

## TracNav

- [JPFWiki - Welcome Page](#)

### Introduction...

#### Installing JPF

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

#### User Guide...

#### Developer Guide...

#### Projects...

- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

#### About...

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

## **Building and Running**

If you downloaded binary snapshots, you don't have anything to build (except of creating/updating your [site.properties](#) file).

### **Building JPF**

If you have cloned the project repositories you are interested in (which at least includes jpf-core), you can build and test each of them by means of their included [Ant](#) build.xml scripts.

#### **Using the command line**

```
> cd jpf-core
> bin/ant test

... lots of output, at the end you should see something like:
BUILD SUCCESSFUL
Total time: 2 minutes 31 seconds
```

#### **Within NetBeans**

1. run "File/Open Project.." from the application menu, entering the JPF project you just downloaded (e.g. "jpf-core")
2. select the project that appears in our project pane (e.g. "jpf-core")
3. run "Build" from the project context menu

## Within Eclipse

- Ensure that the JAVA\_HOME environment variable points to the jdk1.6xxx directory. If it is empty or points to a JRE then errors such as **javac not found** maybe seen. If you do not want the system Java settings to point to jdk1.6xxx, you can also set project specific settings in eclipse.
- If you eclipse settings are set to **Build Automatically** then the project after being cloned will be built.
- To build a particular project in the Project menu select **Build Project**. All the dependencies for the project will be built automatically.

## Project Specific JDK settings within Eclipse

1. In Eclipse go to **Project -> Properties**
1. Select **Builders**
1. Pick **Ant\_Builder** -> click **Edit**
1. Click on the **JRE** tab
1. Select **Separate JRE** -> **Installed JREs**
1. On Windows and Unix-based systems pick **JDK1.6xxx**. If it is not listed under the installed JREs, click on **Add**, browse your file system to where **JDK1.6xxx** resides and select. On OSx systems pick the **JVM 1.6.0**.

## Running JPF

### Using the command line

```
> cd jpf-core
> java -jar build/RunJPF.jar src/examples/Racer.jpf
JavaPathfinder v5.0 - (C) 1999-2007 RIACS/NASA Ames Research Center
.....
=====
statistics
elapsed time: 0:00:00
states: new=9, visited=1, backtracked=4, end=2
search: maxDepth=5, constraints=0
choice generators: thread=8, data=0
heap: gc=8, new=291, free=32
instructions: 3112
max memory: 79MB
loaded code: classes=73, methods=1010

===== search finished: 1/12/10 2:30 PM
```

### Using eclipse plugin

1. Right click on a .jpf file. Examples can be found in the src\examples directory in jpf-core
2. If the eclipse plugin is correctly installed, a Verify option will appear.
3. Select the Verify option and the verification process of the system specified in the .jpf file begins

Note that the Application.jpf file essentially replaces previous uses of eclipse launch configurations. The required element of a .jpf file is the **target=MAIN\_CLASS** where **MAIN\_CLASS** is the class containing main method of the system under test. Any other configurations that need to be specified can be added here. for example 'listener=gov.nasa.jpf.tools.SearchMonitor'.

Specify **classpath=PATH\_TO\_BIN\_DIRECTORY** to add the class files for the *program under test* to JPF's class path. Windows users will need to use the double-backslash notation in specifying paths in the .jpf file. An example .jpf file for the Windows platform is included below for convenience:

```
target=mutex.DekkerTestMain
report.console.property_violation=error,trace,snapshot
listener=gov.nasa.jpf.listener.EndlessLoopDetector
classpath=C:\\Users\\fred\\Documents\\ch02-mutex\\bin
```

```
sourcepath=C:\\\\Users\\\\fred\\\\Documents\\\\ch02-mutex\\\\src,C:\\\\Users\\\\Fred\\\\Documents\\\\ch02-mutex\\\\src-test
```

The .jpf file not only indicates the **target** and **classpath**, but it also turns on error reporting with trace generation (**report.console.propertyViolation**) and configures the source path (**sourcepath**). Note that multiple source directories are specified using the comma separator.

## Using eclipse Run Configuration

1. Select a .jpf file by clicking on it in the Package Explorer
2. Click Run -> Run Configurations -> **run-JPF-core**. It is important the correct .jpf file is selected when the configuration is run.

### Windows users

After a fresh install of jpf-core you may see the following when trying to use the Eclipse Run-jpf-core configuration:

```
java.lang.NoClassDefFoundError: gov/nasa/jpf/Main
Caused by: java.lang.ClassNotFoundException: gov.nasa.jpf.Main
    at java.net.URLClassLoader$1.run(Unknown Source)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(Unknown Source)
    at java.lang.ClassLoader.loadClass(Unknown Source)
    at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
    at java.lang.ClassLoader.loadClass(Unknown Source)
Exception in thread "main"
```

In this particular case, the error was generated after the initial build after the clone had completed. To resolve the issue, **refresh the Eclipse workspace** (F5 or right click Refresh). After the refresh, the Run-jpf-core configuration should work as described above.