

Wikiprint Book

Title: IdleFilter

Subject: Java Path Finder - projects/jpf-core/IdleFilter

Version: 1

Date: 02/21/2013 11:08:55 AM

Table of Contents

TracNav	3
Introduction	3
Installing JPF	3
User Guide	3
Developer Guide	3
MJJ	3
Projects	3
About	4
IdleFilter	4
Properties	5
Examples	5

[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](#)

IdleFilter

The `gov.nasa.jpf.listener.IdleFilter` is a listener that can be used to close state spaces with loops. Consider a simple "busy waiting" loop

```
for (long l=0; l<10000000; l++);
```

While not a good thing to do in general, it is benign if executed in a normal VM. For JPF, it causes trouble because it adds a lot of useless steps to the stored path, and slows down execution considerably.

In addition, people who expect JPF to match states can get surprised by programs like

```
while (true){
  // no transition break in here
}
```

not being state matched, and hence not terminating (it wouldn't terminate in a normal VM either).

IdleFilter is a little tool to deal with such (bad) loops. It counts the number of backjumps it encounters within the same thread and stackframe, and if this number exceeds a configured threshold it takes one of the following actions:

- *warn* - just prints out a warning that we have a suspicious loop
- *break* - breaks the transition at the backjump `goto` instruction, to allow state matching
- *prune* - sets the transition ignored, i.e. prunes the search tree
- *jump* - skips the backjump. This is the most dangerous action since you better make sure the loop does not contain side-effects your program depends on.

Properties

Consequently, there are two options:

- **idle.max_backjumps** = <number> : max number of backjumps that triggers the configured action (default 500)
- **idle.action** = warn|break|prune|jump : action to take when number of backjumps exceeds threshold

Examples

(1) The test program

```
...
public void testBreak () {
  int y = 4;
  int x = 0;

  while (x != y) { // JPF should state match on the backjump
    x = x + 1;
    if (x > 3) {
      x = 0;
    }
  }
}
```

would never terminate under JPF or a host VM. Running it with

```
> bin/jpf +listener=.listener.IdleFilter +idle.action=break ...
```

does terminate due to state matching and produces the following report

```
...
===== search started: 4/8/10 4:14 PM
[WARNING] IdleFilter breaks transition on suspicious loop in thread: main
         at gov.nasa.jpf.test.mc.basic.IdleLoopTest.testBreak(gov/nasa/jpf/test/mc/basic/IdleLoopTest.java:42)
...
===== results
no errors detected
```

(2) The following program would execute a long time under JPF

```
...
public void testJump () {
```

```
for (int i=0; i<1000000; i++){
    //...
}

System.out.println("Ok, jumped past loop");
}
```

If we run it with

```
> bin/jpf +listener=.listener.IdleFilter +idle.action=jump ...
```

JPF comes back quickly with the result

```
===== search started: 4/8/10 4:20 PM
[WARNING] IdleFilter jumped past loop in: main
         at gov.nasa.jpf.test.mc.basic.IdleLoopTest.testJump(gov/nasa/jpf/test/mc/basic/IdleLoopTest.java:74)
Ok, jumped past loop
...
```