Java Packaging for Developers

(Get someone to package your Java application)

Red Hat

Author: *Stanislav Ochotnický*

Date: *5th February 2011*

Abstract

Basic guidelines for developers to make it easier for their applications to get into Linux distributions. Handling source releases, build systems and dependencies and making packagers happy in the process.

Main audience: Java developers with little packaging experience.

Copyright © 2011 Author: *Stanislav Ochotnický, Red Hat.*
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

fedora

Agenda

1. **Overview**
2. **General Tips**
   - Releases and dependencies
   - Version numbers and microrevisions
3. **Buildsystem specific**
   - Ant
   - Maven

fedora

Overview of problems

Linux packaging rules

- Build everything from source
- No bundled dependencies
- Keep compatibility in released distributions
- Keep FHS compliance if possible

Java style

- Binary-only releases
- Bundling deps even in source releases
- No notion of binary compatibility
- One directory contains whole application

There are reasons Linux distributions have strict rules

- Bundling causes security problems
- Building from source ensures ability to fix bugs quickly

fedora

Releases and dependencies

- Don't add another xml parser dep just because you can
- Try to pick dependencies from major projects
- Make complete source releases with build scripts

Results of no source releases

```
# latest release doesn't generate javadocs and there is no source
# tarball with pom.xml or ant build file
#
# svn export -r86 http://atinject.googlecode.com/svn/trunk atinject-1
# tar caf atinject-1.tar.xz atinject-1
```

fedora

Using maven-assembly-plugin to release source

```
<build>
<plugins>
  ...
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
      <configuration>
        <descriptorRefs>
          <descriptorRef>project</descriptorRef>
        </descriptorRefs>
      </configuration>
      <executions>
        <execution>
          <id>make-assembly</id>
          <phase>package</phase>
          <goals> <goal>single</goal>  </goals>
        </execution>
      </executions>
    </plugin>
   ...
</plugins>
</build>
```

fedora

Z, oh where are thou?

What are micro revisions?

- Z in X.Y.Z version string
- Micro revisions are smallest released changes for projects
- Usually contain only bugfixes
- ALWAYS backward compatible

Java ecosystem understanding of Z in X.Y.Z

- Anything goes (for a LOT of projects)
- Binary compatibility does not matter in Java [a]

[a] It's actually more important because it's impossible to figure out if the dependant library is compatible. That is because missing/changed methods will show up only during runtime!

fedora

Z, oh where are thou? (cont.)

Example Maven update from 3.0 to 3.0.1

- Required Aether update from 1.7 to 1.8
- Aether added new dependency on async-http-client
- Async-http-client had netty, jetty 7.x and other new deps
- Fortunately only netty was runtime dependency

How to do it?

- Z updates contain only backward-compatible bugfixes
- No changes to dependencies
- Check required dependency update for new dependencies

fedora

How to use Ant properly?

Short answer?

DON'T

Long answer

- Try apache-ivy instead of bundling dependencies
- Use properties to reuse definitions
- Place all dependencies into one directory (preferably lib)
- Use name-version.jar for dependencies
- Don't be overly smart when writing build.xml
- Worst case: maven-antrun-plugin

fedora

Advantages of Maven over Ant

For developers

- Declarative instead of descriptive
- Project metadata information in one place
- Good integration with other tools
- Support for running Ant for parts of build (if needed)

For packagers

- Declarative instead of descriptive
- Clear dependencies including their versions
- No bundling of dependencies
- Problems are the same in all projects

fedora

Maven woes

It's not easy to make a mess, but...

- Sometimes too easy to add new dependency
- Use maven-dependency-plugin:copy-dependencies carefully
- Use maven-bundle-plugin carefully
  - Useful for metadata manipulations
  - Export-Package can include dependencies in resulting jar
  - This is practically static linking
- Please don't use maven-shade-plugin
  - Relocates package classes into different package
  - Hidden static linking
  - Impossible to reveal just from contents of jar

fedora

maven-shade-plugin in action

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <configuration>
    <relocations>
      <relocation>
        <pattern>org.objectweb.asm</pattern>
        <shadedPattern>org.packager</shadedPattern>
      </relocation>
    </relocations>
  </configuration>
</plugin>
```

fedora

maven-shade-plugin in action (cont.)

Contents of resulting jar

```
META-INF/
META-INF/MANIFEST.MF
META-INF/maven/
META-INF/maven/org.packager/
META-INF/maven/org.packager/Pack/
META-INF/maven/org.packager/Pack/pom.properties
META-INF/maven/org.packager/Pack/pom.xml
org/
org/packager/
org/packager/signature/
org/packager/signature/SignatureReader.class
org/packager/signature/SignatureVisitor.class
org/packager/signature/SignatureWriter.class
org/packager/Pack.class
```

Can you tell where is signature sub-package coming from?

fedora

maven-bundle-plugin in action (cont.)

```
<plugin>
  <groupId>org.apache.felix</groupId>
  <artifactId>maven-bundle-plugin</artifactId>
  <extensions>true</extensions>
  <configuration>
    <instructions>
      <Export-Package>org.objectweb.asm.signature</Export-Package>
    </instructions>
  </configuration>
</plugin>
```

fedora

maven-bundle-plugin in action (cont.)

Contents of resulting jar

```
META-INF/MANIFEST.MF
META-INF/
META-INF/maven/
META-INF/maven/org.packager/
META-INF/maven/org.packager/Pack/
META-INF/maven/org.packager/Pack/pom.properties
META-INF/maven/org.packager/Pack/pom.xml
org/
org/objectweb/
org/objectweb/asm/
org/objectweb/asm/signature/
org/objectweb/asm/signature/SignatureReader.class
org/objectweb/asm/signature/SignatureVisitor.class
org/objectweb/asm/signature/SignatureWriter.class
org/packager/
org/packager/Pack.class
```

fedora

Summary

Please

- Provide source releases with build scripts
- Do not use niche libraries for dependencies
- Give us stable, bugfix-only, no-new-deps micro releases
- If you want ant, try apache-ivy and don't be too smart
- Careful with certain maven plugins

The end.

Thanks for listening.