

Code coverage: making sure your code is tested

Egon Willighagen 

Published November 28, 2006

Citation

Willighagen, E. (2006). Code coverage: making sure your code is tested. In *chem-bla-ics*. chem-bla-ics. <https://doi.org/10.59350/gtt31-tee03>

Keywords

Opensource, Cdk, Junit

Copyright

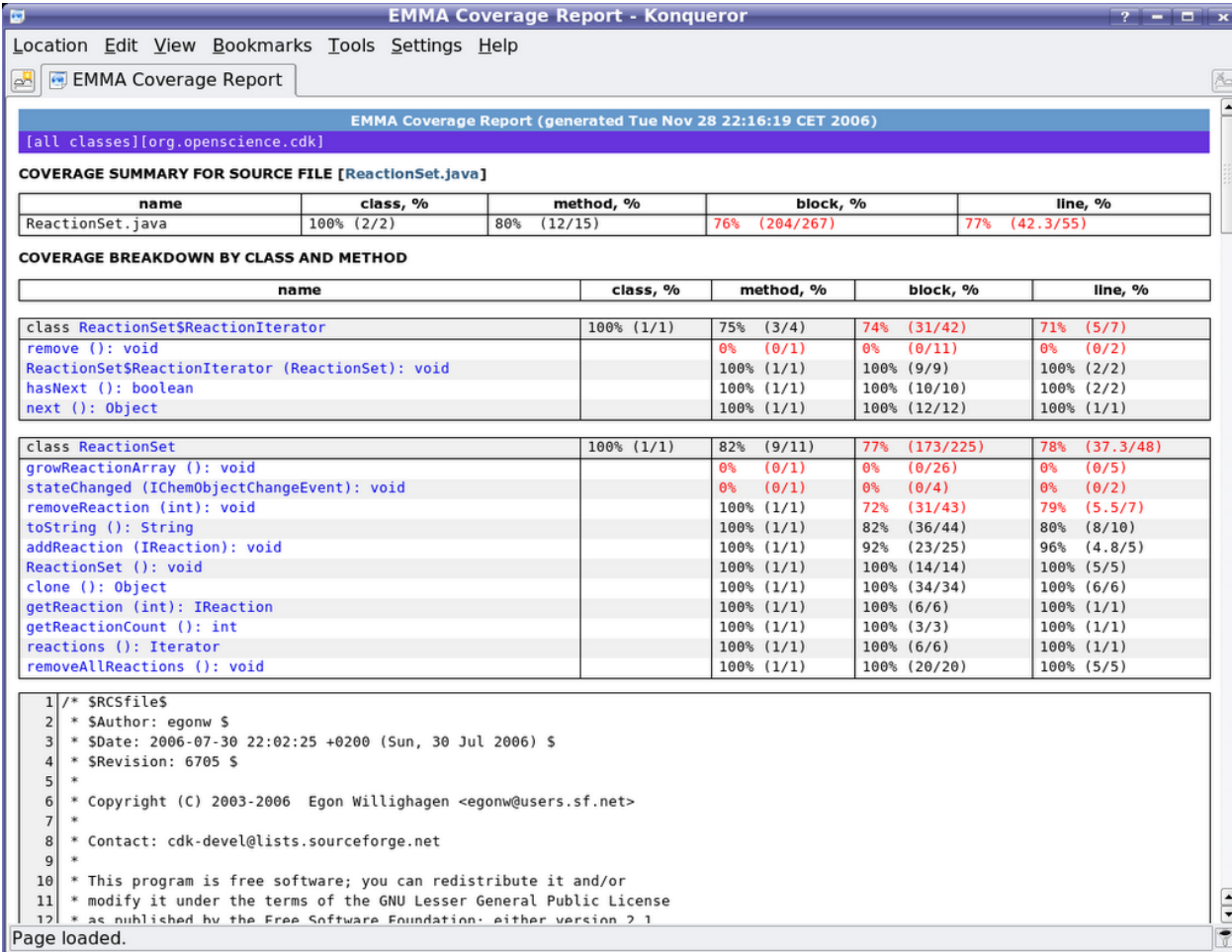
Copyright © Egon Willighagen 2006. Distributed under the terms of the [Creative Commons Attribution 4.0 International License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

