

Updates to Java Development with Ant

Steve Loughran Erik Hatcher

September 11, 2002

Abstract

This document covers changes, errata and discoveries since the publishing of Java Development with Ant [1]. This is not an attempt to keep it up to date with what is happening with Ant1.6 or Ant2.0. For that, get on the ant-dev mailing list and join in the project. If you are a good Java programmer, you can always find a project in Apache's Jakarta or XML projects to keep you busy.

1 Introduction

Physics has analogies in software, and books about software. Because one of use who has worked in High Energy Physics (control software for matter/antimatter colliders), we are allowed to use these analogies.

The Heisenberg Effect: the act of observing changes that which you are observing. Often it is easier to fix a bug than to explain why something doesn't work. Sometimes you explain why it isn't working, then it gets fixed, so you have to go back and strip out the explanation. The end result is beneficial, but the development effort is higher. You can follow our progress through this book by looking at defects filed against Ant in bugzilla, or the CVS changelog recording those fixes we (and others) made to correct them.

Schrödinger’s Cat: something can be neither alive nor dead until you open the box and the quantum state collapses into either of the two possible outcomes. Although in Ant1.4 it was claimed that properties were immutable, we discovered this wasn’t true. The ramifications of that fix involved some fundamental changes to Ant, and there is still one glaring loophole, `<available>`.

The Inevitability of Entropy: our book was finished on the same day that Ant1.5 was released. Right up to the end we were having to track changes to Ant, such as the last minute addition of the subversion patterns to the `defaultExcludes` patterns.

In open source projects change is not only ongoing, it is visible to anyone who grabs the source, subscribes to the mailing list or gets the “have you tried the nightly build” response to a bug report.

We aren’t going to try and track all of the ongoing changes in this document. If we write a second edition, an action which not only needs motivation, it requires signoff from our respective spouses, then we will worry about that. For now, we are all taking a well-earned rest.

However, we do want to provide a few pointers and notes about where we now know we were in fact horribly wrong, or where something has changed so significantly that it invalidates part of the book.

2 Ant1.5.1

Ant 1.5.1 is a quick update to the Ant1.5 release; bug fixes only. The main area for change is probably in the boot scripts, especially for cygwin, changes that do not impact any build files.

There will be other minor fixes; we will indicate where they impact our book when the new release ships.

3 Errata

We’d like to thank everyone who has found an error and told us. So far nothing major has cropped up. We should really be doing defect tracking, defect salting

and errata discovery graphing to estimate the number of likely defects remaining.

3.0.1 About the cover illustration, Page xxxiii

The city of Bombay is now officially called Mumbai.

3.0.2 Page 56

The `<include>` and `<exclude>` elements do not parse into separate patterns by comma, so you cannot send a list of patterns, as we do in this example, an example nominally about nesting patternsets.

Replace

```
<patternset>
  <include name="**/*.gif,**/*.jpg"/>
  <patternset>
    <exclude name="**/*.txt,**/*.xml"/>
  </patternset>
</patternset>
```

with

```
<patternset>
  <include name="**/*.gif"/>
  <include name="**/*.jpg"/>
  <patternset>
    <exclude name="**/*.txt"/>
    <exclude name="**/*.xml"/>
  </patternset>
</patternset>
```

3.0.3 Page 74, Section 3.12.6, Building with a different library version

Replace

```
<property name="${lucene.jar}" location=... />
```

with

```
<property name="lucene.jar" location=... />
```

3.0.4 Page 83, Summary bullet points

We mixed up / and \ when differentiating Unix and MS-DOS file paths.

Replace

the MS-DOS conventions of semicolon (;) and slash mark (/) or the Unix conventions of colon (:) and backslash (\);

with

the MS-DOS conventions of semicolon (;) and backslash (\) or the Unix conventions of colon (:) and slash mark (/);

Of course, in your build file it makes no difference whatsoever, because, as the bullet point explained, Ant lets you use either and sorts it all out” `c:/dir1\dir2/dir3\file.xml` is a valid path to ant ¹.

¹But not in property files; backslashes need to be doubled up to stop the Java properties file loader from going to work on them. We get burned by this sporadically.

3.0.5 Page 353, Listing 14.4, Line 8

The trailing quotation mark for the `publicId` attribute of the `<dtd>` element got lost. What you should see is :

```
<dtd publicId="-//Sun Microsystems, Inc.//  
DTD Enterprise JavaBeans 1.1//EN"  
location="${j2ee.dir}/${j2ee.subdir}/ejb-jar_1_1.dtd"/>
```

Thanks to Eric Jablow for finding this.

