

Why switch eXist-db from Ant to Maven?

adam@exist-db.org

Why move away from Ant?



The Current Situation

- Lots of pain associated with our current Ant build system
 - Completely ad-hoc project layout
 - 10 years of evolution with lots of committers = a big heap of code + many different approaches
 - Maintenance is difficult
 - Integrating eXist-db cleanly with another project is almost impossible
 - Developer learning curve is steep!
 - After 5+ years, I still do not know all the Ant targets (or care about them).
 - But... It works!

Why move away from Ant?



The Current Situation – the hard facts

- Our current Ant build system comprises -
 - > 6807 lines of Ant (many include files)
 - Maintaining relative paths from include files is tricky
 - > 182 lines of XSL
 - > 227 lines of build.properties
 - 99 lines of Bash script
 - 92 lines of Batch script
 - 2 Ant plugins – IzPack and Ant Fetch
- In addition we have to manually and independently maintain -
 - Support for 3 IDE's (Eclipse, NetBeans, IntelliJ)

Why move away from Ant?



There 'must' be an easier way!

- The Goal – Make Life Simpler!
 - Spend less time hacking the build scripts and more time hacking code.
- Desired Outcome -
 - Reduction in build script code
 - Reduction in complexity
 - Automated dependency management
- How to achieve?
 - Reduce explicit definitions
 - Build by convention
 - Specify the 'What and not the How'
 - Build systems have improved e.g. Buildr, Gradle, Maven

Why Maven and not...



There 'must' be an easier way!

- Maven provides both building and dependency management
 - We need both
- Yes, We could just use Ant for building and Ivy for dependency management ← If you only want to solve ½ a problem!
- Why Maven and not X?
 - Looked at others -
 - Gradle, non-XML Syntax and dependency on Groovy
 - Buildr, non-XML Syntax and dependency on Ruby
 - Not well established
 - Maven does what we need
 - We already have Maven knowlegde in the eXist-db community

What Maven gives you



Robust dependency management

- Keeping JARs in SVN is wrong!
 - JARs are duplicated on every copy/branch operation.
 - JAR associated Source Code and JavaDoc is hard.
 - Increases SVN checkout size and time.
 - Multiple versions of multiple JAR's have to be maintained.
- Reducing JAR dependencies by using Reflection is wrong!
 - Complicates code, hides intention.
 - Dependencies are not self-evident.
 - Fails later rather than sooner!
- SVN provides no dependency management, protection or conflict resolution!
 - Manually manage versions of JAR's you depend on (and that they depend on).
 - Manually test for broken references when new versions of JAR's are added.
 - Manually test dependency references when you add new versions of JAR's.

What Maven gives you



The features for us

- Standard project layout by convention
 - Customisable (best avoided)
 - Now used for almost all other Java build tools
- Single POM (configuration) file* -
 - Define the 'What' not the 'How'
 - Define and resolve the dependencies externally from VCS.
 - Manage the builds
 - Manage the releases and VCS branched (including JAR, WAR, IzPack artifacts)
 - Produce resources that integrate well with other projects and build systems
 - Support multiple IDE's with zero overhead (NetBeans, Eclipse, IntelliJ etc)

What we gain

- Simpler more powerful build system
- Automated Dependency Management
- Release management
- First step of eXist-db Modularisation...
 - Its easier to break a project into modules when your build system supports this at almost no cost
 - Why modularise eXist-db?
 - Cleaner separation of concerns
 - Faster builds
 - Faster tests and test feedback
 - Easier to contribute

With and without manual dependencies

- What impact do JAR's in SVN have on eXist-db?

	trunk with JARs	trunk without JARs	Change
Size	111 MB	78 MB	-33MB / ~30%
Checkout Time	07:32	05:18	02:14 / ~30%
On-Disk Size (with .svn metadata)	249 MB	182 MB	-67MB / ~27%

Any questions?