how to implement a language generator in MPS.
It consists of six related demos. Each demo is based on the previous one and demonstrates more advanced (yet sometimes just different) features, practices and approaches.

The problem we are 'solving' in these demos is generation of a Java Swing GUI from XML.
Although all demos are logically related, we will create a fresh generator in each of the demos. Because a generator can't exist in itself we will create a new language in each demo as well to wrap the generator.

Those languages (except for the last one in demo6) do not introduce own concepts. Instead they give new semantic to existing concepts.
For instance, xml element with name 'button' is not just an xml element any more but 'virtual concept' representing GUI component button.

Note that this is not a way the Language Oriented Programming is normally done. Languages usually are not 'virtual'. But this approach allows us to focus upon our topic - generators development, and leave other aspects of language development out of scope of this document.

How to Use This Documentation

Best of all is to install JetBrains MPS on your computer, set-up demo project and carry out all the steps described in demos. This way you will gain great experience in generators development and beyond.

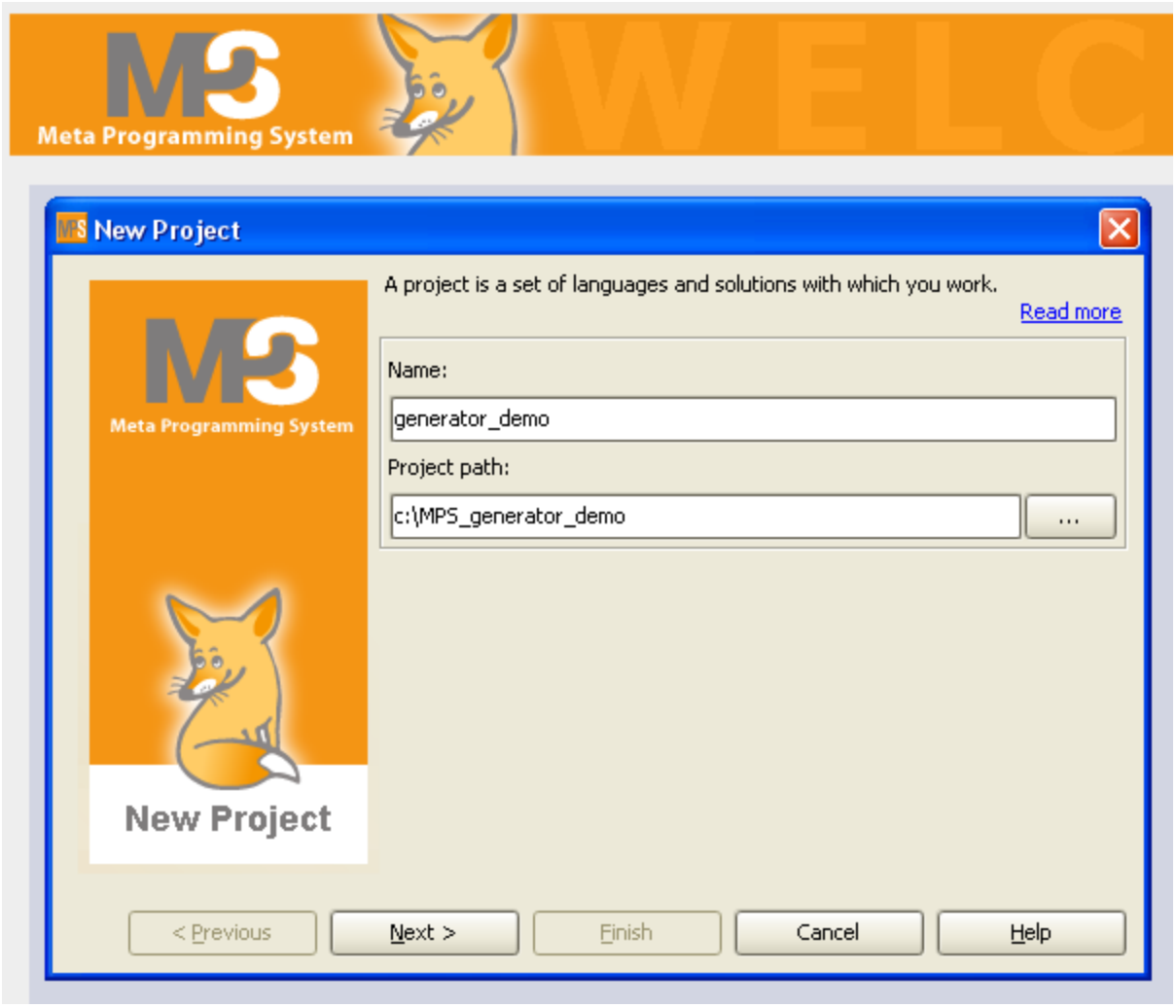
The second option is to read the documentation and, perhaps, browse generator demo project included into MPS build. The generator demo project includes all the demo languages and generators, as well as test models which we are going to create in these demos.
The generator demo project is located at:

```
{mps}/samples/generator_demo/generator_demo.mpr
```