3.0.6 Page 404, Third Bullet

Ends with

“...but not it is not a required component to run Gump.”

The first “not” should be dropped.

4 Chapter 5: Executing programs

We left out any mention of the `vmlauncher` attribute of `<exec>`, because we didn't think it was that important, but one of the submissions for *The Elements of Ant Style* suggested always setting it, so lets give it a quick explanation.

Before Java1.3, there was no way to specify the current directory when `exec()`-ing a program in Java. To let you change directory in `<exec>`, Ant has helper scripts, `antRun.bat` and `antrun.sh` that will change directory and then run your program, from inside the batch file.

Java 1.3 and above do have a library call to set the directory when you run a program, so Ant will, by default, defer to the OS on this platform. It should work on all Java1.3 platforms, and because it depends upon no external batch files, is less brittle. IDEs that dont set `ant.home` properly, platforms that don't have a helper application: these systems can `<exec>` with `vmlauncher="false"` version, when they may not run with `vmlauncher="true"`.

Why then, did we leave the suggestion to run with `vmlauncher="true"` in the style guide? Because the submitter had clearly encountered problems with the Java runtime. Executing via the helper scripts does have some advantages. Firstly, it offers consistency: execution will be the same on Java1.2, Java1.3 and up. Even on Java1.1, though of course we strongly discourage continued use of that platform.

Secondly, running from inside a batch file or shell script may give you more access to OS specific stuff. In the batch file in particular, all the parameters are joined together into one big command, into which you could *maybe* stick in shell commands, as you now know that you are running inside a batch file that is executing the following line built from parameters to the `<exec>` task:

```
%ANT_RUN_CMD% %PARAMS%
```

Neither of us have ever needed to do many fancy shell commands from inside a build file, so we haven't done this —and, whenever the need arises, we turn to section 5.3.4 in our book to remind ourselves how to start a shell explicitly. That, and the fact we never run on anything less than Java1.3 means we don't have to worry about consistency.

There is one final reason to launch programs via the helper scripts, and that is that there is a recurrent bug report related to running relative programs in a different directory to the current one :

```
<exec executable="../bin/validate" dir="xml" />
```

When you use the `vmlauncher`, sometimes it won't find the relative executable; part of the problem is the choice of what happens first: the directory change or the location of the program, and in a multi-level build file process (see chapter 9), the location of `"."` in the `executable` attribute is actually that of the first directory in which you invoked `ant`, not the current build file.

When you set `vmlauncher="false"` you know that the directory change takes place before the execution. And, equally importantly, the batch file changes to the directory of the current directory when no `dir` attribute is set, so relative paths are always resolved in relation to the current base directory. For this reason, you may always want to consider disabling the `vmlauncher` when executing relative programs.

There is an alternative solution to the same problem: use a

```
<property location="../bin/validate">
```

assignment to resolve the relative path to an absolute one before calling `<exec>`. This works with all launchers, is portable, and easy to debug when running ant in `-verbose` mode.

5 Chapter 6: Packaging and Documentation

We understand the indexing function in JAR files better now; only applets make use of the information (we think), and to work properly the classloader needs to have a pre-prepared index of all dependent JAR files, as well as in the classes of the JAR containing the index. Without these, it is not effective.

6 Chapter 12: Web Applications

Look at the reports on bugzilla vis-a-vis `<jspc>` before using it; quirks and issues on this little task are still surfacing. Some need to be fixed in Ant, others in Jasper.

Ant1.5.1 will have a version of `<jspc>` that will only work with Tomcat4.1 or later versions of Jasper, versions that create classes with `_jspc` at the end of the classname. These created classes should be ready to import into a web application, especially if you get the `<jspc>` task to create the web.xml declarations mapping jspc paths to java classes.

7 Chapter 14: EJB

We should mention that we know that EJB beans shouldn't access the file system, something our example session bean does when searching the index. Obviously you can't go near the file system when you want transacted access, and even with this read-only example there is the problem of what to do if your system needs a hot update. The other reason for not doing file access is resource management; app servers don't manage resource handles, so theoretically you could run out of handles under heavy load. Of course, the sockets used for all inbound network requests, be they RMI, Corba, HTTP or SOAP are all file

handles too, so that argument is a bit tenuous, but it is there anyway. We should really move our index to a database, though a transacted file system would be cooler.

