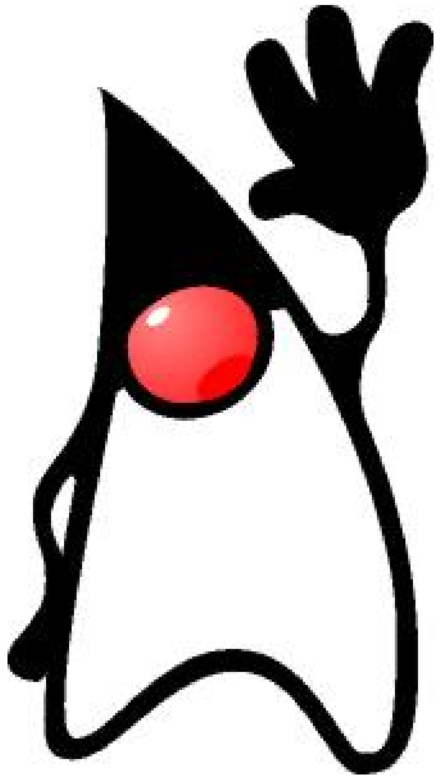


The Java Packaging Nightmare

St. Augustin, FrOSCon 2010

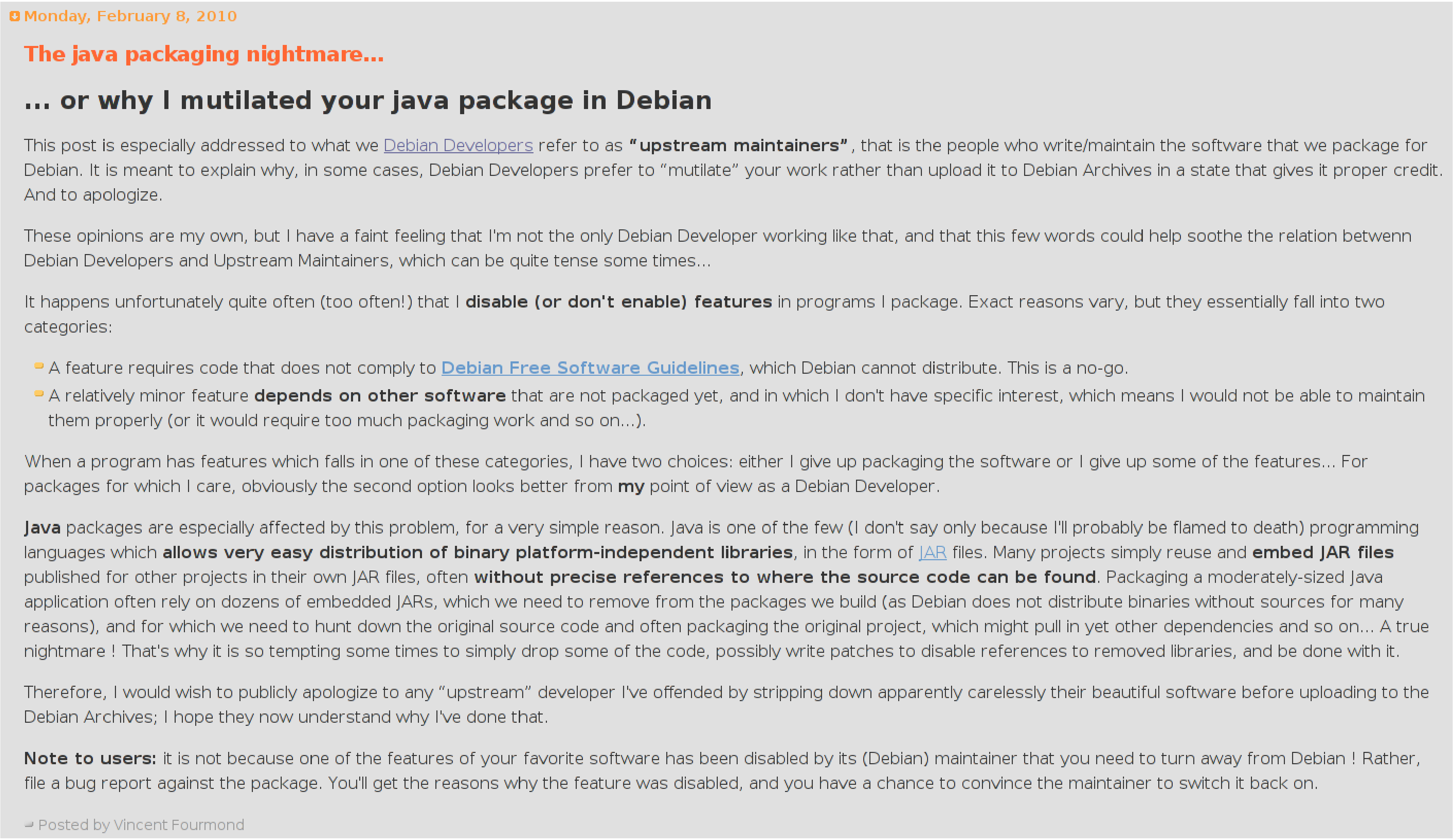
Topics

- Motivation for the Talk
- History of Free Java
- Current State of Java in Debian
- Problems in Upstream's Packaging and Solutions
- Packaging with Apache Maven
- Questions and Answers

  BUT

- This is not a talk about details of the programming language Java.

Blog Post by Vincent Fourmond



Monday, February 8, 2010

The java packaging nightmare...

... or why I mutilated your java package in Debian

This post is especially addressed to what we Debian Developers refer to as **"upstream maintainers"**, that is the people who write/maintain the software that we package for Debian. It is meant to explain why, in some cases, Debian Developers prefer to "mutilate" your work rather than upload it to Debian Archives in a state that gives it proper credit. And to apologize.

These opinions are my own, but I have a faint feeling that I'm not the only Debian Developer working like that, and that this few words could help soothe the relation betwenn Debian Developers and Upstream Maintainers, which can be quite tense some times...

It happens unfortunately quite often (too often!) that I **disable (or don't enable) features** in programs I package. Exact reasons vary, but they essentially fall into two categories:

- A feature requires code that does not comply to Debian Free Software Guidelines, which Debian cannot distribute. This is a no-go.
- A relatively minor feature **depends on other software** that are not packaged yet, and in which I don't have specific interest, which means I would not be able to maintain them properly (or it would require too much packaging work and so on...).

When a program has features which falls in one of these categories, I have two choices: either I give up packaging the software or I give up some of the features... For packages for which I care, obviously the second option looks better from **my** point of view as a Debian Developer.

