

Google Android Tool to check performance requirements

Mattia Gustarini (Mattia.Gustarini@usi.ch)

Software Performance Laboratory project - Matthias Hauswirth (Matthias.Hauswirth@unisi.ch)

Friday, December 11, 2009

Project proposal	2
Topic	2
How much	2
Dependencies	2
Core feature	2
Optional features	2
Deliverables	2
RoadMap	3
Performance guidelines evaluation	3
Schedule	3
How to, Android Performance Tool at work	4
Examples	4
From command line	4
As Eclipse Plugin	4

1. Project proposal

1.1. Topic

In the Google Android developers web site¹ in the *Dev Guide* it is possible to find a section that describes the *Best Practice* to follow in order to write an android application. One of its subsections gives a detailed list on how to design applications for performance². Here you can find a series of guidelines and some of them can be checked automatically by looking the byte code of the .class Java files. Unfortunately Google doesn't provide a tool that can check automatically those guidelines and the developer must remember and apply all the suggestions on its own.

1.2. How much

The aim of this project is to provide the missing tool to the developers of the Android applications and maybe in a second step integrate it on the Google ADT plugin for Eclipse.

The tool will try to cover all the practices that can be checked by looking directly to the Java class code. It will be a stand alone "product" like all the tools that are currently offered by Google to develop Android applications. A developer that doesn't use Eclipse must be able to execute it on the Android application project structure without the help of the plugin.

How to integrate the tool on the Google plugin is not know and this option will be exploited only when at least the automatic checks directly related to the source code are done. The tool will for sure produce an *xml* file that can be used to easily integrate it in the plugin. The integration can be problematic so we must check in advance this possibility, but how? (where is the Google project of this plugin? Is it open source? Is it integrated with the Android platform open source project³).

1.3. Dependencies

1. Google Android application performance guidelines
2. Android application project structure
3. ASM
4. ADT plugin source code (possible)

1.4. Core feature

We expect to deliver a tool able to automatically check the Google Android performance guidelines by looking to the Java .class byte code of an Android application project.

1.5. Optional features

Integrate the tool in the ADT plugin used to develop Android application in Eclipse.

1.6. Deliverables

This updated project report and the software tool.

¹ <http://developer.android.com>

² <http://developer.android.com/guide/practices/design/performance.html>

³ <http://source.android.com/>

2. RoadMap

2.1. Performance guidelines evaluation

guideline	difficulty
Avoid Creating Objects	hard
Use Native Methods	hard
Prefer Virtual Over Interface	easy
Prefer Static Over Virtual	medium
Avoid Internal Getters/Setters	medium
Cache Field Lookups	hard
Declare Constants Final	medium
Use Enhanced For Loop Syntax With Caution	medium
Avoid Enums	easy
Use Package Scope with Inner Classes	medium
Avoid Float	easy

2.2. Schedule

The project has a duration of 6 weeks, for each week we define the guideline we wish to implement and possibly some extra work that we have to do. The decision will be based on the difficulty assigned to each guideline.

Week 1:

1. create the structure to retrieve the bytecode from an Android project
2. [Prefer Virtual Over Interface](#)
3. create the structure to represent the tool's output
4. update this document

Week 2:

1. [Avoid Float](#)
2. [Use Package Scope with Inner Classes](#)
3. [Avoid Internal Getters/Setters](#)
4. update this document

Week 3:

1. [Avoid Internal Getters/Setters](#) (cont.)
2. [Avoid Enums](#)
3. check if possible to generate an output for eclipse (error markers)
4. update this document

Week 4:

1. [Prefer Static Over Virtual](#)
2. [Declare Constants Final](#)
3. update this document

Week 5:

1. [Use Enhanced For Loop Syntax With Caution](#)
2. feasibility study: is it possible to easily create a plugin for Eclipse?
3. update this document

Week 6:

1. Create an Eclipse plugin to use the tool
2. update this document

3. How to, Android Performance Tool at work

3.1. Android test projects

In the project zip you can find two Android projects ready to be used to test the tool. The one called *AndroidToolTestProject* is a project created to check all the analysis implemented in the tool. The other project called *ApiDemos* is a demo project from Google containing a lot of classes that can be checked with the tool.

3.2. From command line

To run the tool from the command line do the following steps:


1. create the *apt.jar* file with the script *create_apt.jar*
2. launch the tool with the script *aptool* with the following arguments:
 - a) *aptool <android project path>*
(to execute the verbose version of the tool)
 - b) *aptool <android project path> <xml file name>*
(to use the quiet option, to output the result of the analysis to an xml file)


For example with the two given project:

1. *AndroidToolTestProject*
 - a) *aptool AndroidToolTestProject*
 - b) *aptool AndroidToolTestProject androidtooltestproject.xml*
2. *ApiDemos* (it will produce a “lot” of output)
 - a) *aptool ApiDemos*
 - b) *aptool ApiDemos apidemos.xml*

3.3. As Eclipse Plugin

To test the plugin simply import the *com.android.ide.eclipse.apt* project in Eclipse (the used Eclipse must support plug-in Development!) and then launch it as an Eclipse application. In the Eclipse version with the plugin that is running import the two Android projects *ApiDemos*, *AndroidToolTestProject*. The two projects are already Eclipse Android projects (they must have the Android project nature), so in order to test the plugin you may not need to install the Android development plugin (if it is not the case go here: [Installing ADT Eclipse plugin](#)).

To analyze a project with the tool select one of the Android projects (only with them it is going to work, otherwise an error message will be showed...) and press this little icon .

Now all the resource where the above guidelines are not respected should contain info-markers with the description of the problems found by the analyzers of the tool. If you want you can correct the problems, but in order to see if your work is successful you need to save the resource/s (in order to build the modified Java file/s, remember the tool works on class files), reselect the project (sorry analysis of a single resource is not yet supported...) and click on the little icon  again. If you did a good job the marker/s should disappear.

As last comment we need to say that markers are transient, so when you close the Eclipse application where the plugin “is running” they are going to disappear. This as well the fact that the user has to press the button to start the analysis are implementation choices to not have a lot of markers on the resources and to allow the user to chose when to see them.