8 Chapter 15: Web Services

(commentary by Steve)

Apache Axis is nearly ready. This changes little in the chapter, but there are a couple of points to note.

Firstly, Axis now ships with its own version of an Axis health page, also called `happyaxis.jsp`. This is not a coincidence; I added it to Axis and based it upon the example in this chapter. However, it is a pure JSP page, with nary a taglib in site. Although this is considered bad practice, it does mean that the page can be dropped into any webapp without any extra code or configuration.

The Axis `happyaxis.jsp` is the better one for testing the health of Axis; it has more tests and more diagnostics. But for your own webapps, we strongly believe that writing your own happiness pages through taglibs is the safest and most rigorous approach.

Secondly, there is another JAR that axis needs, `commons-discovery.jar`. This extra JAR came in at the last minute, and can catch out people who have been working with the beta copies. If you work inside `axis.war`, this is not an issue, but if you follow our process to add Axis to a new WAR file, then this JAR needs to come across into `WEB-INF/lib`. The Axis `happyaxis.jsp` page probes for this.

If you want to get into Web Service dev with Axis or the Sun reference implementation of JAX-RPC, you need to read the JAX-RPC specification [3].

This little bunny is the definitive guide to what SOAP and WSDL data types you can handle with JAX-RPC implementations, what the mappings are, how method names get converted, what the exception mechanism is etc. Actually the exception handling mechanism is a bit vague, at least the cross-platform ramifications are —I am going to have to spend some time experimenting with this to find out exactly what is going on.

Tip: build your web service entry points with debug information included, the WSDL generation code will use the symbol table information to get the parameter names right.

Apache Axis is now an interesting example of how to use Ant, and bits of it bear more than a passing resemblance to both this chapter and chapter 9, the chapter on structuring large projects. The `test/httpunit` test suite is derived from the examples in chapter 12 and 15; the whole structure of the many build files takes on aspects of chapter 9 and raises it a level, certainly enough to discover memory leaks in Ant when using the ant-contrib `<foreach>` task on a very large scale. It is well worth a look from a big-Ant-project perspective.

Finally, I am a little behind on getting one of the citations, *Making Web Services that Work* out the door; but it will be ready soon.

9 How this update was made

For the curious, this document was written as a `.tex` file and then turned into a PDF file using MikTeX, calling the `pdflatex` program from inside Ant. Emailing it out is a trivial followup action.

```
<exec executable="pdflatex"
  failonerror="true">
  <arg value="-silent"/>
  <arg value="-c-style-errors"/>
  <arg value="-interaction" />
  <arg value="nonstopmode"/>
  <arg file="${title}.tex" />
</exec>
```

Building PDF documents during a build is a useful technique for any project involving specifications and other complex documents. We are not convinced that \LaTeX is the best choice here, but with DocBook the only alternative we are somewhat reluctant to migrate. To build PDF from DocBook you could use FOP from xml.apache.org. That would be harder work to set up but an interesting option in the long term.

10 How the book was made

Some day soon we will explain how the book was done, and how books could be done better in future in a little paper, *Refactoring the Publishing Process* [4]. For now, key points are:

- CVS server on a Redhat 7.1 system, “eiger” in the home DMZ.
- Home stateful firewall (WebRamp) set to allow port 22, ssh through.
- Office XP with tracking enabled.
- Ant wherever we could.

Steve says “I don’t know how it has ended up that you need to have a home DMZ to keep your server and 802.11b LAN separate from your other boxes, with two levels of firewall to make deploying across a house complex, but it has. It’s bad enough defending against script kiddies, pretty soon I’ll have to worry about the RIAA too.”

11 Acknowledgements

Products used while writing this book; all highly recommended.

- jEdit 4.0
- IntelliJ IDEA 3.0 beta releases. Ant refactoring in an IDE. Woo-hoo!
- laptops with lots of memory.
- Steve’s CDs: Chemical Brothers, Massive Attack, Paul Oakenfold.
- Illy coffee.
- Volkswagen Passat GLS 1.8Turbo

References

- [1] Hatcher, E. and Loughran, S. *Java Development With Ant*, Manning Press, 2002.
<http://manning.com/antbook> 1

- [2] Loughran, S. *Mount Hood*, 2001,
<http://iseran.com/Stories/hood.html>
- [3] Java Community Process. *JAX-RPC Specification*, 1.0,
<http://java.sun.com/xml/downloads/jaxrpc.html> 8
- [4] Loughran S., and Hatcher E. *Refactoring the Publishing Process*. 10