Setting-Up Demo Project

New Project

- choose File->New Project
- enter project name: 'generator_demo' on the 1st page of the 'New Project' wizard
- uncheck 'Create new language' option on the 2nd page
- uncheck 'Create new solution' option on the 3d page

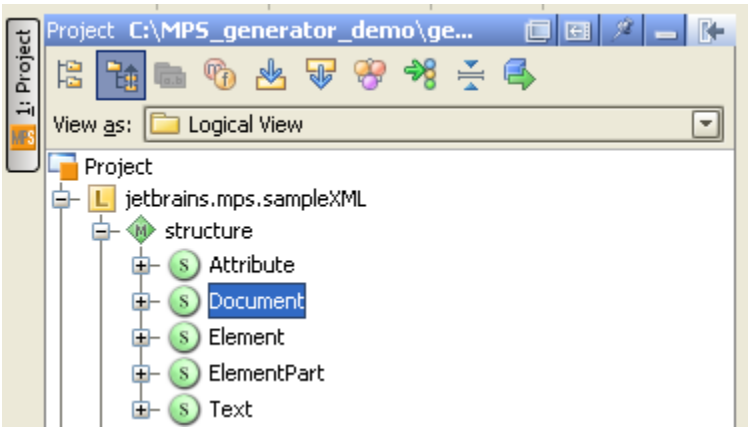


In this demos we are going to define new semantic for xml elements, create test models with some xml in them and generate java code from that xml.

To start with, let's add the 'jetbrains.mps.sampleXML' language to the 'generator_demo' project.

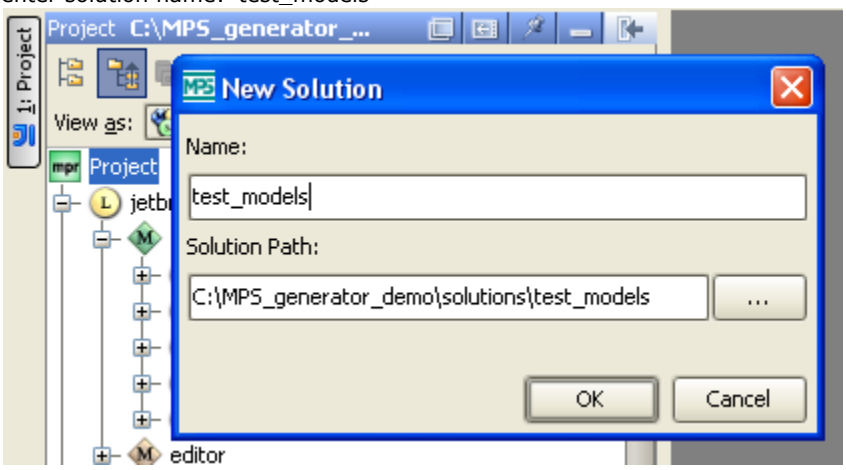
- choose Go To -> Go To Language in the main menu to find the 'jetbrains.mps.sampleXML' language
- in the 'Go To' dialog type 'sampleXML'
- choose 'jetbrains.mps.sampleXML' in pull down menu - MPS will navigate you to the 'jetbrains.mps.sampleXML' language module in project tree (logical view)
- in the 'sampleXML' tree node's popup menu choose Add To Project

Now the 'jetbrains.mps.sampleXML' language is easily accessible.



