Evolution of Java(TM) Software on GNU/Linux

Dalibor Topić
Java F/OSS Ambassador
Sun Microsystems

http://robilad.livejournal.com

Dalibor.Topic@Sun.com
OpenJDK
License

- **GPL v2**
  - No proprietary forks
  - Popular & trusted license
  - Compatible with GNU/Linux
  - Fostering adoption

- **+ Classpath Exception**
  - Programs can have any license
  - Improvements remain in the community
  - FSFs license for GNU Classpath

- No proprietary forks
- Popular & trusted license
- Compatible with GNU/Linux
- Fostering adoption
Why GNU/Linux?

- **Values**
  - Freedom as a core value

- **Stack**
  - Free Software above and below the JVM

- **Demand**
  - Increasing demand for Java integration
Linux distributions

- Linux kernel
- GNU libc + utilities
- X11, GNOME, KDE, ...
- Package management
- Built-in way to download, install, manage, uninstall all software in a distribution, including dependencies, from a single source
- Killer feature!
Package management

- Sources → Binaries + Metadata + Glue
- Sources = upstream source + patches
- Binaries = 1..N packages from build
- Metadata = versioning, deps, description
- Glue = (de)installation scripts, etc.
Benefits of package management

- Installation state in packaging database
- Anyone can rebuild anything anytime
- Creating patched/new packages possible
- Easy to customize distributions
- Built-in integrity & security checks
Leads to ...

• All software installable as packages
• Thousands of interdependent packages
• Package repositories
• Demand for stable releases
• Consolidation
Further benefits

- Ability to 'rebuild the world' from scratch
- Implications for security & QA
- Bill of materials (licenses, etc.)
- Build logs
How to scale

- Make room for error in versioning
- Versioned dependencies, ranges, ...
- Epochs
- Try to only have one version of a library
- Introduce virtual dependencies
- Separate build and runtime deps
- Separate development and stable
- Welcome contributions, but enforce a strict social process
So much fun, everyone is doing it

- Haskell : Hackage/Cabal
- Lua : Rocks
- Perl: CPAN
- PHP: PEAR
- Python: EasyInstall/eggs
- Ruby: Gems
- …
- Hold that thought!

where is Java?
Distributing software for Linux

- As source code
- As a binary package
- More than 300 distributions
- At least 6 major ones
- At least as many packaging formats, processes, guidelines
- Not very appealing to Java developers
- Who are used to passing JARs around
- Very low overhead, works pretty well
Pragmatic approaches

• One way: Pick the ones you care about
• Rely on community for the rest
• Another: OpenJDK 6 is source code only
• Patching+Packaging by IcedTea&Distros

• Both a technical and a cultural gap
OpenJDK 6 Status

- Fedora
- Ubuntu
- Debian
- Gentoo
- OpenSUSE
- Mandriva
- ... and others
Building GlassFish v3 on Ubuntu

- Needs to build from source for 'main'
- Requires Maven2 to build
- Maven2 is a build tool with a large JAR repository
- Maven2 downloads 500+ JARs
- ~ 150 of them are third party libraries
- Phew: Most of them in multiple versions
- Oh no! Many of them unpackaged
- Repeat the work for each dependency
The GlassFish Dependency Graph
Problems

• One needs to track down third party library versions, transitive dependencies, and source code
• If documented, hard to compile into 'package library X in version Y with source code URL Z' form
• No single source of metadata to make the analysis a matter of minutes
• Poor maintenance of binary compatibility between consecutive (open source) library versions
Java Software Deployment

- First Problem: Where is my JVM?
- Solved by OpenJDK 6
- Available in a Linux distro near you
- Or coming to ... soon
- The JVM isn't a second class citizen on Linux any more
Java Software Deployment

- Second: Where are my dependencies?
- SVN? Maven? OSGI?
- Packaged by distro would be best
- `apt-get build-dep openjdk6`
- OpenJDK 6 provides foundation for packaging work
- Increasing interest in providing Java software as packages on top of it
- Still a lot of work to do – everyone's turn
The good old Java way

• JARs = ZIP files + a bit of metadata
• Metadata: “attribute: value” pairs
• Often distributed without source code
• Even for open source software
• Rarely used existing features:
  > Version & Class-Path metadata
  > Package sealing
  > cryptographical JAR signing
• Mildly frustrating for everyone.
JAR Hell

- Predictable outcome of the Java way
- One $CLASSPATH per ClassLoader
- If two JARs with the same library are on $CLASSPATH, the first one wins
- If the first one is not sealed, classes in packages in first one could still be loaded from the second one
- If those classes depend on incompatible versions of classes existing in both the first and the second JAR: FAIL
Getting out of there

• Existing JAR/Manifest mechanism is inadequate and/or unused
• Get developers to version their stuff
• To provide dependency information
• Make it all part of the standard JDK
• Put right in the language
• OpenJDK modules project
Modules to the rescue

• New concept: module
• One module can contain many packages
• Some classes can be 'module-private'
• Dependencies and versioning info can be specified at source code level
• Modules can live in repositories
Benefits

• Developers: express versioning and dependency metadata in the code
• Packagers: extract and analyze metadata on its own
• Basic building blocks for Java module distributions for end users
Java on Linux

- Linux Foundation : working on adding Java as Trial Use module to LSB 4.0
- OpenJDK 6 : in 'core'/'main' section of Fedora, Debian, Ubuntu
- Trickling down into derived distros
- More open source Java projects looking into packaging for Linux now
- Distributions interested in interoperability of package management tools with Java modularity solutions
Thank you for coming!

http://OpenJDK.java.net

dalibor.topic@sun.com

irc://irc.oftc.net/#openjdk