Eclipse Artifacts on Maven Central

Stephan Herrmann
Preface

GK Software SE

- Adopted Modeling in 2007 based on Xtext 0.7
- Generators to be invoked using Maven
- Fornax maven plugin was voodoo
- Dependency management cargo-cult:
  - manually upload tons of jars to Nexus
  - manually add tons of artifacts to maven plugin dependencies
- Builds were frequently broken by changes beyond our control
- DSLs were used more and more widely
- Upgrading to newer versions was close to impossible
GK Software SE

- Adopted Modeling in 2007 based on Xtext 0.7
- Generators to be invoked using Maven
- Fornax maven plugin was voodoo
- Dependency management:
  - manually upload tons of jars to Nexus
  - manually add tons of artifacts to maven plugin dependencies
- Builds were frequently broken by changes beyond our control
- DSLs were used more and more widely
- Upgrading to newer versions was close to impossible
A Plea against Xenophobia
This is a Releng talk

- Do you want your stuff to be *consumed*?

- No Java code!

- But this is the cool part!?
Comfort Zones

- PDE/Build, Buckminster, tycho
- OSGi bundles
- Java Coding
- modules
- export / require
- real stuff
- provisioning metadata
- building <yuck>
Comfort Zones

maven

metadata
provisioning
building

...
Welcome to the Real World

› Stuff may / may not be *intended* for use via Maven

› Stuff *is used* via Maven

› It's all about jar files
  › Some produce them
  › Others consume them

› That's all?
Issues on the Road

1. Extent
2. Coordinates
   › names
   › versions
3. Dependencies
   › ranges
   › optional, circular, transitive ...
4. More metadata
5. Javadoc & sources
6. Sign, Seal & Deliver

Publishing to Maven Central:
   Eclipse Platform
   Eclipse JDT
   Eclipse PDE
1. Extent

› Grab any p2 repository

› Discard irrelevant stuff, e.g.:
  • features (no representation in Maven)
  • tests (?)
  • gc (stuff that was needed for removed stuff)
2. Coordinates

› Names
  › artifactId: bundle symbolic name
  › groupId?
    • approx. 1st 3 segments:
      - org.eclipse.jdt
      - org.eclipse.pde
      - org.eclipse.platform (artificial name)

› Versions
2. Coordinates

› Names
  › artifactId: bundle symbolic name
  › groupId?
    • approx. 1st 3 segments:
      - org.eclipse.jdt
      - org.eclipse.pde
      - org.eclipse.platform

› Versions
2. Coordinates

› Names
  › artifactId: bundle symbolic name
  › groupId?

Projects must own their names incl. groupId

- org.eclipse.platform

› Versions
2. Coordinates

› Names

› Versions

› syntax
  • OSGi: `<major>.<minor>.<service>.<qualifier>`
  • Maven: anything is accepted
    – some formats have specific meaning
    – others are just strings

› matching
# 2. Coordinates

- **Names**

- **Versions**
  - **syntax**
  - **matching**
    - real world-ish example:

```
needed:
g:org.eclipse.core
a:org.eclipse.core.runtime
v:1.2.3
```

resolved to
```
g:org.eclipse.core
a:org.eclipse.core.runtime
v:1.2.3.v20140815
```
2. Coordinates

› Names

› Versions
  › syntax
  › matching
    • real world-ish example:

needed:
g:org.eclipse.core
a:org.eclipse.core.runtime
v:1.2.3

not resolvable

\{
  g:org.eclipse.core
  a:org.eclipse.core.runtime
  v:1.2.3v20140815
\}

\{
  g:org.eclipse.core
  a:org.eclipse.core.runtime
  v:1.2.3v20140815
\}
2. Coordinates

› Names

› Versions

› syntax – be nice to maven:
  • 3 – part versions for releases
  • some 3rd party artifacts omit service .0
  • future: snapshots?

› matching
2. Coordinates

› Names

› Versions
  › syntax
  › matching

Never publish versions with unclear semantics.
3. Dependencies

› Ranges?
   › re-usable library uses ranges (as we do in Eclipse)
   › application freezes all versions
     • Eclipse: target platform
     • Maven: dependencies pom, required with scope “import”

```xml
<dependency>
  <groupId>com.mycomp</groupId>
  <artifactId>myplatform</artifactId>
  <version>${platform.version}</version>
  <scope>import</scope>
</dependency>
```
3. Dependencies

 › Ranges?

 › Capabilities & Filters
   › Supported by OSGi & P2
   › Cannot be modeled in Maven
   › Example: platform specific fragments
3. Dependencies

› Ranges?

