PMD

Description

Runs a set of static code analysis rules on some Java source code files and generates a list of problems found.

Parameters

Attribute	Description	Required
rulesetfiles	A comma delimited list of ruleset files ('rulesets/basic.xml,rulesets/design.xml'). If you write your own ruleset files, you can put them on the classpath and plug them in here.	Yes, unless the ruleset nested element is used
failonerror	Whether or not to fail the build if any errors occur while processing the files	No
failOnRuleViolation	Whether or not to fail the build if PMD finds any problems	No
minimumPriority	The rule priority threshold; rules with lower priority than they will not be used	No
shortFilenames	Places truncated filenames in the report. This can reduce your report file size by 15%-20%.	No
failuresPropertyName	A property name to plug the number of rule violations into when the task finishes	No
encoding	The character set encoding (e.g. UTF-8) to use when reading the source code files	No
suppressMarker	The series of characters to use to tell PMD to skip lines - the default is NOPMD.	No
maxRuleViolations	Whether or not to fail the build if PMD finds more than the value of this attribute. Note that setting this attribute does not require to set the failOnRuleViolation to true.	No

`formatter` nested element - specifies the format of and the files to which the report is written.

Name	Values
type	xml,ideaj,textcolor,text,textpad,emacs,csv,html,xslt,yahtml,summaryhtml,vbhtml
showSuppressed	Whether to show suppressed warnings; "false" is the default.
toFile	A filename to which to write the report
toConsole	Whether to output the report to the console; "false" is the default.

The `formatter` element can contain nested `param` elements to configure the formatter in detail, e.g.

linkPrefix
Used for linking to online HTMLized source (like this). See example below.
linePrefix
Used for linking to online HTMLized source (like this). See example below.

`classpath` nested element - useful for specifying custom rule. More details on the `classpath` element are in

the Ant documentation here and there's an example below.

`auxclasspath` nested element - extra classpath used for Type Resolution rules.

`sourceLanguage` nested element - specify which language (Java, Ecmascript, XML,...) and the associated version (1.5, 1.6,...)

`ruleset` nested element - another way to specify rulesets. Here's an example:

```
<target name="pmd">
 <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
 <pmd shortFilenames="true">
  <sourceLanguage name="java" version="1.6"/>
  <ruleset>rulesets/java/design.xml</ruleset>
  <ruleset>java-basic</ruleset>
  <formatter type="html" toFile="pmd_report.html">
    <param name="linkPrefix" value="http://pmd.sourceforge.net/xref/"/>
  </formatter>
  <fileset dir="/usr/local/j2sdk1.4.1_01/src/">
   <include name="java/lang/*.java"/>
  </fileset>
 </pmd>
</target>
```

Language version selection

The specific version of a language to be used for parsing is selected via the `sourceLanguage` nested element. Possible values are:

```
<sourceLanguage name="cpp" version=""/>
<sourceLanguage name="fortran" version=""/>
<sourceLanguage name="ecmascript" version="3"/>
<sourceLanguage name="java" version="1.3"/>
<sourceLanguage name="java" version="1.4"/>
<sourceLanguage name="java" version="1.5"/>
<sourceLanguage name="java" version="1.6"/>
<sourceLanguage name="java" version="1.7"/>
<sourceLanguage name="java" version="1.8"/>
<sourceLanguage name="jsp" version=""/>
<sourceLanguage name="php" version=""/>
<sourceLanguage name="ruby" version=""/>
<sourceLanguage name="plsql" version=""/>
<sourceLanguage name="xsl" version=""/>
<sourceLanguage name="xml" version=""/>
<sourceLanguage name="vm" version=""/>
```

Postprocessing the report file with XSLT

Several folks (most recently, Wouter Zelle) have written XSLT scripts which you can use to transform the XML report into nifty HTML. To do this, make sure you use the XML formatter in the PMD task invocation, i.e.:

```
<formatter type="xml" toFile="${tempbuild}/$report_pmd.xml"/>
```

Then, after the end of the PMD task, do this:

```
<xslt in="${tempbuild}/$report_pmd.xml" style="${pmdConfig}/wz-pmd-report.xslt" out="${pmdOutput}/$report_pmd.html" />
```

Examples

nsole as well)

```
<target name="pmd">
    <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
    <pmd rulesetfiles="java-imports">
        <formatter type="html" toFile="pmd_report.html" toConsole="true"/>
        <fileset dir="C:\j2sdk1.4.1_01\src\java\lang\">
            <include name="**/*.java"/>
        </fileset>
    </pmd>
</target>
```

