Chapter 6. Using the FindBugs™ Ant task

Table of Contents

1. Installing the Ant task
2. Modifying build.xml
3. Executing the task
4. Parameters

This chapter describes how to integrate FindBugs into a build script for Ant, which is a popular Java build and deployment tool. Using the FindBugs Ant task, your build script can automatically run FindBugs on your Java code.

The Ant task was generously contributed by Mike Fagan.

1. Installing the Ant task

To install the Ant task, simply copy *$FINDBUGS_HOME*/lib/findbugs-ant.jar into the lib subdirectory of your Ant installation.

Note

It is strongly recommended that you use the Ant task with the version of FindBugs it was included with. We do not guarantee that the Ant task Jar file will work with any version of FindBugs other than the one it was included with.

2. Modifying build.xml

To incorporate FindBugs into build.xml (the build script for Ant), you first need to add a task definition. This should appear as follows:

```
<taskdef name="findbugs" classname="edu.umd.cs.findbugs.anttask.FindBugsTask"/>
```

The task definition specifies that when a findbugs element is seen in build.xml, it should use the indicated class to execute the task.

After you have added the task definition, you can define a target which uses the findbugs task. Here is an example which could be added to the build.xml for the Apache BCEL library.

```
<property name="findbugs.home" value="/export/home/daveho/work/findbugs" />

<target name="findbugs" depends="jar">
  <findbugs home="${findbugs.home}"
            output="xml"
            outputFile="bcel-fb.xml" >
    <auxClasspath path="${basedir}/lib/Regex.jar" />
    <sourcePath path="${basedir}/src/java" />
```

```
      <class location="${basedir}/bin/bcel.jar" />
    </findbugs>
  </target>
```

The `findbugs` element must have the `home` attribute set to the directory in which FindBugs is installed; in other words, *`$FINDBUGS_HOME`*. See Chapter 2, *Installing FindBugs™*.

This target will execute FindBugs on `bcel.jar`, which is the Jar file produced by BCEL's build script. (By making it depend on the "jar" target, we ensure that the library is fully compiled before running FindBugs on it.) The output of FindBugs will be saved in XML format to a file called `bcel-fb.xml`. An auxiliary Jar file, `Regex.jar`, is added to the aux classpath, because it is referenced by the main BCEL library. A source path is specified so that the saved bug data will have accurate references to the BCEL source code.

3. Executing the task

Here is an example of invoking Ant from the command line, using the `findbugs` target defined above.

```
[daveho@noir]$ ant findbugs
Buildfile: build.xml

init:

compile:

examples:

jar:

findbugs:
 [findbugs] Running FindBugs...
 [findbugs] Bugs were found
 [findbugs] Output saved to bcel-fb.xml

BUILD SUCCESSFUL
Total time: 35 seconds
```

In this case, because we saved the bug results in an XML file, we can use the FindBugs GUI to view the results; see Chapter 4, *Running FindBugs™*.

4. Parameters

This section describes the parameters that may be specified when using the FindBugs task.

`class`

> A optional nested element specifying which classes to analyze. The `class` element must specify a `location` attribute which names the archive file (jar, zip, etc.), directory, or class file to be analyzed. Multiple `class` elements may be specified as children of a single `findbugs` element.
>
> In addition to or instead of specifying a `class` element, the FindBugs task can contain one or more `fileset` element(s) that specify files to be analyzed. For example, you might use a fileset to specify that all of the jar files in a directory should be analyzed.

`auxClasspath`

An optional nested element which specifies a classpath (Jar files or directories) containing classes used by the analyzed library or application, but which you don't want to analyze. It is specified the same way as Ant's `classpath` element for the Java task.

`sourcePath`

An optional nested element which specifies a source directory path containing source files used to compile the Java code being analyzed. By specifying a source path, any generated XML bug output will have complete source information, which allows later viewing in the GUI.

`home`

A required attribute. It must be set to the name of the directory where FindBugs is installed.

`quietErrors`

An optional boolean attribute. If true, reports of serious analysis errors and missing classes will be suppressed in the FindBugs output. Default is false.

`reportLevel`

An optional attribute. It specifies the confidence/priority threshold for reporting issues. If set to "low", confidence is not used to filter bugs. If set to "medium" (the default), low confidence issues are supressed. If set to "high", only high confidence bugs are reported.