› Capabilities & Filters
  › Supported by OSGi & P2
  › Cannot be modeled in
  › Example: platform specific fragments
3. Dependencies – fragments

› Bundle

Bundle-SymbolicName: org.eclipse.swt
Export-Package:
  org.eclipse.swt,
  org.eclipse.swt.accessibility,
  org.eclipse.swt.awt,
  ...

Eclipse-ExtensibleAPI: true

› Fragment(s)

Bundle-SymbolicName: org.eclipse.swt.gtk.linux.x86_64
Fragment-Host: org.eclipse.swt
Eclipse-PlatformFilter: (& (osgi.ws=gtk) (osgi.os=linux) (osgi.arch=x86_64))
Export-Package:
  org.eclipse.swt,
3. Dependencies – fragments

› Bundle (p2 metadata)

```xml
<unit id='org.eclipse.swt' version='3.108.0...'>
<provides size='20'>
  <provided namespace='java.package' name='org.eclipse.swt'/> ...
<requires size='8'>
  <required ... name='org.eclipse.swt.gtk.linux.x86_64' ...>
    <filter>
      (&amp;(osgi.arch=x86_64)(osgi.os=linux)(osgi.ws=gtk)...) ...
  </filter>
</requires>
</unit>
```

› Fragment(s) (p2 metadata)

```xml
<unit id='org.eclipse.swt.gtk.linux.x86_64' version='3.108.0...'>
<provides size='25'>
  <provided namespace='java.package' name='org.eclipse.swt'/> ...
<requires size='1'>
  <required namespace='osgi.bundle' name='org.eclipse.swt'/> ...
<filter>
  (&amp;(osgi.arch=x86_64)(osgi.os=linux)(osgi.ws=gtk)) ...
</filter>
</unit>
```
3. Dependencies – fragments

Bundle (p2 metadata)

```
<unit id='org.eclipse.swt' version='3.108.0...' />
<provides size='20'>
  <provided namespace='java.package' name='org.eclipse.swt'/> ...
<requires size='8'>
  <required namespace='osgi.bundle' name='org.eclipse.swt'/> ...
  <filter>
    (&(osgi.arch=x86_64)(osgi.os=linux)(osgi.ws=gtk)) ...
</filter>
</unit>

<unit id='org.eclipse.swt.gtk.linux.x86_64' version='3.108.0...' />
<provides size='25'>
  <provided namespace='java.package' name='org.eclipse.swt'/> ...
<requires size='1'>
  <required namespace='osgi.bundle' name='org.eclipse.swt'/> ...
  <filter>
    (&(osgi.arch=x86_64)(osgi.os=linux)(osgi.ws=gtk)) ...
  </filter>
</unit>
```

This is not supported in Maven
3. Dependencies – fragments

› Bundle (pom)

```xml
<artifactId>org.eclipse.swt</artifactId>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <artifactId>org.eclipse.swt.${osgi.platform}</artifactId>
  </dependency>
</dependencies>
```

› Fragment(s) (pom)

```xml
<artifactId>org.eclipse.swt.gtk.linux.x86_64</artifactId>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <artifactId>org.eclipse.swt</artifactId>
  </dependency>
</dependencies>
```
3. Dependencies – fragments

› Bundle (pom)

```xml
<id>org.eclipse.swt</id>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <id>org.eclipse.swt.${osgi.platform}</id>
  </dependency>
</dependencies>
```

› Fragment(s) (pom)

```xml
<id>org.eclipse.swt.gtk.linux.x86_64</id>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <id>org.eclipse.swt</id>
  </dependency>
</dependencies>
```

› Usage single platform

MyApplication dependency
3. Dependencies – fragments

› Bundle (pom)

```xml
<groupId>org.eclipse.swt</groupId>
<artifactId>org.eclipse.swt</artifactId>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <groupId>org.eclipse.swt</groupId>
    <artifactId>org.eclipse.swt</artifactId>
    <version>3.108.0...</version>
  </dependency>
</dependencies>
```

› Fragment(s) (pom)

```xml
<groupId>org.eclipse.swt</groupId>
<artifactId>org.eclipse.swt</artifactId>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <groupId>org.eclipse.swt</groupId>
    <artifactId>org.eclipse.swt</artifactId>
  </dependency>
</dependencies>
```

› Usage single platform

```xml
<exclusion>
  <groupId>org.eclipse.swt</groupId>
  <artifactId>org.eclipse.swt</artifactId>
</exclusion>
```

Stephan Herrmann @EclipseCon Europe 2018 – published under the EPL
3. Dependencies – fragments

