

## **Wikiprint Book**

**Title: JPF Inspector**

**Subject: Java Path Finder - projects/jpf-inspector**

**Version: 11**

**Date: 02/21/2013 11:54:50 AM**

## Table of Contents

<b>JPF Inspector</b>	<b>3</b>
Contact	3
Description	3
User guide	3
Repository	3

## JPF Inspector

JPF Inspector is a tool for inspection and control of JPF execution. It supports breakpoints and "single step" execution (forward and backward) at different levels of granularity, and it allows the user to examine and modify program state (threads, call stacks, and heap objects). Unlike with standard debuggers (GDB), it is also possible to control thread scheduling explicitly.

Development of this tool started in the scope of the [Google Summer of Code \(GSoC\)](#) program in 2010 and continued in 2011. However, many changes to the code base are now being made outside of GSoC.

### Contact

Pavel Jancik <pavel.jancik (at) d3s.mff.cuni.cz>, Pavel Parizek <parizek (at) d3s.mff.cuni.cz>

### Description

Current features of the JPF Inspector tool include:

- Breakpoints. They can be set to: a specific line of code, instruction at a given position, specific instruction type (invoke, return), read or write access of a variable, object creation and destruction, garbage collection (start and stop), transition boundary (next value choice). Each breakpoint can be in any of the following three states: enabled, disabled, and log. If a breakpoint in the log state is reached, a log message is printed and the execution is not interrupted.
- Single-step execution. It is possible to tell JPF to execute the SuT in a single-step fashion. Both forward and backward direction is supported with the following levels of granularity: instruction and source code line (w/o stopping at nested method calls). Inspector also supports a running mode in which the next choice at each state is selected by the user.
- Program state inspection. Inspector provides a mechanism for exploring program state, including values of heap objects and call stacks of individual threads. Navigation over references (object tree) is supported too - a custom expression language is used internally.
- Program state modification. The user can modify program state to a limited degree, and then check how the modifications influence the program behavior.
- Record and replay. Inspector allows to save all commands executed in the current session into a file and replay them later.
- Dynamic assertions. It is possible to add simple assertions (conditional breakpoints associated with source code locations) dynamically.

Planned features include some of the following: JPF introspection (displaying raw state of JPF data structures, choice generators and listeners), pure command-line interface, and better support for automated debugging of concurrent programs.

JPF Inspector currently provides GUI that is implemented as a new panel for the [JPF shell](#) framework.

### User guide

[User guide](#) describes the currently supported commands and extension points (API) of the Inspector tool. It includes many examples and some screenshots of the Inspector user interface.

### Repository

The Mercurial repository with all the sources is on <http://babelfish.arc.nasa.gov/hg/jpf/jpf-inspector>