CDK Developer's Guide

by Egon L. Willighagen

CDK Developer's Guide

by Egon L. Willighagen

Table of Contents

Preface

This guide gives information on how the CDK library is being developed. It contains links to software that is needed to compile the software, or to create packages.

It also includes guide lines on how to write documentation about the software and as JavaDoc in the software. None of this is currently enforced or required.

Documentation

Chapter 1. JavaDoc

Most of the documentation about the CDK classes is done in the java source files itself. The format used for this is the JavaDoc format. An overview/introduction can be found at http://java.sun.com/j2se/javadoc/writingdoccomments/ [http://java.sun.com/j2se/javadoc/writingdoccomments/].

JavaDoc documentation is now generated using Javadoc and Ant:

```
cdk/$ ant -buildfile javadoc.xml html
```

JavaDoc Tags

The two sub sections contain suggested policy and is not yet as such approved.

Tagging classes

Java classes should have the JavaDoc tags @copyright and @author.

Tagging procedures

Procedures of classes should give information about what they do and information about parameters (@param) and things they return (@return).

If the procedure performs a method found in literature, it should mention the reference. No format for the reference has been determined, but the BibTex format should do fine.

DocbookDoclet

The JavaDoc documentation can be converted into DocBook code with the DocBookDoclet (http://freshmeat.net/projects/dbdoclet/ [http://freshmeat.net/projects/dbdoclet/]). The DocBook output can in turn be generated into PDF files.

Chapter 2. CDK Programming Tutorial

There is a sort of CDK Programming Tutorial online. It is written in DocBook XML 4.1.2 (like most of our documentation).

The CDK Library

Chapter 3. The Goal

The goal of the CDK is to be a library of java classes to support java software to solve typical chemical computational and informational problems.

Chapter 4. Class categories

All CDK classes are categorized into four classes: core, standard, extra, and project.

Core classes

Core classes represent chemical entities, like bonds, atoms, molecules, and more abstract entities, like connectivity tables, used for in common computational and information chemistry.

Standard classes

Standard classes are all classes that do not represent some entity, but are common to many applications dealing with chemical problems. These classes deal with manipulation and other ways of dealing with the core classes. These can be further grouped into two subclasses. One class changes the content of the entities, the other does not.

Example classes that do not change the content of core classes, are IO classes and viewers. In this subclass belong also the classes that generate instances of core classes, example classes are classes that generate 3D coordinates, for example.

The other subclass contains classes that change the content of the core classes. E.g. classes implementing a molecular dynamics run and editors.

This subclassification seems to be too artificial.

Extra classes

Classes that are too specific for general use belong to the class Extra.

Project classes

Java classes and packages in this category are not really part of the CDK library, but are aimed to help manage the project, by ways of testing and generating documentation and project statistics.

Chapter 5. Compiling the CDK source

Compiling and jar-ing the software is done with Jakarta's Ant [http://jakarta.apache.org/ant/].

CDK can be compiled with and without support for Java3D. How to compile these, is explained in the next two sections.

Compiling without Java3D installed

If not Java3D is installed or available for your platform, you can compile CDK with:

```
cdk/$ ant
```

Compiling with Java3D installed

If Java3D is installed or available for your platform, you can compile CDK with:

```
cdk/$ ant -Dpathtojava3d=/some/path compile-with-java3d
```

The parameter pathtojava3d can be used to tell Ant where your Java3D installation (read: jar files) can be found.

Chapter 6. Testing CDK classes

JUnit testing

After you compiled the code, you can do **ant test** to run the test suite of non-interactive, automated tests. You will get output with a part something like:

```
test:
    [junit] Running org.openscience.cdk.test.CDKTests
    [junit] Tests run: 21, Failures: 5, Errors: 0, Time elapsed: 9.63 sec
    [junit] TEST org.openscience.cdk.test.CDKTests FAILED
```

These tests results in some serious problems.

The actual tests are done by JUnit [http://www.junit.org/] which you must have installed. JDK 1.4 requires version 3.7, but 3.6 is good for other Java versions.

A recent alternative is to install the cdk-test package. You can then do tests with

```
$ cdk-test org.openscience.cdk.test.CDKTests
```

There are currently two major test suites:

- org.openscience.cdk.test.CDKTests
- org.openscience.cdk.test.io.cml.CMLIOTests

Interactive tests

There are also run interactive tests, like the Controller2DTest. In order to try them, you can edit the "run" target in the build.xml file to look like this:

```
          <target name="run" depends="dist">
                <java classname="org.openscience.cdk.test.ControllerTest"
fork="yes">
                        <arg value=""/>
                        <classpath>
                                <pathelement location="${dist}/jar/cdk.jar"/>
                                <pathelement path="${java.class.path}"/>
                                <pathelement location="."/>
                                <fileset dir="jar">
                                <include name="*.jar"/>
                                </fileset>
                        </classpath>
                </java>
          </target>
```

Then, a **ant run** should give you a window where you can add bonds to a given structure.

The CDK Website

Chapter 7. HTML Generation

The website is written as DocBook XML 4.1.2 fragments and compiled into HTML with the Website DTD 2.0.

Binary Packages

Chapter 8. Tar packages

Currently only tar.gz packages are available. The packages can be found in CVS /cdk/packages/tar. These packages can be installed on UNIX type operating systems in a few steps (replace the 20020206 with the latest release date):

```
$ gunzip cdk-libs-20020206.tar.gz
$ tar xvf cdk-libs-20020206.tar
$ cd cdk-libs-20020206
$ ./configure && make && install
```

Remember the correct order of installation is: cdk-libs, cdk-bin, cdk-test.