

Wikiprint Book

Title: JET - JPF Eclipse Tools

Subject: Java Path Finder - summer-projects/2010-mji-editor

Version: 11

Date: 02/23/2013 12:29:48 PM

Table of Contents

TracNav	3
Introduction	3
Installing JPF	3
User Guide	3
Developer Guide	3
MJJ	3
Projects	3
About	4
JET - JPF Eclipse Tools	4
Abstract	4
Contact	5
Description	5
(1) JPF Project Creation	5
(2) JPF Configuration Editing	5
(2a) JPF site properties	5
(2b) JPF project properties	5
(2c) JPF app properties	5
(3) NativePeer Creation/Navigation	5
Program & Timeline	6
Repository	6
Project Blog	6

[TracNav](#)

- [JPFWiki](#) - Welcome Page

[Introduction](#)

- [What is JPF](#)
- [Testing vs model checking](#)
- [Random Example](#)
- [Race Example](#)
- [JPF classification](#)

[Installing JPF](#)

- [System requirements](#)
- [Download snapshots](#)
- [Download repositories](#)
- [Create site.properties](#)
- [Install NetBeans IDE plugin](#)
- [Install Eclipse IDE plugin](#)
- [Building and testing](#)

[User Guide](#)

- [Application Types](#)
- [JPF Components](#)
- [Configuring JPF](#)
- [Running JPF](#)
- [JPF Output](#)
- [The JPF API](#)

[Developer Guide](#)

- [Design](#)
- [Choice Generator](#)
- [Partial Order Reduction](#)
- [Attributes](#)
- [Listener](#)

[MJJ](#)

- [Mangling for MJJ](#)
- [Bytecode Factory](#)
- [Logging](#)
- [Report](#)
- [Embedded](#)
- [JPF tests](#)
- [JPF project layout](#)
- [Create a JPF project](#)
- [Coding Conventions](#)
- [Hosting update site](#)

[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](#)

- [About this Wiki](#)
- [About the Mailing Lists](#)
- [About the Development Process?](#)
- [About the Repository?](#)
- [How to Contribute](#)
- [JPF contributor account](#)
- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

JET - JPF Eclipse Tools

Abstract

This project aims at providing support for JPF specific programming within the [Eclipse IDE](#). It differs from the [jpf-shell?](#) project by focusing on editing related tasks for JPF specific files such as [NativePeers](#), whereas `jpf-shell` is meant to support running JPF and displaying its various outputs outside of an IDE.

Programming tasks supported by JET include especially creation of JPF projects, editing of JPF properties files, and NativePeer creation and navigation.

Contact

student: Matt Kirn <mattkse "at" gmail.com> (MK)

mentor: Peter Mehlitz <pcmehlitz "at" gmail.com> <PM>

co-mentor: Darko Marinov <marinov "at" illinois.edu> (DM)

Description

Details about ongoing work can be found on: <http://bitbucket.org/mattkse/jet/wiki>.

The main programming tasks targeted by JET are

(1) JPF Project Creation

This will provide a wizard dialog to create a [JPF project](#) with the required directory structure, configuration files (e.g. jpf.properties) and build scripts (build.xml). The actual creation is performed by the existing `gov.nasa.jpf.tool.CreateProject` class, the wizard mostly provides the required arguments such as project name and location.

(2) JPF Configuration Editing

JET supports editing `site.properties`, project properties (`jpf.properties`) and application properties (`*.jpf`) according to the [JPF configuration model](#).

(2a) JPF site properties

- Properties are configurable via the Eclipse preferences manager
- A default setup is provided upon fresh install of plug-in. Error messages are displayed if the default setup is invalid on the user's system
- Automated editing support consists of creating `jpf-core`, project name and respective `extensions` entries for selected `jpf-core` and project directories

