

oAW 4 UML2 Example

Markus Voelter, voelter@acm.org, www.voelter.de

PRIMARY SPONSORS



SECONDARY SPONSOR

sTable of Contents

SETTING UP ECLIPSE.....	3
SETTING UP THE PROJECT.....	3
CREATING A UML2 MODEL.....	4
MODELLING THE CONTENT.....	4
CODE GENERATION.....	5
DEFINING THE TEMPLATES.....	5
DEFINING THE WORKFLOW.....	6

Setting up Eclipse

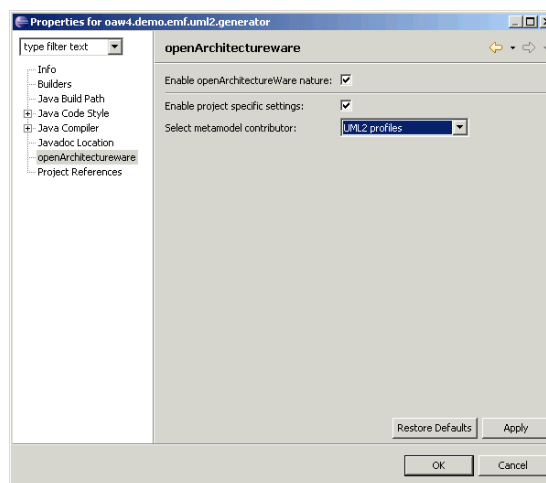
Before you can use oAW with Eclipse UML2, you first have to install the UML2 plugins into your Eclipse installation. Also, since the Eclipse UML2 infrastructure does a number of things differently from regular EMF, you have to install additional oAW plugins. Those two plugins can be found in the *oaw-uml2.4.x.x* package available from the oAW download page.

Setting up the project

Create a new Java project. You have to add the following libraries:

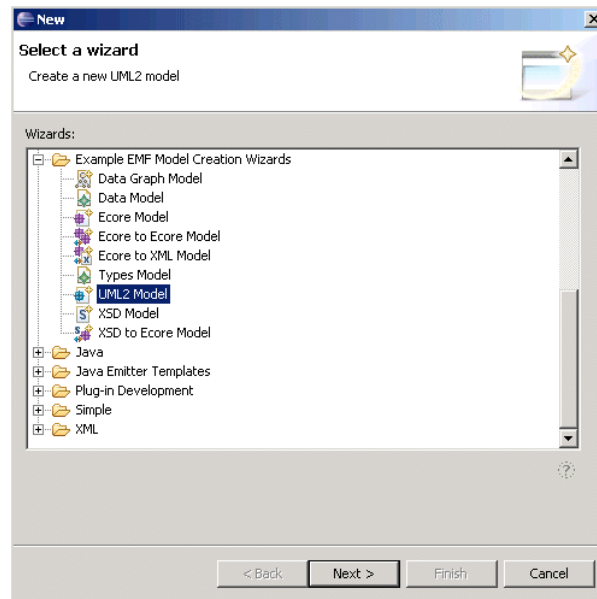
- all jars in *oaw4/oaw-core-4.x.x*, as well as those in its *lib* subdir
- And, **importantly**, you have to add the plugin jars of the following plugins. This is important that the generator finds the Java classes that represent the UML2 metamodel.
 - org.eclipse.uml2
 - org.eclipse.uml2.uml
 - org.eclipse.uml2.uml.resource
 - org.openarchitectureware.uml2.adapter

To tell the oAW Eclipse plugins that this project is a UML2 specific one, you need to specify that in the oAW preferences. Open the project properties, select the openArchitectureWare tab and select the *UML2 profiles* metamodel:

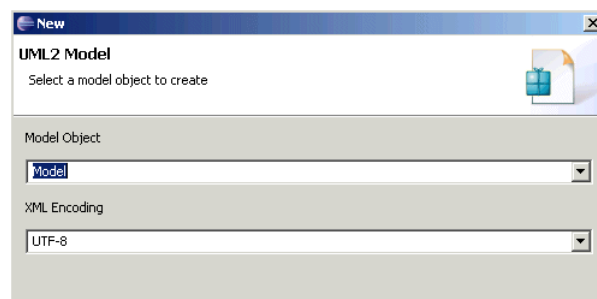


Creating a UML2 Model

You start by defining a uml2 model, i.e. an instance of the UML2 metamodel. In the new Java project, in the source folder, you create a UML2 model that you should call *example.uml2*.

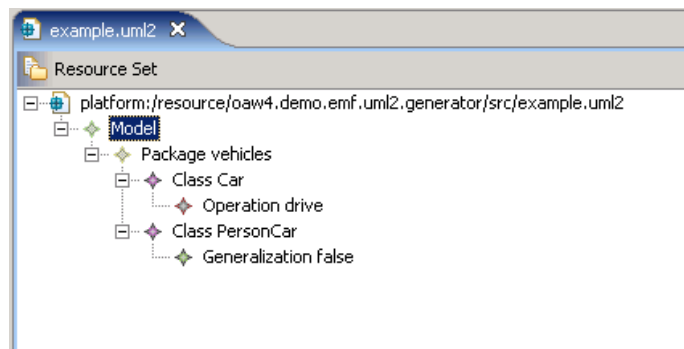


You then have to select the model object. Make sure its a *Model*, not a *Profile*.



