

Packaging with Javahelper

Matthew Johnson mjj29@debian.org

Niels Thykier niels@thykier.net

2nd August 2010

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Upstream releases

- ▶ Lots of jar or zip releases not tars
- ▶ Releases include 3rd party libraries
- ▶ Releases include build products (javadoc, class files, etc)
- ▶ Repacking needs to be done with every new release
- ▶ Solution: `jh_repack`

Upstream releases

jh_repack watchfile

```
version=3
```

```
http://www.matthew.ath.cx/projects/salliere/ \  
  (?:.*/)?salliere-?_?([\d+\.]+\d+)\.(jar|zip) \  
  debian jh_repack
```

jh_repack manual

```
jh_repack --upstream-version <version> <archive>
```

Building upstream

- ▶ Build system expects 3rd party libraries in the build tree
- ▶ Build system broken
- ▶ Build system non-existent
- ▶ Build system doesn't build javadoc

- ▶ Solution: `jh_linkjars`
- ▶ Solution: `jh_build`

Building upstream

jh_linkjars (old-style)

```
jh_linkjars libs
```

jh_linkjars (debian/linkjars)

```
libs
```

Building upstream

jh_build (old-style)

```
JAVA_HOME=/usr/lib/jvm/default-java  
CLASSPATH=/usr/share/java/esd.jar : ...  
jh_build weirdx.jar src
```

jh_build (dh7 - debian/javabuild)

```
salliere.jar src
```

jh_build (dh7 - debian/rules)

```
export JAVA_HOME=/usr/lib/jvm/default-java  
export CLASSPATH=/usr/share/java/csv.jar : ...
```

jh_build (cdbs - debian/rules)

```
export JAVA_HOME=/usr/lib/jvm/default-java  
export CLASSPATH=/usr/share/java/csv.jar : ...  
JH_BUILD_JAR=salliere.jar  
JH_BUILD_SRC=src
```

Fixing upstream's build products

- ▶ Upstreams often fail to set manifest entries
- ▶ Debian-specific manifest entries may be needed
- ▶ Solution: `jh_classpath`
- ▶ Solution: `jh_manifest`

Fixing upstream's build products

jh_classpath (debian/classpath)

```
src/my.jar /usr/share/java/dep1.jar /usr/share/...  
src/my-other.jar /usr/share/java/dep1.jar /...
```

jh_manifest (debian/manifest)

```
usr/share/weirdx/weirdx.jar :  
  Main-Class: com.jcraft.weirdx.WeirdX  
  Debian-Java-Home: /usr/lib/jvm/default-java
```

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Installation

- ▶ There are specific requirements for installation of jars and javadoc in policy
- ▶ Code duplication to support those requirements is badd
- ▶ Being able to make a single source change then rebuild to change the requirements is good
- ▶ Solution: `jh_installlibs`
- ▶ Solution: `jh_installjavadoc`

Installation

`jh_installlibs (debian/jlibs)`

`foo.jar`

`build/bar.jar`

`jh_installjavadoc (debian/javadoc)`

`doc/api`

`internal`

`api /usr/share/doc/libfoo-java/api`

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Runtime support

- ▶ Every java program needs a wrapper script
- ▶ Looking up the path to libjvm.so is hard
- ▶ Solution: jarwrapper
- ▶ Solution: jh_exec
- ▶ Solution: java-vars.sh

Runtime support

- ▶ Every java program needs a wrapper script
- ▶ Looking up the path to libjvm.so is hard
- ▶ Solution: jarwrapper
- ▶ Solution: jh_exec
- ▶ Solution: java-vars.sh

Dependencies

- ▶ Autogenerating Depends: lines (a-la dh_shlibdeps) is helpful
- ▶ Policy specifies a variety of alternate depends on JREs
- ▶ jarwrapper is needed in the Depends for executable jars
- ▶ A single place that generates dependencies makes it easier to make policy changes

- ▶ Solution: jh_depends

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Eclipse - PDE based build

- ▶ PDE-based build (auto-generates ant-build files)
- ▶ Clean does not clean properly.
- ▶ Solution: `jh_setupenvironment`

Eclipse - External (Orbit) dependencies

- ▶ Eclipse plug-ins uses non eclipse libraries.
- ▶ Requires OSGi-metadata to be loaded (unless embedded)
- ▶ Lazy upstreams embed the jar and use Bundle-Classpath
- ▶ Eclipse needs bundles named by their Symbolic Name.
- ▶ Eclipse fetches the bundles from Orbit.
- ▶ Setting up Orbit is a tedious task.

- ▶ Require-Bundle + OSGi-metadata in the Orbit dependency.
- ▶ OSGi-metadata must be added manually, but is generally trivial to make.
- ▶ For everything else there is `jh_generateorbitdir`.

Eclipse - Building features

- ▶ Feature is a set of related plug-ins.
- ▶ Java sources are scattered neatly into separate plug-ins.
- ▶ Manifest files lists dependencies.
- ▶ The PDE handles all of this, but ...
- ▶ It takes over 10 arguments to make PDE build one “trivial” feature.
- ▶ LinuxTools/eclipse-build provides a wrapper, but it is still a pain.
- ▶ Solution: `jh_compilefeatures`

Eclipse - Install features

- ▶ Features are compiled into zip files.
- ▶ Must be unzipped and located so that eclipse finds it.
- ▶ The zip file contains a full copy of Orbit Dependencies.
- ▶ Solution: `jh_installeclipse`

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

dh7

debian/rules

```
#!/usr/bin/make -f
```

```
export JAVA_HOME=/usr/lib/jvm/default-java
```

```
export CLASSPATH=/usr/share/java/csv.jar:...
```

```
%:
```

```
dh $@ --with javahelper
```

debian/rules

```
#!/usr/bin/make -f
export JAVA_HOME=/usr/lib/jvm/default-java
export CLASSPATH=/usr/share/java/csv.jar:...
JH_BUILD_JAR=salliere.jar
JH_BUILD_SRC=src

include /usr/share/cdbs/1/class/javahelper.mk
```

From here to there in 60 seconds

jh_makepkg can be used to generate most of a debian dir for a simple Java application or a simple library. It will prompt for a few answers it can't work out and infer some others.

Outline

Upstream

- Upstream releases

- Upstream build systems

Installing

Binary packages

- Runtime support

- Dependencies

Eclipse

Packaging frameworks

- Package autogeneration

Questions

Questions?

- ▶ <http://packages.debian.org/javahelper>
- ▶ </usr/share/doc/javahelper>
- ▶ <http://pkg-java.alioth.debian.org/>
- ▶ <http://pkg-java.alioth.debian.org/examples>
- ▶ debian-java@lists.debian.org