(2b) JPF project properties

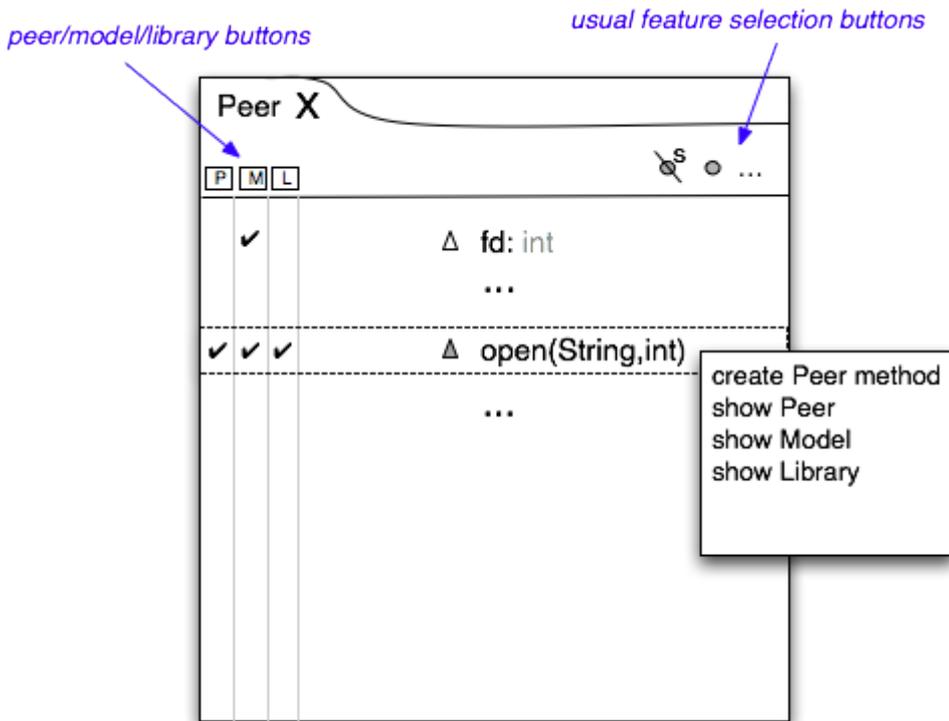
- Each project can configure `jpf.properties` via a new category in the project properties manager
- When creating a new JPF project, there is an option to launch the properties manager to configure the properties
- All properties that can be overridden are displayed in the properties manager
- User can finish without modifying properties and accept a default configuration
- Automated editing support consists of selecting project specific `classpath`, `native_classpath` and `sourcepath` directories, providing default values according to the standard JPF project layout

(2c) JPF app properties

- User can create app property files by selecting a context menu item on a target class (signified by its location in `src/examples` or `src/tests`), which subsequently opens a GUI dialog for user input
- All properties that can be overridden are displayed in the GUI dialog
- User can finish without overriding any properties to accept a default configuration
- Automated editing support consists of adding `target` and `target_args` entries according to the selected target class, and by adding the standard "`<project-name> = ${config_path}`" preamble

(3) NativePeer Creation/Navigation

This will provide support to create [NativePeer classes](#), and to navigate between the native peer, the (optional) respective model class, and a respective standard library version of the model class (if any). These functions will be accessible through a new "Peer View" that is a specialization of the standard Eclipse "Outline View", tying together the aforementioned NativePeer, model and library class according to the following drawing.



Creating a NativePeer class involves selecting the target (model) class, and then selecting the methods within this class for which NativePeer method stubs should be created. Based on these selections, the Eclipse plugin has to provide the following functions:

- creating NativePeer sources with correct names (JPF_<model-class-package>_<model-class-name>) within the `src/peers` directory
- computing correctly mangled NativePeer method names that include signature (model class argument types) and return type, according to the scheme described in [devel/mji MJJ?](#)
- setting required method modifiers (`public static`)
- creating proper argument lists (which always start with a `(MJIEnv env, int objref,...)` prefix)
- proper mapping of model class reference arguments or return types into `int ref` types (JPF internally uses `int` to represent reference values)
- creating method stub bodies (with proper return values)

This should be based on bytecode model classes since there might not be any model sources (e.g. for standard library classes). To avoid classpath setup problems, it seems best to use the BCEL library to parse the selected model class files (*.class). Using BCEL instead of Java reflection deviates from the approach taken with the `GenPeer` tool that was part of JPF v4, but does not require providing complete SUT classpaths that are otherwise just relevant when running JPF (i.e. are usually specified in *.jpf application property files).

Navigation functions should allow jumping between corresponding methods in the associated NativePeer, the model and the library class, to be displayed in different editors. Library classes should be displayed in read-only editors

Program & Timeline

This project is funded by a NASA Ames Internship. It follows the GSoC timeline (start 05/24, finish 08/16)

Repository

The sources for this project are available from a Mercurial repository on <http://bitbucket.org/mattkse/jet>

Project Blog

(most recent on top)

2010-06-09 (PM) - updated project description according to discussions

2010-05-21 (PM) - added some project description

2010-05-07 (NR) - project page created