Modelling the content

You should then build a model that looks somewhat like this:



By the way, if you rename the *.uml2* file to *.ecore*, you can edit the model using the ecore editors. To inspect the model, they provide a somewhat better view, so you might try!

Code generation

Defining the templates

Inside the source folder of our project, create a *templates* package. Inside that package folder, create a template file *Root.xpt* that has the following content. First, we define the entry template that is called *Root*. Since we expect a UML model element to be the top element to the model, we define it for *uml::Model*. Note the use of the *uml* Namespace prefix, as defined in the UML2 metamodel. Inside that template, we iterate over all owned elements of the model and expand a template for the packages defined in it.

```
«DEFINE Root FOR uml::Model»
  «EXPAND PackageRoot FOREACH ownedElement»
«ENDDDEFINE»
```

In the package template, we again iterate over all owned elements and call a template that handles classes. At this point we expect that only classes are in that package.

```
«DEFINE PackageRoot FOR uml::Package»
  «EXPAND ClassRoot FOREACH ownedType»
«ENDDDEFINE»
```

This template handles classes. It opens a file that has the same name as the class, suffixed by *.java*. Into that file, we generate an empty class body.

```
«DEFINE ClassRoot FOR uml::Class»
  «FILE name+".java"»
  public class <name> {}
«ENDDDEFINE»
```

Defining the workflow

In order to generate code, we need a workflow definition. Here is the workflow file; you should put it into the source folder. The file should be generally understandable if you read the emfExample docs.

```
<?xml version="1.0" encoding="windows-1252"?>
<workflow>
  <property file="workflow.properties"/>

```

You need to setup the UML2 stuff (registering URI maps, Factories, etc.). This can be done declaring a bean in before of the *XmiReader* component:

```
<bean class="oaw.uml2.Setup" standardUML2Setup="true"/>

<component class="oaw.emf.XmiReader">
  <modelFile value="${modelFile}"/>
  <outputSlot value="model"/>
</component>

```

The *XmiReader* reads the model and stores the content (a list containing the model element) in a slot named 'model'. As usual, you might want to clean the target directory.

```
<component id="dirCleaner"
  class="oaw.workflow.common.DirectoryCleaner"
  directories="${srcGenPath}"/>

```

and in the generator we also configure the EMF meta model for oAW, together with the UML2 metamodel package, because the UML2 metamodel refers to it.

```
<component id="generator" class="oaw.xpand2.Generator"
  skipOnError="true">
  <metaModel class="oaw.type.emf.EmfMetaModel">
    <metaModelPackage value="org.eclipse.emf.ecore.EcorePackage"/>
  </metaModel>
  <metaModel class="oaw.uml2.UML2MetaModel"/>
  <expand value="templates::Root::Root FOR model"/>
  <outlet path="${srcGenPath}"/>
    <postprocessor class="oaw.xpand2.output.JavaBeautifier"/>
  </outlet>
</component>

```

Of course, we also need the properties file:

```
modelFile=example.uml2
srcGenPath=src-gen

```

If you run the workflow (by right clicking on the .oaw file and select *Run As -> oAW Workflow* the two Java classes should be generated.

Profile Support

OAW4 is shipped with a special UML2Profiles metamodel implementation. The implementation maps Stereotypes to Types and Tagged Values to simple properties. It also supports Enumerations defined in the profile and Stereotype hierarchies.

!TODO example and explanation!

About our Sponsors

itemis GmbH & Co. KG is an independent IT service company with an emphasis on consulting, coaching, and software development. Every single itemis expert provides many years of project experience and widespread knowledge about all object oriented and component based software development issues - especially in the field of model driven software development.

b+m is the founder of the openArchitectureWare project. The software was originally developed within the scope of many successful projects. b+m opened the software to the community in late 2003. All of the paradigms of Model-Driven Software Development including Product Line Engineering and not only the generator framework have become a key concept for product and customer specific development at b+m. b+m customers can make use of long time experience and substantial know-how in that field. Located at the company headquarters in Melsdorf/Kiel and at its subsidiaries in Berlin, Cottbus, Hamburg, Hanover and Kiel the b+m staff of 205 provides practical solutions for customized business applications, business process optimization and comprehensive architecture, project and quality management.

oose Innovative Informatik GmbH offers coaching, consulting and training in all themes about software engineering. The main focus of their activities are software architecture, requirements engineering and project-management. oose have first-hand information and experience, because our staff take actively part with others in actual trends, standards and innovations. Our staff support this and pass their know-how regularly on by writing and publishing books or being speaker at conferences, etc. Within the OMG oose collaborate actively on the specifications of the UML and also the SysML.

MID Enterprise Software Solutions GmbH is a leading supplier of optimized tool environments for standardsbased and model-centric software development as well as business process modeling. This includes professional tool consulting and tool components to build a complete tool environment using the best techniques and tool modules available - Architectural and Operational Excellence. With innovatorAOX, MID provides a holistic standard tool environment for object- and function-oriented software development as well as business process and data modeling to help its customers establish highly efficient processes and tool environments for software production, The unique and seamless integration of business process modeling into the development process ensures an unprecedented level of convergence of business requirements and implemented IT systems. Project members from all departments speak the same language and all requirements are clearly described.