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:



The screenshot shows the EMMA Coverage Report - Konqueror window. The report is generated on Tue Nov 28 22:16:19 CET 2006 for the source file ReactionSet.java. The coverage summary shows 100% class coverage (2/2), 80% method coverage (12/15), 76% block coverage (204/267), and 77% line coverage (42.3/55). The breakdown by class and method is as follows:

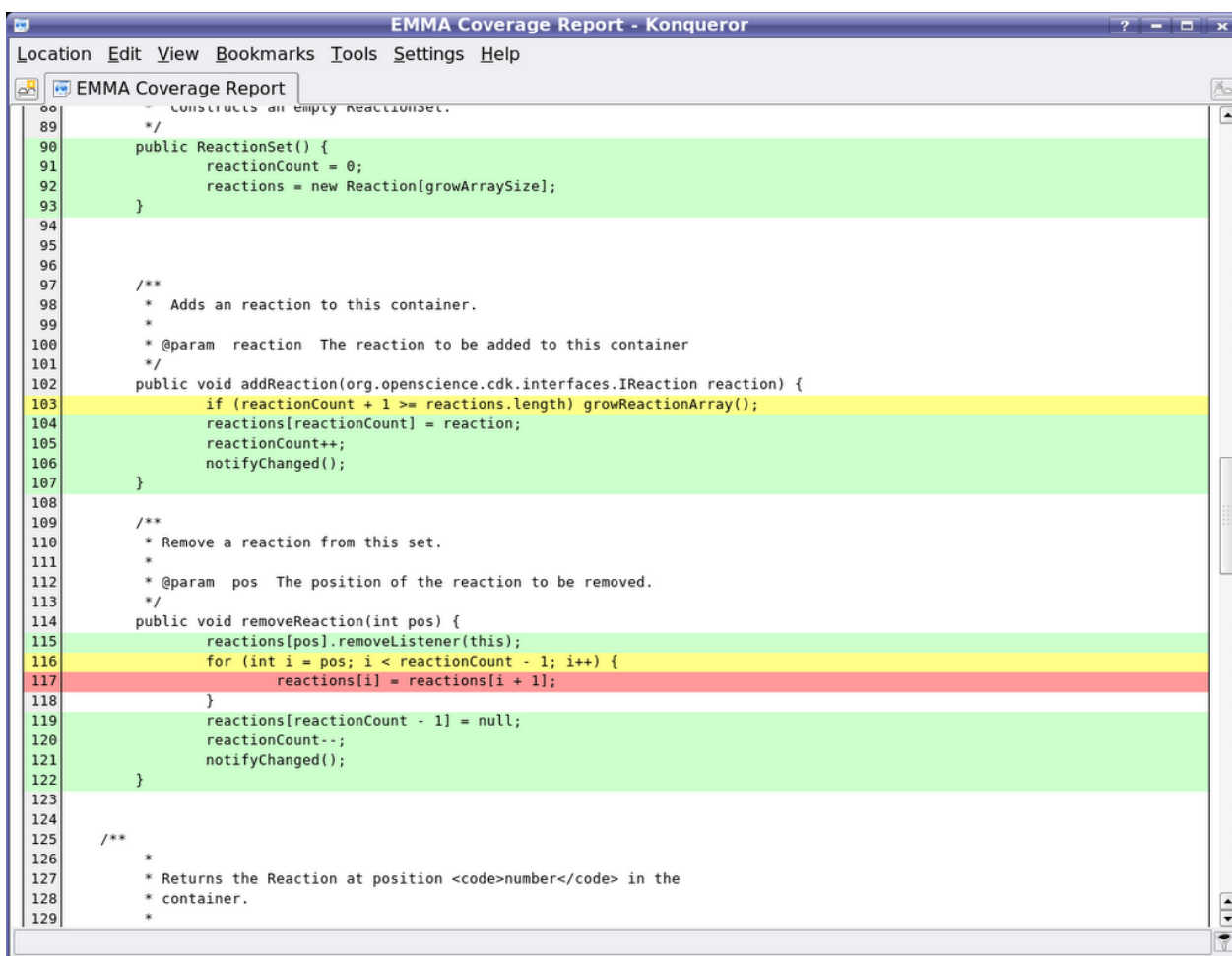
| name | class, % | method, % | block, % | line, % |
|---|------------|------------|---------------|---------------|
| class ReactionSet\$ReactionIterator | 100% (1/1) | 75% (3/4) | 74% (31/42) | 71% (5/7) |
| remove (): void | | 0% (0/1) | 0% (0/11) | 0% (0/2) |
| ReactionSet\$ReactionIterator (ReactionSet): void | | 100% (1/1) | 100% (9/9) | 100% (2/2) |
| hasNext (): boolean | | 100% (1/1) | 100% (10/10) | 100% (2/2) |
| next (): Object | | 100% (1/1) | 100% (12/12) | 100% (1/1) |
| class ReactionSet | 100% (1/1) | 82% (9/11) | 77% (173/225) | 78% (37.3/48) |
| growReactionArray (): void | | 0% (0/1) | 0% (0/26) | 0% (0/5) |
| stateChanged (IChemObjectChangeEvent): void | | 0% (0/1) | 0% (0/4) | 0% (0/2) |
| removeReaction (int): void | | 100% (1/1) | 72% (31/43) | 79% (5.5/7) |
| toString (): String | | 100% (1/1) | 82% (36/44) | 80% (8/10) |
| addReaction (IReaction): void | | 100% (1/1) | 92% (23/25) | 96% (4.8/5) |
| ReactionSet (): void | | 100% (1/1) | 100% (14/14) | 100% (5/5) |
| clone (): Object | | 100% (1/1) | 100% (34/34) | 100% (6/6) |
| getReaction (int): IReaction | | 100% (1/1) | 100% (6/6) | 100% (1/1) |
| getReactionCount (): int | | 100% (1/1) | 100% (3/3) | 100% (1/1) |
| reactions (): Iterator | | 100% (1/1) | 100% (6/6) | 100% (1/1) |
| removeAllReactions (): void | | 100% (1/1) | 100% (20/20) | 100% (5/5) |

