# **Code coverage: making sure your code is tested**



Published November 28, 2006

#### Citation

Willighagen, E. (2006, November 28). Code coverage: making sure your code is tested. *Chem-blaics*. https://doi.org/10.59350/gtt31-tee03

### Keywords

Opensource, Cdk, Junit

#### **Abstract**

Recently I discussed JUnit testing from within Eclipse, and blogged at several occasions about it in other situations. I cannot stress enough how useful unit testing is: it adds this extra set of eyeballs to make bugs shallow. And it does that, indeed.

# Copyright

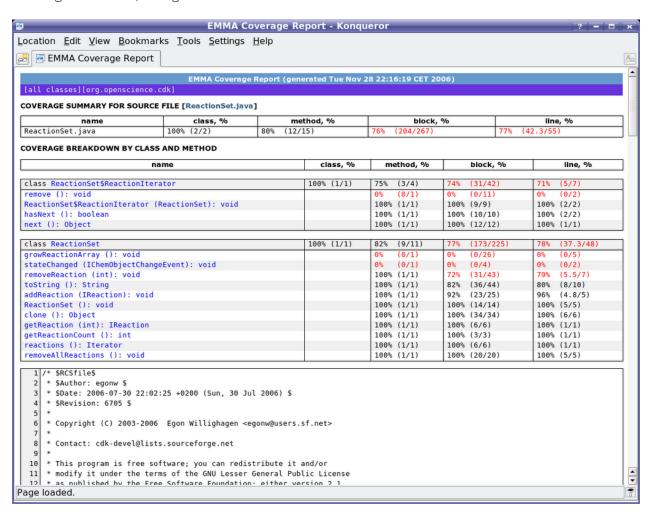
Copyright © None 2006. Distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

#### chem-bla-ics

Recently I discussed JUnit testing from within Eclipse, and blogged at several occasions about it in other situations. I cannot stress enough how useful unit testing is: it adds this extra set of eyeballs to make bugs shallow. And it does that, indeed.

Ensuring that you actually test all the code you write, however, is not easy. A couple of years back I read an article about Hansel, which does code coverage checking, but never got it nicely working for the CDK project. Never looked at that lately, so no idea how the current release would work out. Hansel is an extension of JUnit, and requires hard coding class names, which conflicts with CDK's module setup.

Thomas Kuhn pointed me last week to Emma, which seems a nice tool. It does not require hacking our source, and generates cool HTML:



And even highlights the source code:

## chem-bla-ics

```
EMMA Coverage Report - Konqueror
<u>L</u>ocation <u>E</u>dit <u>V</u>iew <u>B</u>ookmarks <u>T</u>ools <u>S</u>ettings <u>H</u>elp
EMMA Coverage Report
                  constructs an empty reactionset.
                                                                                                                                                  •
  89
              public ReactionSet() {
  90
                      reactionCount = 0;
  92
                      reactions = new Reaction[growArraySize];
  93
  95
  96
  97
               * Adds an reaction to this container.
  98
  99
 100
               * @param reaction The reaction to be added to this container
 101
 102
              public void addReaction(org.openscience.cdk.interfaces.IReaction reaction) {
                      if (reactionCount + 1 >= reactions.length) growReactionArray();
 103
                      reactions[reactionCount] = reaction;
 104
 105
                      reactionCount++;
                      notifyChanged();
 106
 107
 108
 109
 110
               * Remove a reaction from this set.
 111
               st @param pos The position of the reaction to be removed.
 112
 113
 114
              public void removeReaction(int pos) {
              reactions[pos].removeListener(this);

for (int i = pos; i < reactionCount -
 115
 116
 117
                              reactions[i] = reactions[i + 1];
 118
 119
                      reactions[reactionCount - 1] = null;
 120
                       reactionCount--;
 121
                      notifyChanged();
 122
 123
 124
 125
 126
               st Returns the Reaction at position <code>number</code> in the
 127
               * container.
 128
```

BTW, I seem to be in good company: Classpath is using it too.

Below is the command I issued to generate the HTML output. Rajarshi, maybe this can be integrated into Nightly? Note that it only runs the tests for the data module:

```
ant dist-large dist-test-large
java -cp ~/tmp/emma-2.0.5312/lib/emma.jar emmarun \
   -cp develjar/junit.jar:dist/jar/cdk-svn-20061128.jar:dist/jar/cdk-test-
svn-20061128.jar \
   -r html -sp src junit.textui.TestRunner org.openscience.cdk.test.MdataTest
```