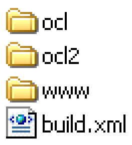
CVS Directory Structure of the Dresden OCL(2) Toolkit

1. General CVS structure

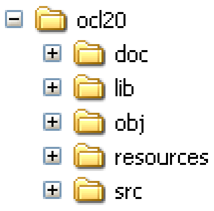


```
ocl
ocl2
www
build.xml
```

Every toolkit (version 1 and version 2) has its own directory. Everything belonging to a toolkit should be inside the according directory. The web-site is placed in the directory *www*. There is an ant script which can build both toolkits using the inner ant scripts.

2. Structure of Dresden OCL2 Toolkit

2.1 Top level



```
⊟ ocl20
    ⊞ doc
    ⊞ lib
    ⊞ obj
    ⊞ resources
    ⊞ src
```

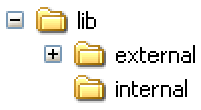
• ***doc***	contains all documentation files including generated JavaDoc
• ***lib***	contains all librarys including generated
• ***obj***	contains compiled classes
• ***resources***	contains all resource files
• ***src***	contains all source files including generated sources^

This directory contains the main readme file and ant build files.
Note that generated files are only stored locally, not in CVS.

2.2 doc

This directory contains all documentation for the Dresden OCL2 Toolkit. Every Tool should put its documentation files into a seperate directory, i.e. *doc/core/parser,* which is similar to the package structure in *src* directory. JavaDoc will be generated into *doc/api.*

2.3 lib



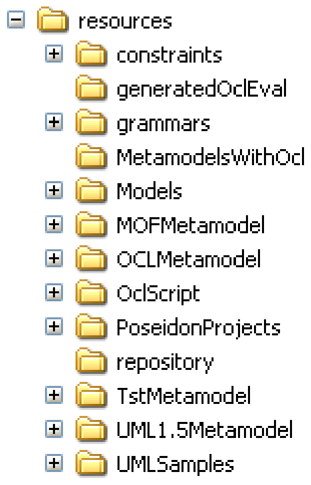
```
⊟ lib
    ⊞ external
      internal
```

The *lib* directory is separated into two parts. All external libraries are stored in the *external* subdirectory. Parts of the toolkit, which are librarys too, are stored in the *internal* subdirectory.

2.4 obj

All compiled classes are stored in this directory.

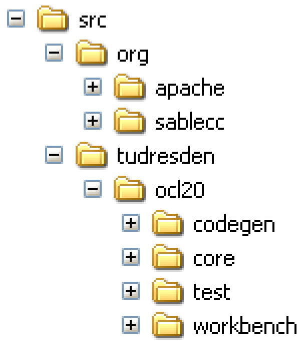
2.5 resources



```
⊟ resources
    ⊞ constraints
      generatedOclEval
    ⊞ grammars
      MetamodelsWithOcl
    ⊞ Models
    ⊞ MOFMetamodel
    ⊞ OCLMetamodel
    ⊞ OclScript
    ⊞ PoseidonProjects
      repository
    ⊞ TstMetamodel
    ⊞ UML1.5Metamodel
    ⊞ UMLSamples
```

This directory contains all resource files. These include grammars, repository files, metamodels, sample files and all other files, that don't belong to one of the other directories.

2.6 src



```
⊟ src
    ⊟ org
        ⊞ apache
        ⊞ sablecc
    ⊟ tudresden
        ⊟ ocl20
            ⊞ codegen
            ⊞ core
            ⊞ test
            ⊞ workbench
```

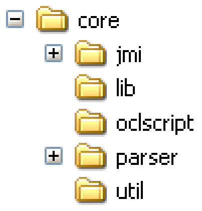
All project sources are contained in the *src* directory. The subdirectories are formed as usual by the package structure.

2.6.1 General package structure

The topmost package of all project packages is *tudresden.ocl20*, if possible. Because of the use of some external frameworks it is necessary to put parts of them in different packages, i.e. *org.sablecc* for the extended sablecc parser generator.

The subpackages of *tudresden.ocl20* belong to the different tools. There are two basic packages, core and codegen, which have several subpackages. All future tools should be created in one of these packages or as direct subpackage of *tudresden.ocl20*.

2.6.2 Package tudresden.ocl20.core



This package contains the core components of the toolkit. These are some interfaces and libraries and of course the parser.

2.6.3 Package turdresden.ocl20.codegen

This Package contains all code generators, i.e. for java or sql, in a separate subpackage.

3. Rules for committing files

3.1 Source files are always committed to the *src* directory. Choose a package name respecting the order mentioned in 3.6.

3.2 If there is a “test GUI” of your tool, put this under your main tool package. Best name for this is *gui* or *testgui*.

3.3 If you use external libraries, put them into the *libs.external* directory.

3.4 Put all files that are no libraries and no source into a subdirectory of the *resources* directory named like the corresponding tool.

3.5 Create an ant script for compiling your tool and include it into the main ant script.