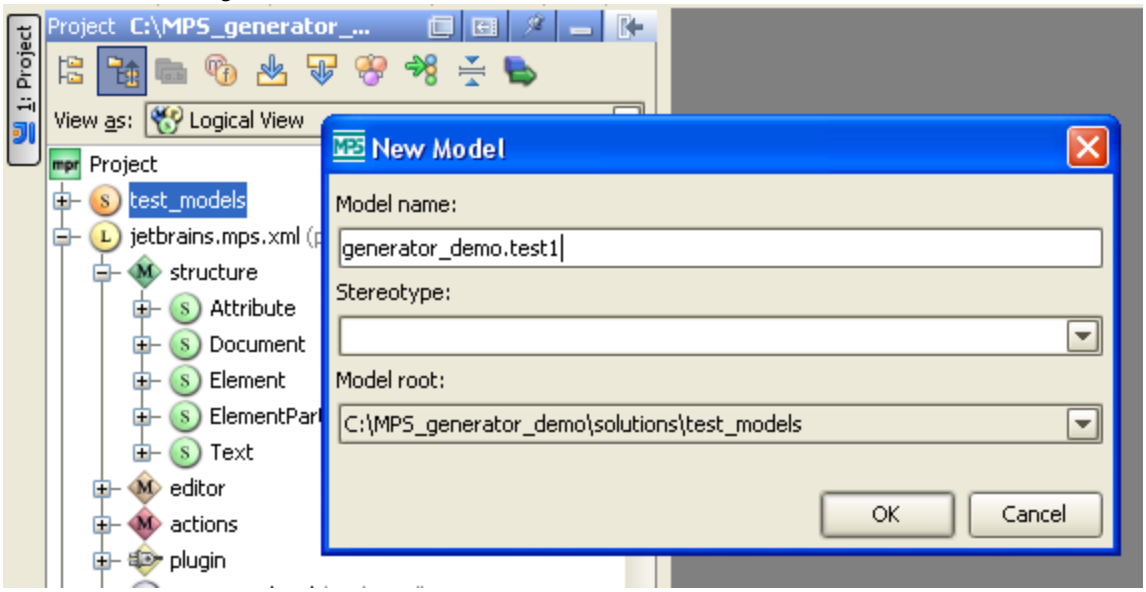
New Demo Solution

- choose New Solution in project's popup menu
- enter solution name: 'test_models'

