

Wikiprint Book

Title: JPF Projects

Subject: Java Path Finder - projects/start

Version: 35

Date: 03/17/2013 07:00:59 PM

Table of Contents

TracNav	3
Introduction...	3
Installing JPF...	3
User Guide...	3
Developer Guide...	3
Projects	3
About...	3
JPF Projects	4
JPF Core Project:	4
Model Checking Projects:	4
Symbolic Execution Projects:	4
JPF Tools Projects:	4

TracNav

- [JPFWiki](#) - Welcome Page

[Introduction...](#)

[Installing JPF...](#)

[User Guide...](#)

[Developer Guide...](#)

Projects

- [jpf-core](#)
- [jpf-actor](#)
- [jpf-awt](#)
- [jpf-awt-shell](#)
- [jpf-concurrent](#)
- [jpf-cv](#)
- [jpf-delayed](#)
- [jpf-guided-test](#)
- [jpf-mango](#)
- [jpf-racefinder](#)
- [jpf-rtembed](#)
- [jpf-statechart](#)
- [net-iocache](#)
- [jpf-aprop](#)
- [jpf-numeric](#)
- [jpf-symbc](#)
- [jpf-concolic](#)
- [jpf-symbc-load?](#)
- [jpf-extended-test-gen](#)
- [jpf-parallel-spf?](#)
- [eclipse-jpf](#)
- [netbeans-jpf](#)
- [jpf-inspector](#)
- [jpf-shell](#)
- [jpf-template](#)
- [jpf-trace-server](#)
- [standard NB example](#)
- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

About...

- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

JPF Projects

JPF is divided into separate [runtime modules](#), each one being maintained as its own project. The *jpf-core* component is always required, but all other ones are optional. If you are familiar with JPF, those are the *extensions*, implementing different execution modes, special properties or alternative library models.

Each component project has its own repository, and project pages can include binary snapshots as attached files.

Here is the list of components that are available from this server, each with more details about purpose, installation and use. If you don't find an extension that was in the JPF v4 distribution (SourceForge), don't worry - we are in the process of porting them.

JPF Core Project:

- [jpf-core](#) - the VM and core mechanisms used by all other projects - explicit state model checking for Java bytecode

Model Checking Projects:

- [jpf-actor](#) - Tool and framework for systematic testing of actor programs (e.g. Scala)
- [jpf-aprop](#) - Java annotation based properties and their corresponding checkers
- [jpf-awt](#) - JPF specific library implementations for `java.awt` and `javax.swing`
- [jpf-awt-shell](#) - specialized JPF shell for model checking `java.awt` and `javax.swing` applications
- [jpf-concurrent](#) - optimized `java.util.concurrent` library implementation for JPF
- [jpf-delayed](#) - postpones non-deterministic choice of values until they are used
- [jpf-guided-test](#) - Framework for guiding the search using heuristics and static analysis
- [jpf-mango](#) - specification and proof artifact generation
- [jpf-numeric](#) - an alternative bytecode set for inspection of numeric programs
- [jpf-racefinder](#) - a precise data race detector in a relaxed Java memory model
- [jpf-rtembed](#) - programs for real-time and embedded platforms (e.g., RTSJ and SCJ)
- [jpf-statechart](#) - UML statechart modeling
- [net-iocache](#) - I/O cache extension to handle network communication

Symbolic Execution Projects:

- [jpf-symbc](#) - Symbolic PathFinder - symbolic execution for Java bytecode
- [jpf-concolic](#) - concolic (mixed concrete/symbolic) execution for Java bytecode
- [jpf-symbc-load?](#) - Symbolic Load Test Generation
- [jpf-extended-test-gen](#) - Using JPF and SPF for generating tests with respect to MC/DC coverage
- [jpf-parallel-spf?](#) - Generic framework for parallelizing SPF (needs `jpf-extended-test-gen`, `jpf-symbc`, `jpf-core`)

JPF Tools Projects:

- [eclipse-jpf](#) - JPF launcher plugin for Eclipse IDE
- [netbeans-jpf](#) - JPF launcher plugin for NetBeans IDE
- [jpf-inspector](#) - a tool for inspection and control of JPF execution
- [jpf-shell](#) - the generic user interface support for JPF
- [jpf-template](#) - a tool for creating new JPF projects
- [jpf-trace-server](#) - enables storing, querying and analysis of the execution trace
- [standard NB example](#) - a NetBeans project example