Evolution of Java(TM) Software on GNU/Linux

Dalibor Topić
Java F/OSS Ambassador
Sun Microsystems

http://robilad.livejournal.com

Dalibor.Topic@Sun.com

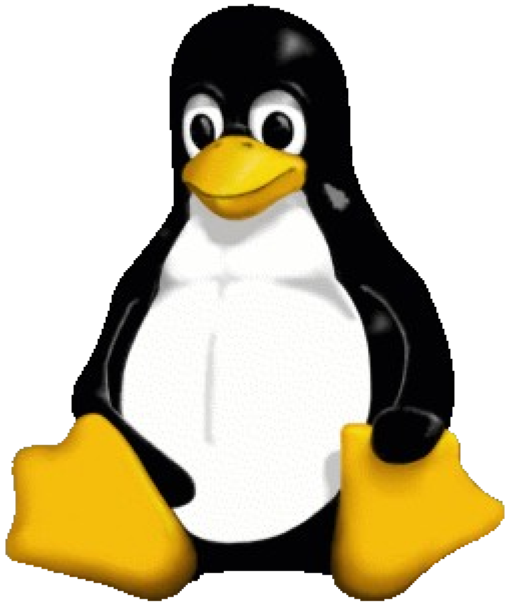
Java™

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



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
- Builtin 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
- … where is Java?
- Hold that thought!

Distributing software for Linux

- As source code
- As a binary package
- More then 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

