JBoss Profiler 2.0 Developer's Guide

Table of Contents

1

About JBoss Profiler 2

JBoss Profiler 2 is a 100% pure Java 5 profiler capable of profiling Java Standard Edition and Java Enterprise Edition applications.

In development are currently a web front-end using JSF/RichFaces and a JVMTI native backend.

1.1. The team

Jesper Pedersen acts as the lead for the JBoss Profiler 2 project. He can be reached at jesper (dot) pedersen (at) jboss (dot) org.

1.1.1. Contributors

Huijuan Shao is a major contributor for the JSF/RichFaces web front-end. He can be reached at hjshao (at) hitachi (dot) cn.

2

Introduction

JBoss Profiler 2 is a 100% pure Java profiler capable of profiling Java Enterprise 5 applications.

- 100% Pure Java
- Integration with JBoss Enterprise Middleware Suite (JEMS)
- Access through multiple protocols using JBoss Remoting
    - Socket
    - Remote Method Invocation (RMI)
    - Hyper Text Transport Protocol (HTTP)
- Reports
    - Overview
    - Packages
    - Classes
    - Methods
    - Hotspots
    - Caller
    - Wait time
    - PerThread
    - PerClass
- Able to specify method visibility
    - Global
    - Package
    - Method
- Component identification
    - Plain Old Java Object (POJO)
    - java.lang.Throwable
    - Enterprise JavaBean: Entity
    - Enterprise JavaBean: Session
    - Enterprise JavaBean: Message
    - Servlet
    - Servlet filter
    - JavaServer Pages
    - JMX MBean
    - JavaServer Faces Converter
    - JavaServer Faces Managed Bean
    - RMI: Remote
    - RMI: Server
    - CORBA: Object

    - CORBA: Servant
    - 3rd party plugins
- Compare snapshots
- Client

    - Apache Ant integration
    - Command line

JBoss Profiler 2 was designed to run on the JBoss Application Server 4.2 and JBoss Application Server 5.0 releases or any other Java applications using Java Runtime Environment 5 or higher.

3

Building

3.1. Prerequisites

3.1.1. Java Development Kit (JDK)

You must have one of the following JDKs installed in order to build the project:

- Sun JDK 1.5.x
- Sun JDK 1.6.x

Remember to ensure that "javac" and "java" are in your path (or symlinked).

```
JAVA_HOME=/location/to/javahome
export JAVA_HOME

PATH=$JAVA_HOME/bin:$PATH
export PATH
```

3.1.2. Apache Ant

You must have Apache Ant 1.7.1+ installed on your system.

Remember to ensure that "ant" are in your path (or symlinked).

```
ANT_HOME=/location/to/anthome
export ANT_HOME

PATH=$ANT_HOME/bin:$PATH
export PATH
```

3.1.3. Subversion

You must have Subversion 1.5+ installed on your system.

Remember to ensure that "svn" are in your path (or symlinked).

3.1.4. cmake

You must have cmake 2.6+ installed on your system in order to build the native backend for JBoss Profiler 2. Furthermore you need a C/C++ development environment such as gcc/g++ as well as one of the supported environment that cmake will generate its file too - such as make.

Remember to ensure that "cmake" are in your path (or symlinked).

3.2. Obtaining the source code

3.2.1. Anonymous SVN access

The anonymous SVN repository is located under:

```
svn co http://anonsvn.jboss.org/repos/jbossprofiler/branches/JBossProfiler2 jboss-profiler-2
```

3.2.2. Developer SVN access

The developer SVN repository is located under:

```
svn co https://svn.jboss.org/repos/jbossprofiler/branches/JBossProfiler2 jboss-profiler-2
```

3.3. Compiling the source code

In order to build the JBoss Profiler project you execute:

```
ant <target>
```

where target is one of

- dist

  Builds the distribution.

- release

  Builds the release archives.

- doc

  Builds the documentation for the project.

- web

Builds the web front-end for the project.

- clean

  Cleans the project of temporary files.

See the full list of targets in the main build.xml file.

The native backend is built using:

```
cmake .
make
```

on a Un*x platform. See the cmake documentation for further information.

4

Coding

4.1. Source code formatting

JBoss Tattletale uses 2 spaces for indentation with curly brakets on the same line as the class or the statement.

```
/**
 * My class
 */
public class MyClass {

  /**
   * My method
   */
  public void myMethod() {
    if (value) {
      // Do something
    } else {
      // Do something else
    }
  }

}
```

Remember to add JavaDoc for all classes and methods and remember to run

```
ant checkstyle
```

before any patch submission.

The same rules applies to the web front-end files such as .xhtml and .xml.

See the source code for more examples of the source code formatting rules.

4.2. Component identification plugins

Component identification plugins must implement the following interface

```
org.jboss.profiler.shared.Plugin
```

and contain a default constructor.

Furthermore the plugin must be enabled in both the agent and the client using the "plugins" configuration parameter.

4.3. Agent Jobs

Agent jobs are tasks that the agent can execute at a specific time and interval.

The jobs can be used to profile your application at a certain time where you think there may be a performance problem.

Current job types include:

```
Start profiling: org.jboss.profiler.agent.jobs.StartProfiling
Stop profiling:  org.jboss.profiler.agent.jobs.StopProfiling
```

A job must extend the following class:

```
org.jboss.profiler.agent.jobs.Job
```

and contain a default constructor.

Warning: The jobs override the current settings used in agent.