`output`

Optional attribute. It specifies the output format. If set to "xml" (the default), output is in XML format. If set to "xml:withMessages", output is in XML format augmented with human-readable messages. (You should use this format if you plan to generate a report using an XSL stylesheet.) If set to "html", output is in HTML formatted (default stylesheet is default.xsl). If set to "text", output is in ad-hoc text format. If set to "emacs", output is in Emacs error message format. If set to "xdocs", output is xdoc XML for use with Apache Maven.

`stylesheet`

Optional attribute. It specifies the stylesheet to use to generate html output when the output is set to html. Stylesheets included in the FindBugs distribution include default.xsl, fancy.xsl, fancy-hist.xsl, plain.xsl, and summary.xsl. The default value, if no stylesheet attribute is provided, is default.xsl.

`sort`

Optional attribute. If the `output` attribute is set to "text", then the `sort` attribute specifies whether or not reported bugs are sorted by class. Default is true.

`outputFile`

Optional attribute. If specified, names the output file in which the FindBugs output will be saved. By default, the output is displayed directly by Ant.

`debug`

Optional boolean attribute. If set to true, FindBugs prints diagnostic information about which classes are being analyzed, and which bug pattern detectors are being run. Default is false.

`effort`

Set the analysis effort level. The value specified should be one of `min`, `default`, or `max`. See Section 3, "Command-line Options" for more information about setting the analysis level.

`conserveSpace`

Synonym for effort="min".

`workHard`

Synonym for effort="max".

`visitors`

Optional attribute. It specifies a comma-separated list of bug detectors which should be run. The bug detectors are specified by their class names, without any package qualification. By default, all detectors which are not disabled by default are run.

`omitVisitors`

Optional attribute. It specifies a comma-separated list of bug detectors. It is like the `visitors` attribute, except it specifies detectors which will *not* be run.

`chooseVisitors`

Optional attribute. It specifies a comma-separated list of bug detectors prefixed with "+" or "-" to selectively enable/disable them.

`excludeFilter`

Optional attribute. It specifies the filename of a filter specifying bugs to exclude from being reported. See Chapter 8, *Filter Files*.

`includeFilter`

Optional attribute. It specifies the filename of a filter specifying which bugs are reported. See Chapter 8, *Filter Files*.

`projectFile`

Optional attribute. It specifies the name of a project file. Project files are created by the FindBugs GUI, and specify classes, aux classpath entries, and source directories. By naming a project, you don't need to specify any `class` elements, nor do you need to specify `auxClasspath` or `sourcePath` attributes. See Chapter 4, *Running FindBugs™* for how to create a project.

`jvmargs`

Optional attribute. It specifies any arguments that should be passed to the Java virtual machine used to run FindBugs. You may need to use this attribute to specify flags to increase the amount of memory the JVM may use if you are analyzing a very large program.

`systemProperty`

Optional nested element. If specified, defines a system property. The `name` attribute specifies the name of the system property, and the `value` attribute specifies the value of the system property.

`timeout`

Optional attribute. It specifies the amount of time, in milliseconds, that the Java process executing FindBugs may run before it is assumed to be hung and is terminated. The default is 600,000 milliseconds, which is ten minutes. Note that for very large programs, FindBugs may require more than ten minutes to complete its analysis.

`failOnError`

Optional boolean attribute. Whether to abort the build process if there is an error running FindBugs. Defaults to "false"

`errorProperty`

Optional attribute which specifies the name of a property that will be set to "true" if an error occurs while running FindBugs.

`warningsProperty`

Optional attribute which specifies the name of a property that will be set to "true" if any warnings are reported by FindBugs on the analyzed program.

`userPreferencesFile`

Optional attribute. Set the path of the user preferences file to use, which might override some of the options abobe. Specifying `userPreferencesFile` as first argument would mean some later options will override them, as last argument would mean they will override some previous options). This rationale behind this option is to reuse FindBugs Eclipse project settings for command line execution.

`nested`

Optional attribute which enables or disables scanning of nested jar and zip files found in the list of files and directories to be analyzed. By default, scanning of nested jar/zip files is enabled.

Prev
Chapter 5. Using the FindBugs GUI

Home

Next
Chapter 7. Using the FindBugs™ Eclipse plugin