

Using OCL with oAW and EMF

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

PRIMARY SPONSORS



SECONDARY SPONSOR



Table of Contents

CONFIGURING ECLIPSE AND OAW.....	3
INSTALLING THE PRE-BUILT TUTORIAL.....	3
CREATING A NEW PROJECT	3
DEPENDENCIES.....	3
DEFINING THE CONSTRAINTS.....	4
THE WORKFLOW.....	5
EXAMPLE MODEL.....	5
RUNNING THE EXAMPLE.....	6

Configuring Eclipse and oAW

Before any of the following can work, please make sure that you install the Kent OCL plugins for Eclipse (note that it's not strictly necessary to install the *plugins*, you just need the libraries available). We have tested things with version 1.1 of Kent OCL. You can get the Kent OCL framework from <http://www.cs.kent.ac.uk/projects/ocl/index.html>.

In order to use some of the oAW support for Kent, you should also make sure you have the *adapter.emf.kent_ocl* oAW project in your workspace. See the *Installation* docs for details.

Installing the pre-built tutorial

Instead of building the tutorial manually, you can also download the *oaw-samples-emf-4.x.x* package and install the contained Eclipse projects into your workspace (you might want to delete the *atl* demo project for this first example). To make the projects compile and run, you have to define the following two Eclipse environment variables:

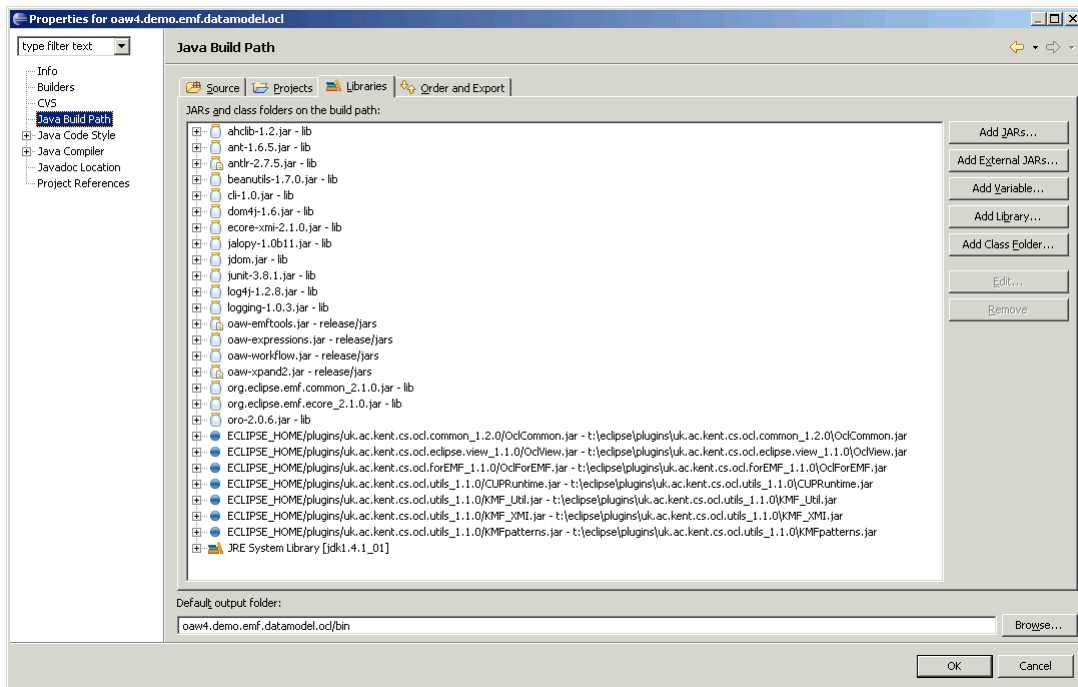
Variable	... points to
OAW_CORE	oaw4/oaw-core-4.x.x
OAW_LIB	oaw4/oaw-core-4.x.x/lib

Creating a new Project

Dependencies

We first create a new Java project. This project must have the following dependencies:

- Project dependency to *oaw4.demo.emf.datamodel* (see *emfHelloWorld* example)
- Project dependency to *adapter.emf.kent_ocl*
- All the Jars from *oaw4/oaw-core-4.x.x/*, also those from the *lib* subdir
- And all jars from all the Kent OCL plugins – see next illustration



Defining the Constraints

As of now, you have to create your own workflow component to check the constraints (we will add more sophisticated integration later). This component must extend the *BasicEmfOclCheckerComponent*.

```
public class DemoOCLChecker extends BasicEmfOclCheckerComponent {
```

You then have to implement the *registerMetamodelPackages* operation by registering the metamodel package(s) with which you want to work. This is basic EMF programming. Here we register our example *data* package from the *emfHelloWorld* example.

```
protected void registerMetamodelPackages(EList model,
    WorkflowContext ctx, ProgressMonitor monitor,
    Logger logger, Issues issues) {
    registerMetamodelPackage( DataPackage.eINSTANCE );
}
```

Now you can define the constraints. In this basic integration scheme, you have to define two things: the constraint expression itself as well as an error message in case it fails. Here we define one constraint that says that Entities must not have Attributes (stupid, but it demonstrates the idea). Note that the constraint is automatically checked against all instances of the context type found in the model.

```
protected void defineConstraints(EList model,
    WorkflowContext ctx, ProgressMonitor monitor,
```

```

        Logger logger, Issues issues) {
    addConstraint(
        "context data::Entity inv:self.attribute->isEmpty()",
        "no attributes allowed" );
    }
}

```

The Workflow

In order to run oAW, we need a workflow file. It looks as follows:

```

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

```

The XMI parser loads the model; see *emfHelloWorld*.

```

  <component id="xmiParser"
    class="org.openarchitectureware.emf.XmiReader">
    <modelFile value="{modelFile}"/>
    <metaModelPackage value="data.DataPackage"/>
    <outputSlot value="model"/>
  </component>

```

Here we simply invoke our newly created workflow component. We pass in the slot name into which the parser put the model.

```

  <component id="ocl" class="ocl.DemoOCLChecker">
    <modelSlot value="model"/>
  </component>
</workflow>

```

Example model

Of course, to check constraints, you need an example model. You might want to use the one from the *emfHelloWorld* example. It's reprinted here, it's the *example.data* file:

```

<?xml version="1.0" encoding="UTF-8"?>
<data:DataModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:data="http://www.openarchitectureware.org/oaw4.demo.emf.datamodel">
  <entity name="Person">
    <attribute name="name" type="String"/>
    <reference target="//@entity.1" name="autos"/>
  </entity>
  <entity name="Vehicle">
    <attribute name="plate" type="String"/>
  </entity>
</data:DataModel>

```

Running the example

If we run the workflow, we get the following output (assuming the Entity in the model *does* have Attributes:

```
27.12.2005 19:10:14 - [INFO] openArchitectureWare v4 -- (c) 2005
                           openarchitectureware.org and contributors
27.12.2005 19:10:14 - [INFO]
-----27.12.2005 19:10:14 - [INFO]
running workflow:
  L:/workspace-oaw4-doku/oaw4.demo.emf.datamodel.ocl/src/workflow.oaw
27.12.2005 19:10:15 - [ERROR] Constraint failed:
      no attributes allowed
      for data.impl.EntityImpl@2bc3f5 (name: Person)
      [/L:/workspace ... tamodel.ocl/src/workflow.oaw]
Errors found, aborting process...
```

So you can see the error message as a consequence of the failed constraint.

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.