Running multiple rulesets to produce an XML report

```
<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>

<target name="pmd">
    <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
    <pmd rulesetfiles="rulesets/java/imports.xml,java-unusedcode">
        <formatter type="xml" toFile="c:\pmd_report.xml"/>
        <fileset dir="C:\j2sdk1.4.1_01\src\java\lang\">
            <include name="**/*.java"/>
        </fileset>
    </pmd>
</target>
```

Using a custom renderer

```
<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>

<target name="pmd">
    <pmd rulesetfiles="rulesets/java/design.xml">
        <formatter type="com.mycompany.MyRenderer" toFile="foo.html"/>
        <fileset dir="/path/to/java/src">
            <include name="**/*.java"/>
        </fileset>
    </pmd>
</target>
```

Using a classpath reference in the taskdef

```
<path id="pmd.classpath">
    <pathelement location="${build}"/>
    <fileset dir="/path/to/my/pmd/lib/">
        <include name="*.jar"/>
    </fileset>
</path>
```

```
<taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask" classpathref="pmd.classpath"/>

<target name="pmd">
      <pmd rulesetfiles="rulesets/java/design.xml">
          <formatter type="net.sourceforge.pmd.renderers.HTMLRenderer" toFile="foo.html"/>
          <fileset dir="/path/to/java/src">
              <include name="**/*.java"/>
          </fileset>
      </pmd>
  </target>
```

Getting verbose output

```
[tom@hal bin]$ ant -v pmd
Apache Ant version 1.6.2 compiled on July 16 2004
Buildfile: build.xml
Detected Java version: 1.4 in: /usr/local/j2sdk1.4.2_03/jre
Detected OS: Linux
parsing buildfile build.xml with URI = file:/home/tom/data/pmd/pmd/bin/build.xml
Project base dir set to: /home/tom/data/pmd/pmd
Build sequence for target `pmd' is [pmd]
Complete build sequence is [pmd, copy, cppjavacc, cpd, delete,
 compile, clean, jar, dist, cpdjnlp, jjtree, javadoc, test, tomserver]

pmd:
      [pmd] Using the normal ClassLoader
      [pmd] Using these rulesets: rulesets/java/imports.xml
      [pmd] Using rule DontImportJavaLang
      [pmd] Using rule UnusedImports
      [pmd] Using rule ImportFromSamePackage
      [pmd] Using rule DuplicateImports
      [pmd] Processing file /usr/local/java/src/java/lang/ref/Finalizer.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/FinalReference.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/PhantomReference.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/Reference.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/ReferenceQueue.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/SoftReference.java
      [pmd] Processing file /usr/local/java/src/java/lang/ref/WeakReference.java
      [pmd] 0 problems found

BUILD SUCCESSFUL
Total time: 2 seconds
[tom@hal bin]$
```

An HTML report with the "linkPrefix" gizmo

```
<target name="pmd">
 <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
 <pmd rulesetfiles="java-basic" shortFilenames="true">
  <formatter type="html" toFile="pmd_report.html">
    <param name="linkPrefix" value="http://pmd.sourceforge.net/xref/"/>
  </formatter>
  <fileset dir="/usr/local/j2sdk1.4.1_01/src/">
   <include name="java/lang/*.java"/>
  </fileset>
 </pmd>
</target>
```

An HTML report with the "linePrefix" gizmo

```
<target name="pmd">
 <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask"/>
 <pmd rulesetfiles="java-basic" shortFilenames="true">
  <formatter type="html" toFile="pmd_report.html">
    <param name="linePrefix" value=".line"/>
  </formatter>
  <fileset dir="/usr/local/j2sdk1.4.1_01/src/">
   <include name="java/lang/*.java"/>
  </fileset>
 </pmd>
</target>
```

Memory Usage

Memory usage has been reduced significantly starting with the PMD 4.0 release. When testing all Java rules on the jdk 1.6 source code (about 7000 classes), the allocated heap space does not go over 60M.

However, on very large projects, the Ant task may still fail with a OutOfMemoryError. To prevent this from happening, increase the maximum memory usable by ant using the ANT_OPTS variable (adjust the size according to your available memory):

On Windows:

```
set ANT_OPTS=-Xmx1024m -Xms512m
```

On Linux

```
export ANT_OPTS="-Xmx1024m -Xms512m"
```

Copyright © 2002-2014 InfoEther. All Rights Reserved.