A Hands-On Java PathFinder Tutorial
ICSE 2013
Part 2: Install and Run JPF

Neha Rungta
SGT / NASA Ames Research Center
<neha.s.rungta@nasa.gov>

Peter C. Mehlitz
SGT / NASA Ames Research Center
<peter.c.mehlitz@nasa.gov>

Willem Visser
University of Stellenbosch
<wvisser@cs.sun.ac.za>
Install JPF

✦ four installation steps:

1. get sources
2. build
3. test
4. configure

✦ all JPF projects (should) follow the same procedure

✦ start with jpf-core
Install: Step 0 - Prerequisites

✦ latest stable OpenJDK release from
  http://www.oracle.com/technetwork/java/javase/downloads

✦ make sure JDK is installed, not JRE (check for javac in the path)

✦ Mercurial (Version Control System, uses Python):
  http://mercurial.selenic.com

✦ optional IDEs:
  • Eclipse: http://www.eclipse.org
    ‣ (also requires MercurialEclipse plugin from
      http://javaforge.com/project/HGE)
  • NetBeans: http://www.netbeans.org
Install: Step 1 - Getting JPF source

- get jpf-core sources from Mercurial repository
  ```bash
  > hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-core
  ```

- alternatively get and unzip *.zip snapshot attachment from
  [http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-core](http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-core)

  ~ or ~

  memory stick passed around

- same procedure for JPF extension projects
Install: Step 2 - Building JPF

✦ jpf-core comes with all the required build tools (except javac)
✦ build from within jpf-core directory

> bin/ant build

Buildfile: .../jpf-core/build.xml
...
-init:
  [mkdir] Created dir: .../jpf-core/build
...
build:
  [jar] Building jar: .../jpf-core/build/jpf.jar
  [jar] Building jar: .../jpf-core/build/jpf-classes.jar
  [jar] Building jar: .../jpf-core/build/jpf-annotations.jar
  [jar] Building jar: .../jpf-core/build/RunJPF.jar
  [jar] Building jar: .../jpf-core/build/RunTest.jar
  [jar] Building jar: .../jpf-core/build/RunAnt.jar

BUILD SUCCESSFUL

✦ same procedure for JPF extension projects
Install: Step 3 - Testing JPF build

✦ JPF comes with regression test suite
✦ test from within downloaded jpf-core directory

> bin/ant test

Buildfile: .../jpf-core/build.xml
...

test:
  [junit] Running TypeNameTest
  [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.385 sec
  ...
  [junit] Running gov.nasa.jpf.util.script.ScriptEnvironmentTest
  [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.028 sec

BUILD SUCCESSFUL
Total time: 1 minute 44 seconds

✦ same procedure for JPF extension projects (if they have tests)
Install: Step 4 - Creating site.properties

- run from within jpf-core directory
  > bin/jpf -addproject
  (this will create a ${user.home}/.jpf/site.properties file)

- .. or just edit ~/.jpf/site.properties manually:

```
jpf.home = ${user.home}/projects/jpf

jpf-core = ${jpf.home}/jpf-core

jpf-numeric = ${jpf.home}/jpf-numeric
...

extensions=${jpf-core},...
```
Run JPF

.joda

for purists (tedious, do only if you have to)

- setting up classpaths
  > export CLASSPATH=...jpf-core/build/jpf.jar...
- invoking JVM
  > java gov.nasa.jpf.JPF +classpath=... x.y.MySUT

using site config and starter jars (much easier and portable)

- explicitly
  > java -jar tools/RunJPF.jar .../x/y/MySUT.jpf
- using scripts
  > bin/jpf .../x/y/MySUT.jpf

running JPF from within JUnit (use JPF test infrastructure, script)

running JPF from your program (tools using JPF)

using NetBeans or Eclipse plugins

- “Verify..” context menu item for selected *.jpf application property file
- using provided launch configs (Eclipse) or run targets (NetBeans)
Run JPF: from Eclipse

✦ use project provided launch configuration (requires `eclipse/run-JPF.launch` in workspace, generic one comes with jpf-core)
  • select *.jpf file in projects view
  • invoke Run As→Run Configurations→run-JPF from context menu
  • results in Output view