The source code is displayed below the tables, with line numbers 1 through 17. The code includes a header with author, date, and license information.

```
1 /* SRCsfile$
2 * $Author: egonw $
3 * $Date: 2006-07-30 22:02:25 +0200 (Sun, 30 Jul 2006) $
4 * $Revision: 6705 $
5 *
6 * Copyright (C) 2003-2006 Egon Willighagen <egonw@users.sf.net>
7 *
8 * Contact: cdk-devel@lists.sourceforge.net
9 *
10 * This program is free software; you can redistribute it and/or
11 * modify it under the terms of the GNU Lesser General Public License
12 * as published by the Free Software Foundation; either version 2.1
```

And even highlights the source code:

chem-bla-ics



```
EMMA Coverage Report - Konqueror
Location Edit View Bookmarks Tools Settings Help
EMMA Coverage Report
88      * Constructs an empty ReactionSet.
89      */
90      public ReactionSet() {
91          reactionCount = 0;
92          reactions = new Reaction[growArraySize];
93      }
94
95
96
97      /**
98       * Adds an reaction to this container.
99       *
100      * @param reaction The reaction to be added to this container
101      */
102      public void addReaction(org.openscience.cdk.interfaces.IReaction reaction) {
103          if (reactionCount + 1 >= reactions.length) growReactionArray();
104          reactions[reactionCount] = reaction;
105          reactionCount++;
106          notifyChanged();
107      }
108
109      /**
110       * Remove a reaction from this set.
111       *
112       * @param pos The position of the reaction to be removed.
113       */
114      public void removeReaction(int pos) {
115          reactions[pos].removeListener(this);
116          for (int i = pos; i < reactionCount - 1; i++) {
117              reactions[i] = reactions[i + 1];
118          }
119          reactions[reactionCount - 1] = null;
120          reactionCount--;
121          notifyChanged();
122      }
123
124
125      /**
126       *
127       * Returns the Reaction at position <code>number</code> in the
128       * container.
129       *

```

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
```