› Bundle (pom)

```xml
<id>org.eclipse.swt</id>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <id>org.eclipse.swt.${osgi.platform}</id>
  </dependency>
</dependencies>
```

› Fragment(s) (pom)

```xml
<id>org.eclipse.swt.gtk.linux.x86_64</id>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <id>org.eclipse.swt</id>
  </dependency>
</dependencies>
```

› Usage multi platform (recommended)

```
$ mvn -Dosgi.platform=gtk.linux.x86_64 ...
```

Stephan Herrmann @EclipseCon Europe 2018 – published under the EPL
3. Dependencies – fragments

› Bundle (pom)

```xml
<id>org.eclipse.swt</id>
<version>3.108.0...</version>
<dependencies>
  <dependency>
    <id>org.eclipse.swt.${osgi.platform}</id>
  </dependency>
</dependencies>
```

› Usage multi platform (recommended)

Usage multi platform (recommended)

MyApplication

```
$ mvn ...
```

.or:

```
mvn
```

mvn folder:

Located with in the projects top level folder, the files maven.config and extensions.xml contain project specific configuration for running Maven.

```
<dependencies>
  <dependency>
    <id>org.eclipse.swt</id>
  </dependency>
```

Stephan Herrmann @EclipseCon Europe 2018 – published under the EPL
4. More Metadata

› Required (by OSSRH et al.):
  › constant content
    • copyright
    • license
    • organization
    • issue management
  › per-artifact content (retrieved mostly from MANIFEST.MF):
    • name (typically matches what the aggregator filled in as description)
    • url
    • scm connection, tag and url
  › semi constant content
    • developers ("Who's Involved" page of corresponding project in PMI)
More Metadata

Required (by OSSRH et al.):

- constant content
  - copyright
  - license
  - organization
  - issue management
- per-artifact content
  - name (typically matches what the aggregator filled in as description)
  - url
- semi constant content
  - developers ("Who’s Involved" page of corresponding project in PMI)

XML example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Copyright (c) 2016, 2018 GK Software SE and others. -->
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.eclipse.jdt</groupId>
  <artifactId>org.eclipse.jdt.core</artifactId>
  <version>3.15.0</version>
  <description>Java Development Tools Core</description>
  <name>Java Development Tools Core</name>
  <url>http://www.eclipse.org/jdt</url>
  <licenses>
    <license>
      <name>Eclipse Public License</name>
      <distribution>repo</distribution>
    </license>
  </licenses>
  <organization>
    <name>Eclipse Foundation</name>
    <url>http://www.eclipse.org/</url>
  </organization>
  <issueManagement>
    <system>Bugzilla</system>
    <url>https://bugs.eclipse.org/</url>
  </issueManagement>
  <scm>
    <connection>scm:git:git://git.eclipse.org/gitroot/jdt/eclipse.jdt.core.git...</connection>
    <tag>I20180905-0800</tag>
    <url>https://git.eclipse.org/c/jdt/eclipse.jdt.core.git</url>
  </scm>
  <developers>
    <developer>
      <url>https://projects.eclipse.org/projects/eclipse.jdt/who</url>
    </developer>
  </developers>
</project>
```
5. More Artifacts

Required (by OSSRH et al.):

› javadoc
  • we have .doc bundles
    – no 1:1 correspondence
  • create .javadoc jar on the fly:
    – create README.txt
    – refer to the corresponding doc bundle

› sources
  • most projects correctly build .sources
    – last remaining offender: org.eclipse.core.net.*
    – source bundle created on the fly (ad-hoc)
6. Sign, Seal & Deliver

› Sign & Upload

› maven-gpg-plugin
  • keys howto: ask webmaster :)
  • goal: gpg:sign-and-deploy-file

› Release

› Via Web interface of OSSRH Nexus
  • “close” => runs validations
  • “release” & “drop”
Tools Used (status quo)

1. Extent (rm & gc)
   - bash script

2. Coordinates
   - names (groupId)
   - versions (format)
   - CBI Aggregator

3. Dependencies
   - ranges (fragments)
   - optional, circular, transitive ...

4. More metadata
   - Java (< 500 LOC)

5. Javadoc & sources
   - bash script

6. Sign, Seal & Deliver
   - mvn
Tools Used (perspective)

1. Extent (rm & gc)
2. Coordinates
   - names (groupId)
   - versions (format) (snapshots)
3. Dependencies
   - ranges (fragments)
   - optional, circular, transitive ...
4. More metadata
5. Javadoc & sources
6. Sign, Seal & Deliver

Explicitly select roots?
Modelled in .aggr

(bash script – doc stub)
mvn

CBI Aggregator