creating unit tests (semi-)automatically?

Asked 15 years, 11 months ago Modified 15 years, 1 month ago Viewed 1k times

6

Is there a framework that supports generating some standard unit tests from annotations? An example of what I have in mind would be:

```
@HasPublicDefaultConstructor
public class Foo {

}
```

This would obviously be used to automatically generate a unit test that checks whether Foo has a default constructor. Am I the only person who thought of something like that yet? ;) While I'm most interested in Java, solutions in other languages would certainly be interesting, too.

EDIT: In response to S.Lott's answer, let me clarify:

I'm trying to test whether the class has a default constructor. (Of course that is just an example.) I could just do so by writing a test, but I find that quite tedious. So I'm looking for a tool that would process the annotations at compile time (via APT) and generate the test for me. Does something like that exist? If not, do you think it is a good idea?

java unit-testing automation code-generation annotations

Share
Improve this question
Follow

edited Dec 28, 2008 at 19:11

asked Dec 28, 2008 at 16:04

Kim Stebel
42k ● 12 ● 129 ● 142

11 Answers

Sorted by: Highest score (default)

7

The question is whether "polluting" the production code with Unit Testing information in the form of additional annotations is really such a good idea. Personally I'd vote against it.

Share Improve this answer Follow

answered Dec 28, 2008 at 19:18

cdecker
4,667 ● 8 ● 50 ● 81

Apart from what Snyke indicated - shouldn't developers pay even more attention when writing unit tests? By using generated tests or half-ass unit tests you can expect quarter-ass functionality. ;-) In the best case you can expect unit test skeletons so that you achieve higher code coverage faster. – Shonzilla Dec 30, 2008 at 0:34

2

It sounds like you are looking for a programming language with more declarative expressive power than standard Java. The tests you postulate fill in the gaps until the compiler can check the semantics of the declarations.

I don't know of any tool that converts from the kind of annotations you suggest into automated tests. It sounds like a nice TDD exercise, though, especially with the recent(-ish) compiler APIs.

Share Improve this answer Follow

answered Dec 29, 2008 at 21:27

Kent Beck

5,011 ● 2 ● 31 ● 31

2

tl;dr: Use code inspection tools instead. They can do what you want, don't pollute your code and can be easily built into an automatic build process.

This doesn't sound like something that should be handled with a unit test. If you are adamant that every class must have a default constructor defined, a static code analyzer might be a better bet. PMD definitely has a detector for this sort of thing. You can also force this sort of detector with Checkstyle.

Both of those are easily customizable and can be made a part of your continuous integration process right along side your unit tests.

Most critical point: This would allow you to implement the functionality that you want without polluting the code base with extra annotations. That's the major tripping point for me as I look at this idea: you've created a huge software management problem.

Just imagine the situation of modifying a class with a default constructor to be immutable (thus requiring that the objects be fully set up during construction). Did you remember to update all of those annotations in that class? Did you remember to change related annotations in other classes that call methods in this one? And so forth.

Share
Improve this answer
Follow

edited Nov 20, 2009 at 12:20

answered May 18, 2009 at 13:27

Bob Cross
22.3k ● 12 ● 62 ● 95

1

Agitar has a [commercial] **product**, AgitarOne, that generates JUnit tests. I'm not sure it currently support annotations, it didn't in 2005.

Jtest is another Java-oriented unit testr generator and Parasoft also offers C++Test to generate C++ unit tests.

I never tested any of them ; I've read the C++Test paper a few years ago, but wasn't convinced.

Share Improve this answer Follow

answered Dec 28, 2008 at 19:04

philant
35.7k ● 11 ● 73 ● 113

1

Somewhat similar to what you've described for testing the presence of a default constructor, I've used TestUtil to automatically test getters and setters. You can use either an XML file or JavaDoc tags to tweak the testing options. Sadly there doesn't currently appear to be an option for annotations.

Share Improve this answer Follow

answered Dec 28, 2008 at 21:08

Robert Simmons
473 ● 3 ● 10

1

"process the annotations ... and generate the test for me"

For a limited number of cases, this might be made to work. In general, however, it can't work.

```
@StandardTestForClassHierarchy1
@StandardTestForClassHierarchy2
@StandardTestForClassHierarchy3
@StandardTestForSomeOtherFeature4
@AspectFeature5
@AspectFeature6
@HasPublicDefaultConstructor
@AspectX
@AspectY
class SomeClass extends SomeClassHierarchy implements SomeOtherFeature {
}
```

I can't distinguish my unittest annotations from my real annotations.

Do the testing annotations describe run-time behavior of my application?

- If they *do* describe run-time behavior, then they're for-real AOP annotations, not descriptions of tests, but real annotations that really generate run-time code. And the annotation has it's own test on mock classes.
- If the annotations *don't* describe run-time behavior, I now have this weird clutter of non-functional and functional annotations. I'm not seeing the value in non-functional annotations.

Share

Improve this answer

Follow

edited Dec 29, 2008 at 21:22

answered Dec 29, 2008 at 17:24

S.Lott
391k ● 82 ● 517 ● 788

0

this might be what you are looking for: http://developer.spikesource.com/wiki/index.php/Projects:testgen4j. other maybe useful links at: http://www.opensourcetesting.org/unit_java.php. here is a current paper: http://www.alexquinn.org/papers/Generating%20Java%20unit%20tests%20with%20AI%20planning%20(2007).pdf

Share Improve this answer Follow

answered Dec 28, 2008 at 16:25

Ray Tayek

10k ● 9 ● 54 ● 96

0

EiffelStudio CDD has an interesting approach. While running in debug mode the IDE collects information about the methods being called and puts the context and the calls into unit tests. Failures are detected using Design by Contract. I know there are some design by contract extensions on top of Java, so this might be a nice way to go.

Share Improve this answer Follow

answered Dec 28, 2008 at 16:41

cdecker

4,667 ● 8 ● 50 ● 81

0

For your specific example it might be better to use java APT (compile time annotation processing) to detect such problems much earlier, at compile time.

Share Improve this answer Follow

answered Dec 28, 2008 at 17:05

maximdim

8,159 ● 3 ● 35 ● 48

0

You can detect this at compile time with APT.

http://www.javaspecialists.eu/archive/Issue167.html

Share Improve this answer Follow

answered Dec 30, 2008 at 17:40

Craig P. Motlin

26.7k ● 18 ● 103 ● 127

0

I think you are unlikely to save yourself anything. But you will add complexity.

If I wanted to be sure there was a default constructor I would add the following to my unit test.

```
Foo foo = new Foo();
```

Share Improve this answer Follow

answered May 18, 2009 at 18:52

Peter Lawrey
533k 82 767 1.1k