Sams Teach Yourself ASP.NET in 24 Hours

Copyright © 2003 by Sams Publishing

International Standard Book Number: 0672325624

Warning and Disclaimer

When reviewing corrections, always check the print number of your book. Corrections are made to printed books with each subsequent printing. To determine the printing of your book, view the copyright page. The print number is right-most number on the line below the "First Printing" line. For example, the following indicates the 4th printing of a title.

First Printing: June 2003

06 05 04 03   10 9 8 7 6 5 4

Misprint	Correction
Page 26, last paragraph: For this example, the buildfile echoes the phrase, 'Ant is working **correctly**' to the terminal from which Ant is running.	For this example, the buildfile echoes the phrase, 'Ant is working **properly**' to the terminal from which Ant is running.
Page 34, next to last sentence in second paragraph: In Listing 2.8, the property dirs.source is created to specify the absolute **pathnames of that directory's**.	In Listing 2.8, the property dirs.source is created to specify the absolute **pathname the directory**.

Page 38, second sentence in "Story" sidebar: The **only** problem with this target is that it leaves the backup on each developer's local machine.	The **one** problem with this target is that it leaves the backup on each developer's local machine.
Page 83, third bullet: Forking will allow the **JVM** to take advantage of a multiprocessor platform.	Forking will allow the **build** to take advantage of a multiprocessor platform.
Page 137, caption for Figure 5.3: This JunitReport output is formatted with frames **format**.	This JunitReport output is formatted with frames.
Page 140, first sentence of first paragraph after "Story" sidebar: CrusieControl is more than a platform-independent cron **job** for build processes.	CrusieControl is more than a platform-independent cron for build processes.
Page 219, first sentence of last paragraph: An example of the resulting **instrumenting** code is shown in Listing 7.13.	An example of the resulting **instrumented** code is shown in Listing 7.13.
Page 230, last bullet: Automated processes will be more consistent than **a group of developers**.	Automated processes will be more consistent than **manual processes**.
Page 287, last sentence of next to last paragraph: In creating this custom task, we had a chance to see how to use the Regular Expression mapper available in Ant, and how to deal with multiple <filesets> that need to be uniquely identified.	In creating this custom task, we had a chance to see how to use the Regular Expression mapper available in Ant, and how to deal with multiple <filesets> that need to be uniquely identified **within a custom task**.
Page 374, last paragraph: What FileSet will the **execute** method see?	What FileSet will the **execute()** method see?

New Appendix C

The Future of Ant

Keeping Your Build Processes Up to Date

At one point, a radical departure from the implementation and features of Ant 1 had been proposed for Ant 2. As of the time this book was written, future Ant development is not expected to be taking radical steps, but an incremental, evolutionary approach.

It is worth considering how to create Ant buildfiles that will stand up over time as new versions of Ant are released. While no one can predict exactly what Ant will look like at some point in the future, there are principles of good design that can be applied that will help to insulate a development team from the changes that are bound to occur. Let's consider some of these principles:

- Study the new release and keep up with the intent of the new release. It's possible to have buildfiles that work, but use archaic techniques. For example, certain tasks may accept a <fileset> where it didn't before. Modify your build files to use more up-to-date techniques.

- Refactor your buildfiles on a regular basis. If you find, for example, that several targets have been added that are minor cut-and-paste variations of one another, take the time to refactor it. This is true of good object-oriented design practices in code, and the same principle applies here.

- Remove deprecated tasks in your buildfiles, because these are guaranteed to eventually disappear in future versions of Ant.

- Keep files clean and well-formatted. Even though it's not mandatory to indent XML files, doing so will make the buildfile easier to read and maintain. It will also make potential problem areas stand out, such as the use of deprecated tasks or obsolete syntax.

- When a new version of Ant is released, read the release notes to assess impact. Also, check the manual for deprecated tasks, and avoid using them in new buildfiles.

- Set aside time to go back and modify existing buildfiles to remove deprecated tasks.
- Make sure you're currently using a recent version of Java. For Ant 1.6, you'll need to use Java 1.2 or above.
- Make sure you're currently a recent version of JAXP. For Ant 1.6, you'll need to use JAXP 1.1 or above.
- Regularly perform spike tests with new versions of Ant when they become available. This includes minor releases. Even if you stay with your current version of Ant for a while, doing this will provide additional insight into what to do in your current Ant buildfiles to minimize the impact of transitioning to a newer version of Ant.

By keeping current with Java and JAXP upgrades, the transition to newer versions of Ant will be easier. You won't be simultaneously faced with the Java and JAXP upgrade issues, along with Ant upgrade issues.

Also, strive to keep up with the changes in new Ant releases. This doesn't mean that you have to immediately adopt each new release, but someone on your team should be assessing the changes and what affect they will have on your overall build and deployment process. As with any product upgrade, the longer you wait to adopt a newer version (or to even assess the impact), the more painful the upgrade will become.

Other areas that might be impacted by a new version of Ant are the custom components you have written. This includes custom tasks, loggers, listeners, filters, data types, and so on. Here are some suggestions for keeping custom components up to date with the latest version of Ant.

- We suggest first that you maintain a separate build environment for building and packaging custom components. This will use the same version of Ant that all of your other build processes use. Use this Ant buildfile to compile and package your custom components into a jar file.
- Create and maintain unit tests for your custom components.

- Make use of what's already available in Ant. For example, in this book, we have a custom task that uses Ant's regular expression engine. We could have instantiated our own, but why introduce unnecessary complexity? This leads to code that requires more work to retrofit and is difficult to maintain. By using the features already available in Ant, you will make the task of upgrading to newer versions of Ant easier.
- When a new version of Ant becomes available, use your existing version of Ant to run your suite of unit tests against the new Ant jar files, and verify that everything works as expected.
- Consider donating custom tasks to the Ant project if they might be of use to others.
- As we recommended earlier in this book, with the XP approach to Test First Design, write your unit tests first, and then the code for your custom components. You'll write better code, and you'll also develop a suite of unit tests that will help in maintaining your code in the future.

If you make use of BSF scripting, remember to test any buildfiles that have scripting code in them to verify that they will run as well.

Perhaps the most important thing to remember with writing Ant tasks is to focus on the concepts rather than the syntax. A developer who gains a solid grasp of object-oriented design techniques will find it easy to transition between C++ and Java. In a similar fashion, a developer who has a grasp of the design of build and deployment processes, especially in an XP methodology, will be able to easily transition their build processes to newer versions of Ant.