Java packages are especially affected by this problem, for a very simple reason. Java is one of the few (I don't say only because I'll probably be flamed to death) programming languages which **allows very easy distribution of binary platform-independent libraries**, in the form of JAR files. Many projects simply reuse and **embed JAR files** published for other projects in their own JAR files, often **without precise references to where the source code can be found**. Packaging a moderately-sized Java application often rely on dozens of embedded JARs, which we need to remove from the packages we build (as Debian does not distribute binaries without sources for many reasons), and for which we need to hunt down the original source code and often packaging the original project, which might pull in yet other dependencies and so on... A true nightmare ! That's why it is so tempting some times to simply drop some of the code, possibly write patches to disable references to removed libraries, and be done with it.

Therefore, I would wish to publicly apologize to any "upstream" developer I've offended by stripping down apparently carelessly their beautiful software before uploading to the Debian Archives; I hope they now understand why I've done that.

Note to users: it is not because one of the features of your favorite software has been disabled by its (Debian) maintainer that you need to turn away from Debian ! Rather, file a bug report against the package. You'll get the reasons why the feature was disabled, and you have a chance to convince the maintainer to switch it back on.

Posted by Vincent Fourmond

(emphasis added)

History of Free Java

- Sun's JDK was non-free for a long time (since 1995)
- A lot of free software has been written in Java like Tomcat (1999), JBoss, Eclipse (2001), Hibernate
- Several alternative JDKs, JREs, VMs like kaffe, gcj (1999), classpath, ecj got better and better over time but never caught up with Sun's JDK
- Relicensing of Sun's JDK under GPL as OpenJDK (2006)
- Additional patches from IcedTea project (2007)
- Free licensed JEE core libraries through Sun's Glassfish relicensing and Apache Geronimo (JavaMail, JAF, JMS, JPA)

Current State in Debian

- 525 source packages maintained by the Java packaging team
- toolchain is good shape at least on major architectures
- some popular packages: Tomcat, Eclipse, Jajuk (jukebox), and libraries like Hibernate (ORM)
- no JEE application server, but some core parts of GlassFish, JBoss, and Geronimo
- no major Webapps like Liferay
- good cooperation with Ubuntu but not with other Linux distros
- almost no interest by upstream developers