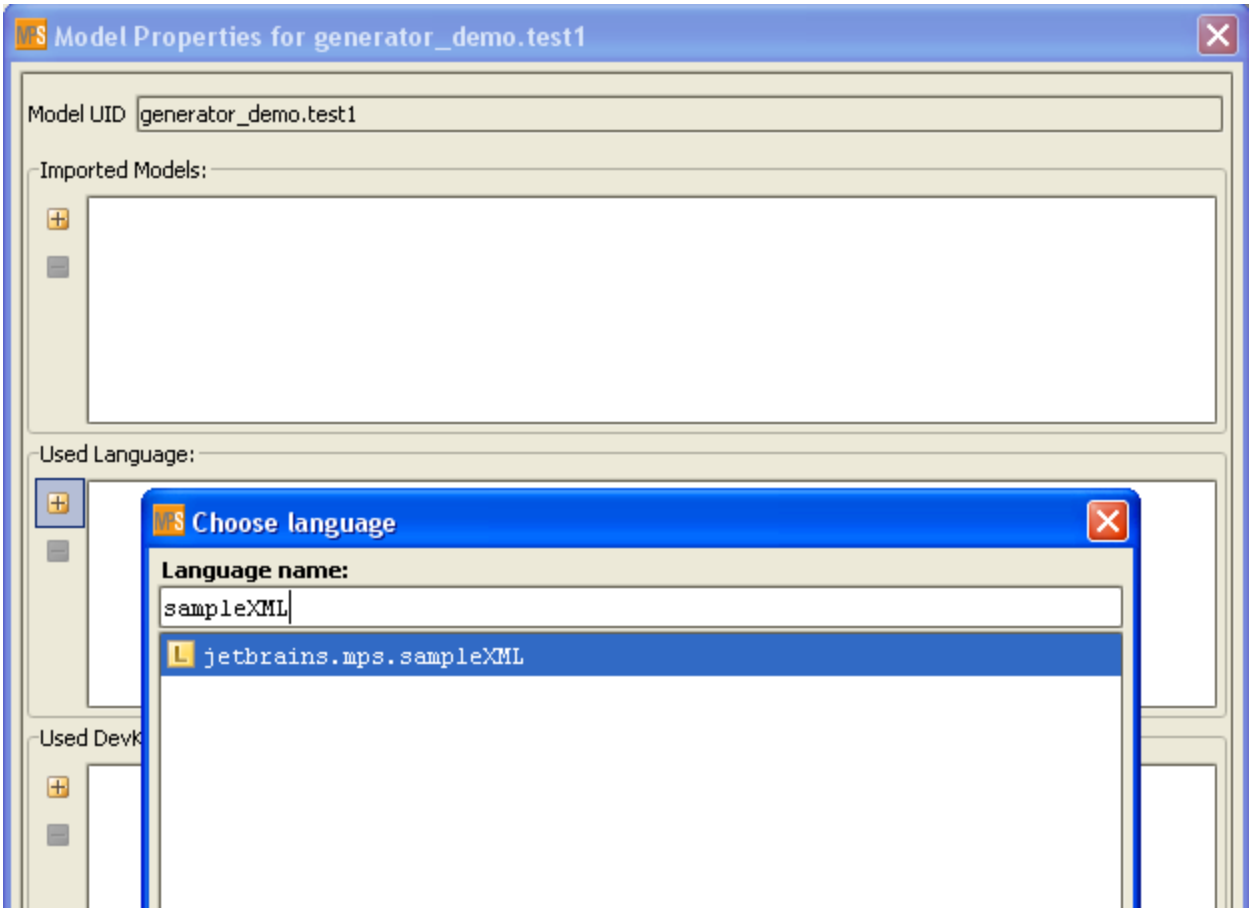


New Test Model

- choose New Model solution's popup menu
- enter model name: 'generator_demo.test1'

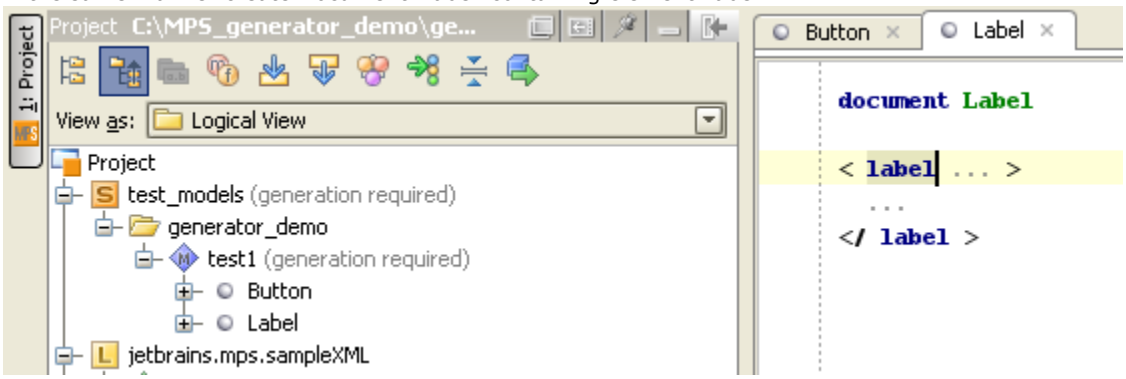


- add language 'jetbrains.mps.sampleXML' to Used Language section in model properties dialog