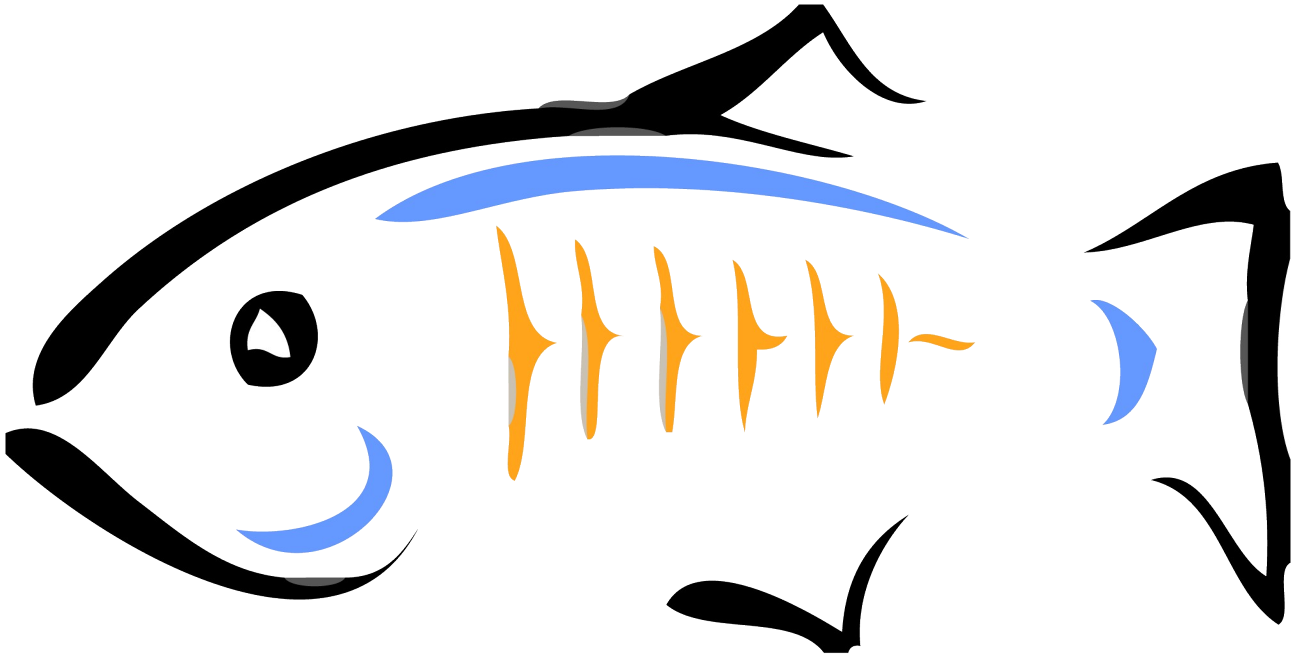
Pragmatic approaches

- 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

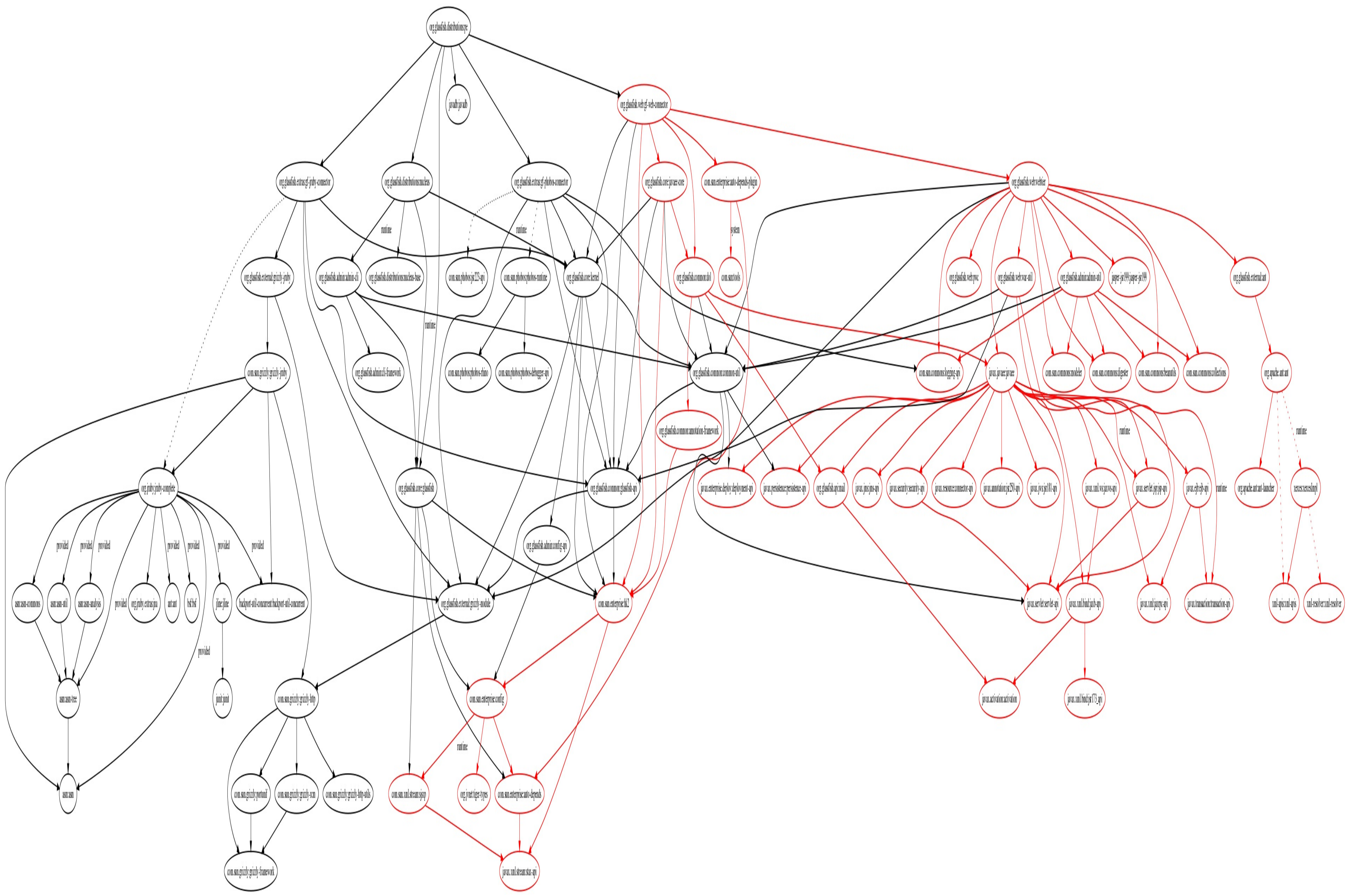
- Fedora
- Ubuntu
- Debian
- Gentoo
- OpenSUSE
- Mandriva
- ... and others



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



org.glassfish.distributions:pe
javadb:javadb
org.glassfish.web:gf-web-connector
org.glassfish.extras:gf-jruby-connector
org.glassfish.distributions:nucleus
org.glassfish.extras:gf-phobos-connector
org.glassfish.core:javaee-core
com.sun.enterprise:auto-depends-plugin
org.glassfish.web:webtier
runtime
system
org.glassfish.external:grizzly-jruby
org.glassfish.admin:admin-cli
org.glassfish.distributions:nucleus-base
com.sun.phobos:jsr223-api
com.sun.phobos:phobos-runtime
org.glassfish.core:kernel
org.glassfish.common:dol
com.sun:tools
org.glassfish.web:pwc
org.glassfish.web:war-util
org.glassfish.admin:admin-util
jasper-jsr199:jasper-jsr199
org.glassfish.external:ant
com.sun.grizzly:grizzly-jruby
org.glassfish.admin:cli-framework
com.sun.phobos:phobos-rhino
com.sun.phobos:phobos-debugger-api
org.glassfish.common:common-util
com.sun.commons:logging-api
javax.javaee:javaee
com.sun.commons:modeler
com.sun.commons:digester
com.sun.commons:beanutils
com.sun.commons:collections
org.apache.ant:ant
org.glassfish.common:annotation-framework
org.jruby:jruby-complete
org.glassfish.core:glassfish
org.glassfish.common:glassfish-api
javax.enterprise.deploy:deployment-api
javax.persistence:persistence-api
org.glassfish.api:mail
javax.jms:jms-api
javax.security:security-api
javax.resource:connector-api
javax.annotation:jsr250-api
javax.jws:jsr181-api
javax.xml.ws:jaxws-api
javax.servlet.jsp:jsp-api
javax.ejb:ejb-api
org.apache.ant:ant-launcher
xerces:xercesImpl
provided
org.glassfish.admin:config-api
asm:asm-commons
asm:asm-util
asm:asm-analysis
org.jruby.extras:jna
ant:ant
bsf:bsf
jline:jline
backport-util-concurrent:backport-util-concurrent
org.glassfish.external:grizzly-module
com.sun.enterprise:hk2
javax.servlet:servlet-api
javax.xml.bind:jaxb-api
javax.xml:jaxrpc-api
javax.transaction:transaction-api
xml-apis:xml-apis
xml-resolver:xml-resolver
asm:asm-tree
junit:junit
com.sun.grizzly:grizzly-http
com.sun.enterprise:config
javax.activation:activation
javax.xml.bind:jsr173_api
asm:asm
com.sun.grizzly:portunif
com.sun.grizzly:grizzly-rcm
com.sun.grizzly:grizzly-http-utils
com.sun.xml.stream:sjsxp
org.jvnet:tiger-types
com.sun.enterprise:auto-depends
com.sun.grizzly:grizzly-framework
javax.xml.stream:stax-api

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