OpenGrok - a wicked fast source browser

OpenGrok is a fast and usable source code search and cross reference
engine, written in Java. It helps you search, cross-reference and navigate
your source tree. It can understand various program file formats and
version control histories like SCCS, RCS, CVS, Subversion and Mercurial.

OpenGrok is the tool used for the OpenSolaris Source Browser.

Requirements
---

* Latest Java http://java.sun.com/ (At least 1.5)
* A servlet container like Tomcat (5.x or later)
  http://tomcat.apache.org/
  supporting Servlet 2.4 and JSP 2.0
* Exuberant Ctags http://ctags.sourceforge.net/
* Subversion 1.3.0 or later if SVN support is needed
  http://subversion.tigris.org/
* Mercurial 0.9.3 or later if Mercurial support is needed
  http://www.selenic.com/mercurial/wiki/
* JFlex Ant task (If you want to build OpenGrok)
  http://www.jflex.org/

Usage

SRC_ROOT refers to the directory containing your source tree.
OpenGrok analyzes the source tree and builds a search index along with
cross-referenced hypertext versions of the source files. These generated
data files will be stored in DATA_ROOT directory.

OpenGrok setup Step.0 - Setting up the Sources.

Source base must be available locally for OpenGrok to work efficiently. No
changes are required to your source tree. If the code is under source control
management (SCM) OpenGrok requires the checked out source tree under SRC_ROOT.
It is possible for some SCM systems to use a remote repository (Subversion),
but this is not recommended due to the performance penalty. CVS must have a
local repository.
Note that OpenGrok ignores symbolic links.

Using command line interface.

Step.1 - Populate DATA_ROOT Directory
==========

Option 1. OpenGrok: There is a sample shell script OpenGrok that is suitable
for using in a cronjob to run regularly. Modify the variables in the script
to point appropriate directories, or as the code suggests factor your local
configuration into a seperate file and simplify future upgrades.

Option 2. opengrok.jar: You can also directly use the Java application. If the sources are all located in a directory SRC_ROOT and the data and hypertext files generated by OpenGrok are to be stored in DATA_ROOT, run

```
$ java -jar opengrok.jar -s SRC_ROOT -d DATA_ROOT
```

See opengrok.jar manual below for more details.

Step.2 - Configure and Deploy source.war Webapp
==========

To configure the webapp source.war, look into the parameters defined in web.xml of source.war file and change them (see note1) appropriately.

* HEADER: is the fragment of HTML that will be used to display title or logo of your project
* SRC_ROOT: the absolute path name of the root directory of your source tree
* DATA_ROOT: absolute path of the directory where OpenGrok data files are stored

Optional Step.3 - Path Descriptions

OpenGrok uses path descriptions in various places (For eg. while showing directory listings or search results) Example descriptions are in paths.tsv file. You can list descriptions for directories one per line tab separated format path tab description. Refer to example 4 below.

Note 1 - Changing webapp parameters: web.xml is the deployment descriptor for the web application. It is in a Jar file named source.war, you can change the :

* Option 1: Unzip the file to TOMCAT/webapps/source/ directory and change the source/WEB-INF/web.xml and other static html files like index.html to customize to your project.

* Option 2: Extract the web.xml file from source.war file

```
$ unzip source.war WEB-INF/web.xml
```

edit web.xml and re-package the jar file.

```
$ zip -u source.war WEB-INF/web.xml
```

Then copy the war files to TOMCAT/webapps directory.

* Option 3: Edit the Context container element for the webapp

Copy source.war to TOMCAT/webapps

When invoking OpenGrok to build the index, use -w <webapp> to set the context.

After the index is built, there's a couple different ways to set the
Context for the servlet container:
 - Add the Context inside a Host element in TOMCAT/conf/server.xml

```
<Context path="/<webapp>" docBase="source.war">
   <Parameter name="DATA_ROOT" value="/path/to/data/root" override="false" />
      <Parameter name="SRC_ROOT" value="/path/to/src/root" override="false" />
      <Parameter name="HEADER" value='...' override="false" />
</Context>
```

 - Create a Context file for the webapp

   This file will be named `<webapp>.xml'.

   For Tomcat, the file will be located at:
   `TOMCAT/conf/<engine_name>/<hostname>', where <engine_name>
   is the Engine that is processing requests and <hostname> is a Host
   associated with that Engine. By default, this path is
   'TOMCAT/conf/Catalina/localhost' or 'TOMCAT/conf/Standalone/localhost'.

   This file will contain something like the Context described above.