✦ use Eclipse JPF plugin from [http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/eclipse-jpf](http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/eclipse-jpf)
  • install from update site if you don’t want to rebuild [http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/install/eclipse-plugin/update/](http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/install/eclipse-plugin/update/)
  • optionally install jpf-shell extension if you want JPF to run in own window
  • launch JPF by selecting *.jpf file and invoking “Verify..” context menu item
Run JPF: Running from NetBeans

✨ use project provided run/debug tasks (requires nbproject/ide-file-targets.xml in project)
  - select *.jpf file in projects view
  - invoke Run → Run File from menubar (not in context menu)
  - results in Output view

✨ use NetBeans JPF plugin
  from http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/netbeans-jpf
  - download & install attached *.nbm if you don’t want to build
  - optionally install jpf-shell extension if you want JPF to run in own window
  - launch JPF by selecting *.jpf file and invoking “Verify..“ context menu item
Run JPF: Random Data Example

- **source:** `src/examples/Rand.java`

```java
public static void main(String[] args) {
    Random random = new Random(42);
    int a = random.nextInt(2);
    int b = random.nextInt(3);
    int c = a / (b + a - 2);
}
```

- **certain combination of random values can cause division by zero**

- **JPF config file:** `src/examples/Rand.jpf`

```plaintext
target = Rand
cg.enumerate_random = true
```

- **run JPF:** `>bin/jpf src/examples/Rand.jpf`

```
java.lang.ArithmeticException: division by zero
    at Rand.main(Rand.java:16)
```

- **JPF error:**

```plaintext
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.ArithmeticException: division by zero
    at Rand.main(Rand.java:16)
```

Monday, May 20, 13
Run JPF: Data Race Example

✦ source: src/examples/Racer.java

```java
int d = 42;

public void run () {
    doSomething(1001);                   // (1)
    d = 0;                               // (2)
}

public static void main (String[] args){
    Racer racer = new Racer();
    Thread t = new Thread(racer);
    t.start();

    doSomething(1000);                   // (3)
    int c = 420 / racer.d;               // (4)
    System.out.println(c);
}
```

✦ JPF application property file: src/examples/Racer.jpf

```properties
target = Racer
listener=gov.nasa.jpf.listener.PreciseRaceDetector
```
Run JPF: Data Race Example (cont.)

run JPF: `>bin/jpf src/examples/Racer.jpf`

================================= system under test
application: Racer.java ...
================================= error #1
gov.nasa.jpf.listener.PreciseRaceDetector
race for field `Racer@13d.d`
  main at Racer.main(Racer.java:16)
    "int c = 420 / racer.d;"
  Thread-0 at Racer.run(Racer.java:7)
    "d = 0;"

================================= trace #1 ...

------------------------------ transition #3 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sleep",2/2,isCascaded:false} ...
  Racer.java:7 : d = 0; // (2) ...
------------------------------ transition #5 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet{id="sharedField",..,{>main,Thread-0}}
  Racer.java:16 : int c = 420 / racer.d; // (4) ...

what if we don’t run with PreciseRaceDetector listener?
(hint: `>bin/jpf +report.console.property_violation=error,trace Racer`)
Run JPF: What does JPF do?

✦ to see what is really going on, run with additional listener:

```
>bin/jpf +listener=.listener.ExecTracker src/examples/Racer.jpf
```

```java
# choice: ThreadChoiceFromSet[id="root",isCascaded:false,>{>main}]
Racer.java:11 : Racer racer = new Racer();
0 : [0] new Racer@317
0 : [1] dup
0 : [2] invokespecial Racer.<init>()V
Racer.java:1 : public class Racer implements Runnable {
0 : [0] aload_0
...
```

```java
----------------------------------- [1] forward: 0 new
----------------------------------- [1] forward: 0 new
# choice: ThreadChoiceFromSet[id="start",isCascaded:false,>{>main,Thread-0}]
0 : [0] executenative JPF_java_lang_Thread.start____V
0 : [1] nativereturn java.lang.Thread.start()V
Racer.java:15 : doSomething(1000); // (3)
...
```

```java
----------------------------------- [4] forward: 3 new end
----------------------------------- [3] backtrack: 2
----------------------------------- [3] done: 2
----------------------------------- [2] backtrack: 1
# choice: ThreadChoiceFromSet[id="sleep",isCascaded:false,>{main,>Thread-0}]
Racer.java:6 : doSomething(1001); // (1)
1 : [−1] runstart
1 : [0] sipush
1 : [1] invokestatic Racer.doSomething(I)V
...```