The University of Manchester
Jodrell Bank
Observatory

VO-DML Tooling Update

Paul Harrison (JBO)

Contents

- Inspiration
- Java Code
- VO-DML changes

Why

- Focus on model-first development (cf. mapping existing data)
  - desire for more true model re-use
  - desire for more participation (~10 people creating models)
    - Proposal DM (see other talk)
- Why move from ant to gradle?
  - Java compilation conventions
  - Maven dependency repositories
  - allows easy start-up, modular, model development

Where

- Code moved to https://github.com/ivoa/vo-dml
  - changes described here are actually on a fork, PR#2

Initial gradle tooling #2

Open **pahjbo** wants to merge 20 commits into `ivoa:master` from `pahjbo:gradle_integration`

Conversation 1 | Commits 20 | Checks 0 | Files changed 85

pahjbo commented on 5 Aug — Member

this PR adds Gradle tooling in parallel with the current ant tooling (i.e. the old stuff still works) The gradle tooling has a fairly large fraction of the functionality of the ant tooling (only really missing the conversion from xmi metamodels). It has significant advantages for java code generation and compilation as that is the core functionality of gradle. The other major advantage is that you can just checkout the project and run gradlew (i.e. you do not have to install anything), and if you have IntelliJ IDEA or eclipse, you will get good IDE integration too.

The medium term idea would be to delete the ant tooling, as the gradle tooling makes it easy to have a self contained project per model, which is a better way to work.

1

How

- Most of the work is done with XSLT2.0 transformations of the VO-DML
  - have restructured this - more use of functions
  - removed absolute URLs - use XML Catalogues
  - altered the modularity
- gradle plugin written in Kotlin.

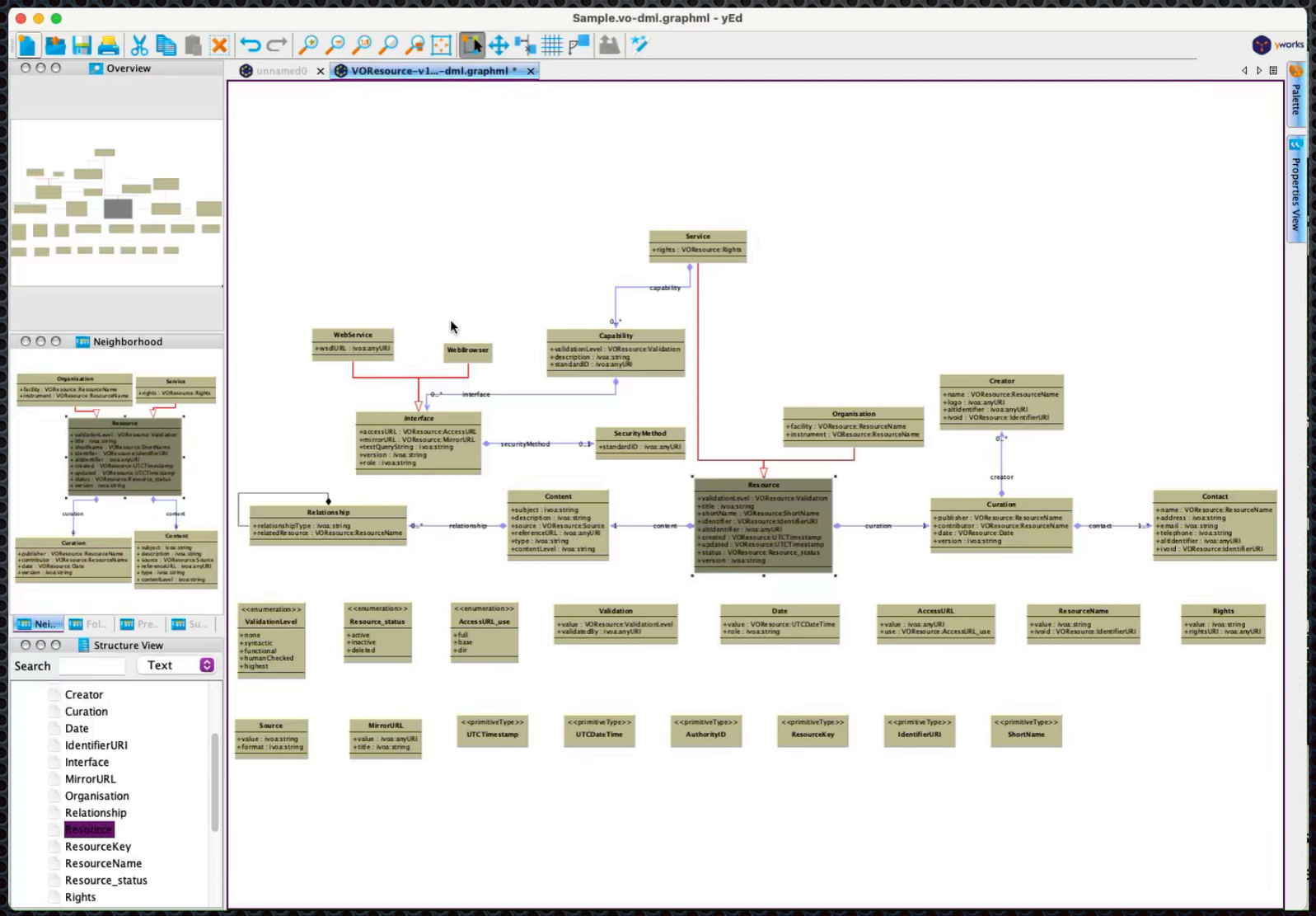
Gradle Plugin

```
plugins {
    id("net.ivoa.vo-dml.vodmltools") version "0.1"
}
```