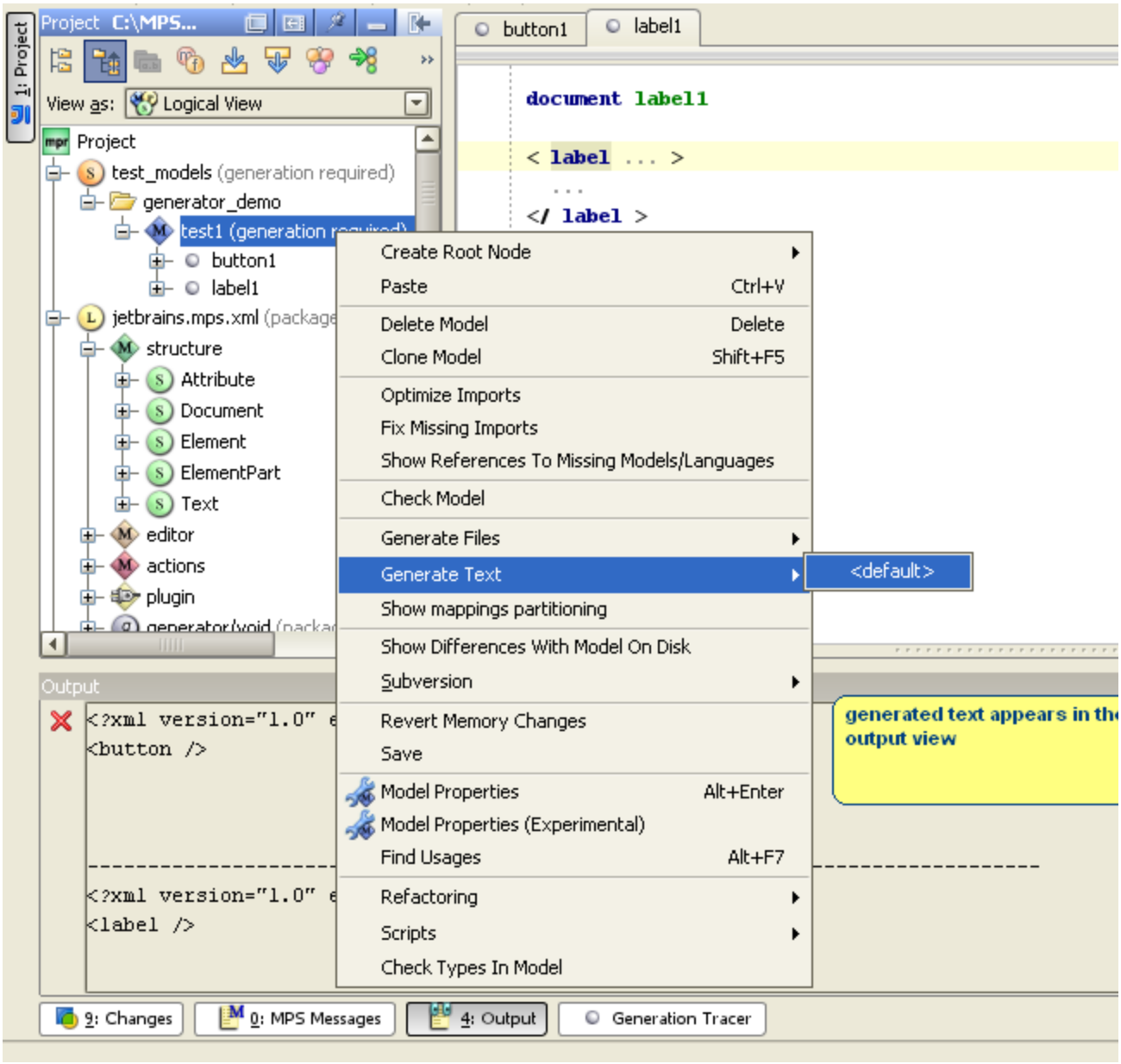
Now let's write some xml.
In the model 'text1' create Document and give it name 'Button'.

- create a Document 'Button' in the model 'test1'
- add root element 'button' to this document
- in the same manner create Document 'Label' containing element 'label'



Test Generation

- choose Generate Text in popup menu of the model 'test1'



Result of the text generation will be shown in MPS output view and it will be more or less the same xml as we have on input.
Now we are all set to go on and define semantics for those otherwise meaningless xml concepts.

What's Inside

Demo 1

Demonstrates usage of root mapping rule and root template; property-macro; SWITCH-macro and template switch.
We will also set-up an IntelliJ IDEA project to review and run our generated java applications.

Demo 2

Demonstrates usage of conditional root rule; LOOP-macro and abandon root rule.

Demo 3

Demonstrates usage of weaving rule; template declaration and template fragment; mapping label; 'unique name' service; refer ence-macro and IF/INCLUDE/MAP_SRC macros.

Demo 4

Demonstrates usage of reduction rule, COPY_SRC-macro and more advanced reference-macro with reference resolving by map ping label.

Demo 5

Explains how to use generation scripts and how to create utility classes in generator.

Demo 6

In this demo we will create a 'real' language defining it own higher-level concepts (button and label) and see how languages are integrated together.
This demo explains in greater details the generation process in MPS and demonstrates usage of transient models and generatio n tracer tool.