Using Findbugs

If you want to run Findbugs (http://findbugs.sourceforge.net/) on OpenGrok,
you have to download Findbugs to your machine, and install it where you have
checked out your OpenGrok source code, under the lib/findbugs directory,
like this:

```
cd ~/.ant/lib
wget http://..../findbugs-x.y.z.tar.gz
gtar -xf findbugs-x.y.z.tar.gz
mv findbugs-x.y.z findbugs
```

You can now run ant with the findbugs target:

```
ant findbugs
...
findbugs:
 [findbugs] Executing findbugs from ant task
 [findbugs] Running FindBugs...
 [findbugs] Warnings generated: nnn
 [findbugs] Output saved to findbugs/findbugs.html
```

Now, open findbugs/findbugs.html in a web-browser, and start fixing bugs!

If you want to install findbugs some other place than ~/.ant/lib, you can untar the
.tar.gz file to a directory, and use the findbugs.home property to tell ant where to find
findbugs, like this (if you have installed fundbugs under the lib directory):

```
ant findbugs -Dfindbugs.home=lib/findbug
```

There is also a findbugs-xml ant target that can be used to generate XML files that can later be parsed, e.g. by Hudson.

Using Emma

If you want to check test coverage on OpenGrok, download Emma from http://emma.sourceforge.net/. Place emma.jar and emma-ant.jar in the opengrok/trunk/lib directory, or ~/.ant/lib.

Now you can instrument your classes, and create a jar file:

```
ant emma-instrument
```

If you are using NetBeans, select File - "opengrok" Properties - libraries - Compile tab. Press the "Add JAR/Folder" and select lib/emma.jar and lib/emma_ant.jar

If you are not using netbeans, you have to edit the file nbproject/project.properties, and add "lib/emma.jar" and "lib/emma_ant.jar" to the javac.classpath inside it.

Now you can put the classes into jars and generate distributables:

```
ant dist
```

The classes inside opengrok.jar should now be instrumented. If you use opengrok.jar for your own set of tests, you need emma.jar in the classpath.If you want to specify where to store the run time analysis, use these properties:

```
emma.coverage.out.file=path/coverage.ec
emma.coverage.out.merge=true
```

The coverage.ec file should be placed in the opengrok/trunk/coverage directory for easy analyzation.

If you want to test the coverage of the unit tests, you can run the tests:

```
ant test   (Or Alt+F6 in NetBeans)
```

Now you should get some output saying that Emma is placing runtime coverage data into coverage.ec.

To generate reports, run ant again:

```
ant emma-report
```

Look at coverage/coverage.txt, coverage/coverage.xml and

coverage/coverage.html to see how complete your tests are.

Using Checkstyle

To check that your code follows the standard coding conventions,
you can use checkstyle from http://checkstyle.sourceforge.net/

First you must download checkstyle from http://checkstyle.sourceforge.net/ ,
You need Version 5.0-beta01 (or newer). Extract the package you have
downloaded, and create a symbolic link to it from ~/.ant/lib/checkstyle,
e.g. like this:

```
cd ~/.ant/lib
unzip ~/Desktop/checkstyle-5.0-beta01.zip
ln -s checkstyle-5.0-beta01 checkstyle
```

You also have to create symbolic links to the jar files:

```
cd checkstyle
ln -s checkstyle-5.0-beta01.jar checkstyle.jar
ln -s checkstyle-all-5.0-beta01.jar checkstyle-all.jar
```

To run checkstyle on the source code, just run ant checkstyle:

```
ant checkstyle
```

Output from the command will be stored in the checkstyle directory.

If you want to install checkstyle some other place than ~/.ant/lib, you can
untar the .tar.gz file to a directory, and use the checkstyle.home property
to tell ant where to find checkstyle, like this (if you have installed
checkstyle under the lib directory):

```
ant checkstyle -Dcheckstyle.home=lib/checkstyle
```

Using PMD

To check the quality of the OpenGrok code you can also use PMD
from http://pmd.sourceforge.net/.

How to install:

```
cd ~/.ant/lib
unzip ~/Desktop/pmd-bin-4.2.2.zip
ln -s pmd-4.2.2/ pmd
```

You also have to make links to the jar files:

```
cd ~/.ant/lib/pmd/lib
ln -s pmd-4.2.2.jar pmd.jar
ln -s jaxen-1.1.1.jar jaxen.jar
```

To run PMD on the rource code, just run ant pmd:

```
ant pmd
```

Outout from the command will be stored in the pmd subdirectory.

```
% ls pmd
pmd_report.html  pmd_report.xml
```

If you want to install PMD some other place than ~/.ant/lib, you can
unzip the .zip file to a directory, and use the pmd.home property
to tell ant where to find PMD, like this (if you have installed
PMD under the lib directory):

```
ant pmd -Dpmd.home=lib/pmd-4.2.3
```

AUTHORS

Chandan B.N, Sun Microsystems. https://blogs.sun.com/chandan
Trond Norbye, norbye.org
Knut Pape, eBriefkasten.de
Martin Englund, Sun Microsystems
Knut Anders Hatlen, Sun Microsystems