# Using FindBugs to create a significantly smaller number of erroneous code

## programming help stack

Some time ago, the java monitor user, the forum of our JCG partner, Kisa Yana, noticed that his system would force a large number of complete garbage assemblies, despite the fact that the overall use of memory was low.Approximate assessment of the cause of the problem assumed the possible System.gc () call, executed by one of the libraries used.Let's see what our partner will tell the elimination of such problems to create a significantly less erroneous code.

... It occurred to me that there is a tool that could help with the problem of System.gc (). It is successfully called FindBugs. In fact, their several (PMD and Checkstyle are other, similar tools), but FindBugs is good because you really do not need an application source code for verification for errors. All you need is JAR files and / or classes files. Another wonderful feature is that each report is very well documented. You do not just get troubled warnings from which you refuse a few minutes, but you can find an explanation for each warning online.

Here is an example of output FindBugs when you run it for the Java monitor's probe.

01.

02.

03.

04.

05.

06.

07.

08.

09.

ten

eleven

12

$ FindBugs -TextUI -EFFORT: Max Java-Monitor-Probes / Build / Web-Inf / Classes

The Following Classes Needed for Analysis Were Missing:

javax.servlet.http.httpservlet.

javax.servlet.filter.

javax.servlet.http.httpservletresponse

javax.servlet.servletexception

javax.servlet.servletconfig

javax.servlet.filterconfig

javax.servlet.servletrequest

Javax.Servlet.ServletResponse

Javax.Servlet.Filterchain.

Missing Classes: 7

From this report you can see that FindBugs cannot find any problems with the code itself. There are several classes from which the code depends. They were not included in the analysis. Since these are all classes provided by Sun, I assume that they do not contain errors. *cough*

Anyway. Let's make the analysis a little more juicy. Here is the output when you start FindBugs against the well-known MYSQL JDBC driver. I am sure that many of you today use it in production. This code gives significantly more results, and FindBugs takes some time to analyze the entire package. I showed only a few rows from 154 warnings.

01.

02.

03.

04.

05.

06.

07.

08.

09.

ten

eleven

12

thirteen

14

15

16

17.

$ FindBugs -TextUI -EFFORT: Max MySQL-Connector-Java-5.1.5-Bin.jar

.....

H C NP: Method Call In Com.Mysql.jdbc.profiler.ProfileRevent.Pack () Passes Null for Unconditionally Dereferenced Parameter of WriteBytes (Byte [], Byte [], Int) Method Invoked At ProfileRevent.java :[line 375]

.....

M D NP: Possible null pointer dereference of s1 on path that might be infeasible in com.mysql.jdbc.connectionimpl.jdbc.connectionimpl.nullsafecompare (String, String) dereferenced at connectimpl.java :[line 341]

.....

M C NP: Method Call in com.mysql.jdbc.databasemetadata.getInstance (ConnectionImpl, String) Passes Null for Unconditionally Dereferenced Parameter of New Databasemetadata

(ConnectionImpl, String) Method Invoked At Databasemetadata.java :[line 632]

.....

H S SQL: Method com.mysql.jdbc.databasemetadata.getcolumnprivileges (String, String, String, String) Passes a nonconstant String to An Execute Method on an SQL Statement at databasemetadata.java :[line 2156]

H S SQL: Method com.mysql.jdbc.databasemetadata.gettableprivileges (String, String, String) Passes a nonconstant String to An Execute Method on an SQL Statement at databasemetadata.java :[line 4638]

M S SQL: Method com.mysql.jdbc.connectionImpl.setSessionVariables () Passes a nonconstant string to an Execute Method On An SQL Statement at ConnectionImpl.java :[line 5074]

.....

M M IS: Inconsistent Synchronization of com.mysql.jdbc.callablestatement.outPutParameterResults; Locked 50% of Time Unsynchronized Access At CALLABLESTATEMENT.JAVA :[Line 1948]

M M M M: Inconsistent Synchronization Of Com.Mysql.jdbc.statementImpl.wascancelledByTimeout; Locked 83% of Time Unsynchronized Access AT PreparedStatement.java :[line 1756]

.....

WARNINGS GENERATED: 154

.....

Learning to read the output of FindBugs takes a little time. What I do is simply working as a certain error or a warning when I get tired or disappointed with the code that I have to write. This is my procrastination. ??

I only chose what it seems problematic to me as a developer: "Skip zero for an unconditional risen parameter." IR. NullPointerexception Someone? Of course, it can be a test code or even unused code that is still under development. How about this: "transmits a non-permanent string to the Execute method for the SQL statement." Hmm If this checkbox is not installed, it may be the cause of SQL injection vulnerability.

Earlier, I said that FindBugs does not require you to access the source code of the application to find errors in it.Just for laughter, let's look at one of the cornerstone of our Java EE applications: Driver Oracle JDBC.

01.

02.

03.

04.

05.

06.

07.

08.

09.

ten

eleven

12

thirteen

$ FindBugs -TextUI -EFFORT: MAX OJDBC6.JAR

.....

M B DM: Oracle.sql.converterarchive.openarchiveForRead () Invokes System.exit (...), Which Shuts Down The Entire Virtual Machine At ConvertRerarchive.java :[line 375]

M B DM: Oracle.sql.converTeRchive.CloseArchiveForRead () Invokes System.exit (...), Which Shuts Down The Entire Virtual Machine At Converterarchive.java :[line 390]

.....

M B ES: COMPARISON OF STRING OBJECTS USING == OR! = IN Oracle.jdbc.connector.orCleConnectionRequestinfo.equals (Object) at oracleconnectionRequestinfo.java :[line 104]

.....

H C IL: There Is An Apparent Infinite Recursive Loop In

Oracle.jdbc.RowSet.oracleCachedRowSet.updateBlob (Int, InputStream, Long) at oraclecchedumet.java :[line 6365]

H C IL: THERE IS An Apparent Infinite Recursive Loop In Oracle.jdbc.Rowset.oracleCachedRowset.UpdateClob (INT, READER, LONG) at oraclecchedurowset.java :[line 6445]

H C IL: There IS An Apparent Infinite Recursive Loop In Oracle.jdbc.Rowset.orCleCachedRowset.Updatenclob (INT, READER, LONG) at oraclecchedurowset.java :[line 6535]

.....

Warnings Generated: 1028

.....

This driver issues at least 1028 warnings. Wow. I do not argue that the Oracle JDBC driver is actually a bad code fragment. I just find it smells like a little. ?? Oracle developers may want to deal with the elimination of warnings about the FindBugs message. There are many small suggestions on performance and stability.

And yes: FindBugs checks the use of System.gc ().

PS. Please keep in mind that I used FindBugs 1.3.7. In this version, FindBugs has an error due to which it generates false responses to clean the database resources.

PPS. Who I'm kidding: I think the driver Oracle is bad. Infinite cycles? System.exit ()? You are welcome.

Always help ... needy codeme ??

Byron.

Articles on the topic:
Things that each programmer should know
10 tips for proper registration of applications
Software development laws
Java Best Practices series
9 Survival Soviets in the Development of Wild West