Toolchain and Runtime

- OpenJDK with IcedTea patches is default-jdk on most Linux architectures with the following JVMs
  - Sun's Hotspot VM where port is available,
  - Zero and Shark VM
  - IcedTea Cacao VM
- GCJ-JDK on FreeBSD, Linux-HPPA, and Hurd
- IcedTea browser applet
- Sun's non-free JDK: available but not used for packaging
- Packaging tools:
  - java-helper, Ant, Maven, and some „home made“
  - CDBS, maven-XXX-helper, eclipse-helper

Debian Free Software Guidelines (DFSG)

1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of The Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of License
8. License Must Not Be Specific to Debian
9. License Must Not Contaminate Other Software
10. Example Licenses (GPL, BSD, and Artistic)

Problems in Upstream's Packaging

- Licenses of source code incorrectly documented
- Inclusion or download of binary only JAR files
    - No source, no download or VCS address
    - Often no license and copyright information
    - Sometimes non-free
- Missing Modularization
- Official Maven repository
    - No notion of a source package
    - Source JARs without build system
    - Unusable dependency handling
- Code Duplication

License Failures

- JBoss Web:
  LGPL-3 in the LICENSE file but GPL|CDDL and Apache licenses in the source code (mainly from Apache Tomcat)
- JRuby:
  release 1.5.1 ships a binary file constantine.jar where no source code could be found anywhere (but it is fixed now)
- Hibernate:
  Java Persistence 2.0 API cannot be redistributed („No license is granted for... distributing the Specification to third parties.“)

Missing Modularization

- Easy in the good old C/C++ world
- Autoconf, Automake, pkg-config:
  - ./configure --enable-feature-a --disable-feature-b
  - pkg-config --libs --cflags library-name
- CMake, pkg-config:
  - cmake -DUSE_FEATURE:BOOL=ON
- Almost nonexistent in the Java world
- Maven profiles would be an option:
  - currently not very popular
  - can configure modules and plugins
  - can be activated by properties
- Source patching is “best practice” in Debian

Configuring Modules in Hibernate

```
<profile>
    <id>jdk6-modules</id>
    <activation>
        <property>
            <name>disableJDK6Modules</name>
            <value>!true</value>
        </property>
    </activation>
    <modules>
        <module>entitymanager</module>
        <module>jdbc4-testing</module>
    </modules>
</profile>
```

Configuring Plugins

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <excludes>
      <exclude>**/HibernateValidator.java</exclude>
    </excludes>
  </configuration>
</plugin>
```

combined with profiles and activation by properties
but a preprocessor like cpp does not exist

Meta Information in Source

- every source package should have enough meta data: e.g. 1 pom.xml as a „source POM“ or just plain text
- important meta data:
  Name <name>
  Short Description <description>
  Homepage <url>
  VCS with guest access <scm>
  Issue Tracker <issueManagement>
  Copyright Holder <!-- ... -->
  License(s) <licenses>
- the source POM should be the parent POM (<parent>) for all other POMs in the same source package
- optionally it might be mentioned in a XML comment <!-- source POM is ... -->
- Check your direct dependencies, too!

Code Duplication

- please read http://fedoraproject.org/wiki/Packaging:No_Bundled_Libraries
- There 4 ways for embedding code:
  - plain copies of foreign source code
  - modifies copies of foreign source code
  - embedding contents of other JAR files (JARs are just ZIP files)
  - embedding modified contents of other JAR files through tools like jarjar that rewrite Java packages (class hierarchies), e.g. org.objectweb.asm.** → jruby.objectweb.asm.@1

Apache Maven Packaging in Debian

- Maven is quite new in Debian but many important artifacts are shipping POM files now
- Ant is still the mostly used packaging tool in combination with cdbs, javahelper, or dh
- using Maven's offline mode during build
- Package based POM repository below /usr/share/maven-repo/
- POM preprocessing and installation according to Debian's “Maven Repository Specification”
- maven-repo-helper
- maven-ant-helper
- maven-debian-helper

Questions and Answers

- Thank you for your interest!
- Join us at http://pkg-java.alioth.debian.org/, debian-java@lists.debian.org, and #debian-java!