

Combining Java Reflection API with Java Annotation Types for Thread Safety

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-10/msg02602.html>

- *From:* pek <kimwlias@xxxxxxxxx>
 - *Date:* Mon, 22 Oct 2007 22:39:10 -0000
-

Hello everyone. I have a really complicated question (and I don't really know if this is even possible).

I am currently developing a program that is multithreaded. I have a lot of methods that start a thread and call other methods from other classes. All the time, I have to make sure that the class's method that I am calling is correctly synchronized (if needed) or not. So here is what I was looking the last couple of days.

I know Java has an Annotation Type system that it can help me mark various methods in a class for a particular purpose (such as a `@Test` or `@FixThis`). So I was thinking, this could also work if every method that inside the code a thread is initiated, I could Annotate it as `@ThreadCreator`. Also, I could annotate a method that is being called from a thread as `@ThreadSafe` (making sure of course that it is indeed a thread-safe method). This is the first part.

The second part is to use Java Reflection API. I saw that I can use this API to find what annotations a class uses and where. So I know I could find out that a particular method in a class is annotated as `@ThreadCreator`.

Now, combining this together, I was wondering if I could create a "system" (I don't really know how to call it) that would take a class that I give it, find which methods are annotated with `@ThreadCreator`, invoke those methods, and see if the methods call other methods that are not annotated as `@ThreadSafe`, in which case the "system" will warn me that this should be `@ThreadSafe`.

Is this even possible? Is there any other solution to this? Is there already an implementation of this?

Thank you very much for your time.

.