- Have released v0.1 of plugin
- self contained - all that is needed is a reference to the plugin
  - all dependencies (including schematron) will be auto-downloaded
- tasks
  - vodmlValidate - runs validation on the models.
  - vodmlDoc - generate standard documentation.
  - vodmlGenerateJava - generate java classes - can publish jar
- https://github.com/pahjbo/vo-dml/blob/gradle_integration/tools/gradletooling/ReadMe.md

Diagram generation

- As well as the standard diagram generation using graphviz - graphML which can be hand-edited in yEd also generated now



Java Generation

- Original was done as part of VO-URP - I have resurrected and extended this code.
  - Generating POJOs with JAXB and JPA annotations
    - Flexible instance construction
    - Uses annotations to record the VO-DML meta-information
- Relies on a “runtime” library for some common functionality
- see https://github.com/pahjbo/vo-dml/blob/gradle_integration/tools/JavaCodeGeneration.md

VO-DML issues

- There are some VO-DML updates that I would make - all backwards compatible
  - remove (make optional) the redundant <name> element from <import> (it was used by schematron rules)
    - I think the same is true for <documentationURL>
  - review some of the schematron rules
    - e.g. used twice in composition

VO-DML extension - Natural Keys

- ORM uses surrogate keys widely - however, in the model it is sometimes better to use a “natural key” i.e. an existing attribute - often the case for the target of “references”.

```
<xsd:complexType name="NaturalKey">
  <xsd:annotation>
    <xsd:documentation>
      This constraint is used to indicate that an attribute is a natural key for its owning ObjectType, meaning that the
      attribute value should be globally unique. This may be applied multiple times to indicate that only a composition
      of several attributes make the globally unique key.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="Constraint">
      <xsd:sequence>
        <xsd:element name="Position" type="xsd:positiveInteger">
            <xsd:annotation>
              <xsd:documentation>In the case where multiple attribute values make up the natural key, this
              value indicates the ordinal number of this particular key in the compound key.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Base model extension

- In a similar fashion to the “natural key” it is sometimes useful to be able to identify attribute as a “key into another system”

```
 abstract primitive identifier "something that an identifier that
can be used as a key for lookup of an entity that is *outside this
datamodel*"
   primitive intIdentifier -> identifier "an integer identifier"
   primitive stringIdentifier -> identifier "a string identifier"
   primitive ivorn -> identifier "an identifier that can be used as
a key to look up in an IVOA registry - see https://www.ivoa.net/
documents/IVOAIdentifiers/"
```

vodsl

TODO

- Merge the pull request! (The gradle plugin itself is pretty much complete)
  - Delete the ant tooling (remove possible confusion).
- Improve some details in the JAX/JPA mapping e.g.
  - subsets should create substitution group XML.
  - JPA embeddable support patchy - using Hibernate at moment.
  - Array support (has some VO-DML implications too…)
  - …
- Python code generation (Volunteers?), C++…
- TODOs actually managed as usual with GitHub issues