Ant1.7.1/Planning

Ant1.7.1 Plans

Here are some draft thoughts on Ant1.7.1.

First, what is the goal? Bugs only, or bugs+features?

- Consensus is that Ant1.7.1 is bug fixes for regressions in Ant1.7.0

Goal

- Get all fixes since ant 1.7.0 into the hands of users
- Refresh dependent libraries (xerces &c)

Release Plan

- Release manager? (kevj volunteered after stalling last release)
- Planning: ApacheCon + email lists: may 2007
- beta: june/july 2007?
- Release: 6+ months after Ant1.7.0

Defects

Must Fix Defects

Java 1.6 support

-all tests passing on Linux, Vista and XP

- who has access to Vista, and which version? I really don't want to install it, even under VMWare!
  XavierHanin: I do have access to Vista Professional edition

  SteveLoughran: I have a VMWare image of Vista Enterprise Edition.
  ..I think we may need to change the install instructions to deal with the many dialogs that pop up before you can edit the env variables.
  JanMatèrne: I'll set up a VMWare image of Vista Ultimate Edition (32bit, German)
  - [http://issues.apache.org/bugzilla/show_bug.cgi?id=41958 Javadocs breaking]

Would like to fix Defects

* Better proxy support on Java 5+

Features

Ivy

- Bundle Apache Ivy 2.0.0?
  - when is ivy2.0.0 going to pass incubation?
    XavierHanin: Difficult to say, it depends on external factors. The major point for the moment is to recruit new committers.
- modify fetch.xml to use Ivy.

Scripting

- make <scriptcondition>, <scriptselector>, etc, more useful through : attributes and elements, re-use
- resource enable script tasks, to take source from any resource
  (done for <scriptdef> as of 1-may-2007)
- more docs
- should we include a scripting language? such as BSF+Rhino for a pre-java1.6 build?
  - need to vote on bundled language - I guess Rhino/JavaScript has mindshare right now

Java 1.6 enhancements

- Read filesystem properties in the packaging tasks. How to do this in a way that is x-platform and reliable? Add a new "useproperties" attribute to zip/zipfileset with "always", "never", "unix", and something on untar/unzip?
- Selectors for selecting on system properties
- chmod to switch to java API? Benefits?
- non-echo input handler for password entry (done)

Big Project support

- BigProject logger with more logging of which project we are in (entry, exit, full location on an error), quiet on targets with no output, and in -verbose mode, log which build file a target actually came from. Maybe print finish time too.

  - done but needs an XML version for use under Cruise Control.

Internals

- add a lifecycle for datatypes?
- add an addExpandedText() method that expands props in nested text before assigning
- add an UnexpandedString() datatype that can be used in attribute assignments for classes that don't want properties expanded...we could use this in macro processing to prevent double expansion of properties. (this would fix a few bug reports too)

User Experience

- Error messages to improve?
- what about the "no build.xml" message? Add something pointing at the ant docs/wiki?
- make the no-task found diagnostics recognise when they are loaded under maven (how?) and warn user that they are in an unsupported configuration and that they have to set up all dependencies in the ant plug in pom. We are already IDE-aware, after all.
- produce an official izpack containing Ant+core libraries.
- produce an official izpack containing Ant+core+apache libraries (Xalan, BSH, &C). And JUnit?
- produce our own RPMs. Which would imply we support them.
- produce our own .deb (dpkg) as the Debian versions are always lagging and Ubuntu is becoming important in the Linux space and is based on Debian.

What to 'remove'/'trim'

- Which JDK are we now building on? (1.2, 1.3 or 1.4?)
- When we know the build JDK, what code can we refactor/strip out?

= Postponed For Ant 1.7.2 ==

Other enhancements

- add an <xmlmessage> element for mail that takes an XML message, so you dont need to CDATA escape XHTML messages.
- add a <antfork> task that runs a completely new ant process, with different env, maybe even JDK. Sometimes people need this; running forrest builds is just one example. We'd use <java> to set it up.
- add support to <subant> a graph of targets, not just a list, so it can skip all dependents of a failing build
- any way to integrate download and declaration of an antlib?

'Antlibization'

- Which optional tasks would be better as an antlib for separate download?

- SCM: Continuus/Synergy Tasks
- SCM: Clearcase Tasks
- SCM: Microsoft Visual SourceSafe Tasks + SourceOffSite
- SCM: Starteam Tasks
- SCM: Cvs + CvsChangeLog + CvsVersion + CVSPass + CvsTagDiff

Testing

- implement something similar to <functionaltest> that steve uses in smartfrog for starting smartfrog and running tests against it.
- allow <waitfor> to fail if it times out (<functionaltest> will need this)
- enhancements to the XML reports format and stylesheets? TestNG and the CI tools folk may have input here.
- could we export a list of failing tests and use this as input for the next test run? to only run the tests that failed?
- could we use resources as a list of test classes to run, so drive off .class files in a JAR?

Resource support

- script tasks to load scripts from named resources (done)
- import task to import from a given resource
- junit to select lists of tests to run from resources

this may imply we need a way to get one resource to resolve a relative link to another resource...a new interface.