Motivation

Javadoc is a documentation tool used for generating API documentation from Java source code. For some methods the generated documentation can contain description of the algorithm and its asymptotic complexity. However, such information is futile when the exact execution time of the method is needed with respect to certain critical characteristics. Typical example is when we need to know the execution time of an operation performed upon a collection, which may for example be determining the execution time of `contains` method on `ArrayList`.

Objectives

- design and implement an interactive measuring tool integrated into the Javadoc
- design reasonable interface for the workload generators, i.e. for the code which prepares the actual workload for the measured method
- the tool should provide accurate measurement results with respect to the Java platform
- the tool should show a trend of the method on an interval (e.g. how long does it take to sort the collection containing from 100 to 1000 items)

Solution

Application architecture

The application consists of two tools: enhanced standard doclet, which generates Javadoc with performance information, and a measuring server, which is responsible for accurate measuring of a method. Both these tools are independent allowing us to run the measuring server on a dedicated reference machine.

![Measurement architecture](https://example.com/measurement_architecture.png)

Javadoc with performance extension

The enhanced standard doclet generates Javadoc, which also contains performance part for methods having appropriate annotation. The whole method documentation now consists of three sections as can be seen in Figure 2. The first one is the unchanged part of the standard doclet. The second and the third part are then the performance extension, where the second part serves for setting arguments to the particular workload generator (selected by the dropdown menu) of the method and the third part is responsible for showing measured results. The user can select to view the results in the form of a graph or a table.

![Documentation structure](https://example.com/documentation_structure.png)

Measuring server

The goal of the measuring server is to measure the execution time of a method as precise as possible. However, the measuring process may take quite a long time; therefore, the measuring server provides approximate results (measured with reflection) in a short time, while the more time demanding and accurate results (measured using direct call on generated sources) are sent after the longer time. The quality of measured results is then in the Javadoc denoted by the type of the line.

The measuring server caches (into database) all results in order not to perform same measurements multiple times. The cache is then available in HTML format allowing user to inspect all measurements. The quality of results may be configured by, for example, setting the time intended for measurement of one point.

Workload generators

We want to measure the method performance; therefore we need to generate the parameters, with which the measured method will be called as well as the instance, in which the measured method will be called. We decided to separate measured method and code to prepare the parameters with the instance, which we call workload generator. This separation allows developers to implement both parts clearly and easily, and also share the generator amongst multiple methods.

The tool utilizes annotations to mark measured method as well as its workload generator. Annotations are also used to provide additional parameter information such as its description or minimal value. The user controls, displayed in the Javadoc allowing user to select values against which the measurement will be performed, are then generated from these parameters.

Conclusion

The implemented tool allows developers to bind performance information of a method with the Javadoc. The tool is interactive and provides the approximate results as fast as possible. The application was designed with respect for possible extensions and provides for them reasonable interface.

Additional information

Author: Jakub Náplava [arahusky@seznam.cz]
Supervisor: Mgr. Vojtěch Marký
https://github.com/arahusky/performance_javadoc