Building and Sharing Your P2 Fortress with Tycho and Nexus

Trent Vigar
Software Engineer

March 10 2015
What will I be talking about?

• Who am I and what do I do?
• Why Tycho? What does it do for us?
• Build setup and how we hit the nexus repos
• How did I improve our system and what were the issues?
• Takeaways and considerations
Who are we and what do we do?

- Healthcare IT founded in 1979
- WHQ located in Kansas City, Missouri (where I work)
- Over 20,000 associates in more than 30 countries
- Development group of 3000+
Meet the team - iAware

- Java, Eclipse RCP, OSGi, SWT
- Platform dev team and solution dev team
- Upgraded platform to eclipse 4 about 3 years ago
Why Tycho?

• New provisioning features

• Enhance our build process

• We want : P2 artifacts
  • Tycho gives us that!

• We have : Many Eclipse plug-ins & Maven
  • Tycho uses that to give us what we want!
How exactly are we using Tycho?

• Pom-first dependency resolution
  • As opposed to manifest-first

• Not building eclipse-plugins
  • Lots of eclipse-features
  • one eclipse-repository for each feature-set
Everybody’s use case is different, this just works best for us
ARE YOU SURE

YOU'RE READY?
How do we get started?

- Maven project
- Feature folders
- eclipse-repository folder
- parent-pom folder
What goes in the feature folder?

- feature.xml
- build.properties
  - Haven’t messed with this yet
- pom.xml
<requires>
  <import feature="iaware.solution.cc.contentset.core" version="7.0.0"/>
  <import feature="iaware.solution.mylist.core" version="6.3.0"/>
  <import feature="iaware.solution.p2da.core" version="7.0.0"/>
  <import feature="iaware.solution.patient.combine.ui" version="6.3.0"/>
  <import feature="iaware.solution.infusions.mdbus" version="7.1.1.qualifier"/>
  <import feature="iaware.platform.base.impl" version="6.4.1"/>
  <import feature="iaware.platform.core" version="6.4.1"/>
</requires>

<plugin
  id="com.cerner.genesis.app.client.nl"
  download-size="0"
  install-size="0"
  version="0.0.0"
  unpack="false"/>
What goes in the eclipse-repository folder?

- category.xml
- pom.xml
<repository>
  <id>infusions.mdbus</id>
  <url>file:${project.basedir}/../iaware.solution.infusions.mdbus/target/site</url>
  <layout>p2</layout>
</repository>

<plugin>
  <groupId>org.eclipse.tycho</groupId>
  <artifactId>tycho-p2-director-plugin</artifactId>
  <version>${tycho-version}</version>
  <configuration>
    <includeAllDependencies>false</includeAllDependencies>
  </configuration>
</plugin>

<plugin>
  <groupId>org.eclipse.tycho</groupId>
  <artifactId>tycho-p2-repository-plugin</artifactId>
  <version>${tycho-version}</version>
  <configuration>
  </configuration>
</plugin>
What goes in the parent folder?

Parent pom!

• We have:
  • Tycho-maven-plugin : packaging types
  • Tycho-packaging-plugin : P2 repo in target
  • Target-platform-configuration
    • pomDependencies = consider
  • Maven-deploy-plugin
  • Repos for retrieving our plugins
  • P2 repos for retrieving the required features

• Bottom Line : put in what works best for you
Where am I at now?
How do I build this and what do I get?

- mvn clean package
  - maven-invoker-plugin

- Deployed zip of repo
  - Consumable as P2 repo with nexus-unzip-plugin
What the heck just happened?

• Just deployed a zip of your P2 repo
And this nexus-unzip-plugin?

- Mirrors a maven repo as virtual P2 repo

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td>Mon Mar 02 04:46:16 CST 2015</td>
<td></td>
</tr>
<tr>
<td>features/</td>
<td>Mon Mar 02 04:46:16 CST 2015</td>
<td></td>
</tr>
<tr>
<td>plugins/</td>
<td>Mon Mar 02 04:46:16 CST 2015</td>
<td></td>
</tr>
<tr>
<td>artifacts.jar</td>
<td>Mon Mar 02 04:46:16 CST 2015</td>
<td>2641</td>
</tr>
<tr>
<td>content.jar</td>
<td>Mon Mar 02 04:46:16 CST 2015</td>
<td>28082</td>
</tr>
</tbody>
</table>
Lets use it!

- Add the shiny new features to your product

```xml
<features>
  <feature id="iaware.platform.core.ml"/>
  <feature id="iaware.platform.library.ml"/>
  <feature id="iaware.solution.allergies.ui"/>
  <feature id="iaware.solution.cc.contentset.personalized.ml"/>
  <feature id="iaware.solution.infusions.core"/>
</features>

<repository>
  <id>platform.features.release</id>
  <url>http://hostname.com/nexus/content/repositories/cerner-feature/</url>
  <releases>
    <enabled>true</enabled>
    <checksumPolicy>fail</checksumPolicy>
  </releases>
  <snapshots>
    <enabled>false</enabled>
  </snapshots>
  <layout>p2</layout>
</repository>
```
• Eclipse product built with shiny new features
What we like/don’t like about Tycho

What we like

- OSGi plugin validation at compile time
- Quick/Easy to make changes
- Automated
- Minimal learning curve, we know maven, just plug-n-play

What we dislike

- Seriously?
- Why is this column even here?
- We love it.
- I can’t believe you fell for that.
Pom-first resolution works for us because our eclipse plugins import/export packages with version ranges
What did I “improve”?

What we had before I joined the team

• Using SSH antrun to deploy

• ALL features (56) built at once

• Build took one hour (for anybody)
Issues with this

• BUILD TOOK AN HOUR
• Want them in a nexus repo
• Stepping on each other’s toes
Baby steps…

- Discovered nexus-unzip-plugin
  - Tricky setup
  - Long paths
  - Be sure to set `useVirtualVersion`

  (were using perl script to workaround before)
Separation of concerns

• Separate for days…

• Avoid branching

• Quicker builds/results for your changes
  • Went from one hour to a few minutes
Your problems are gone

• We have 14 feature repos now (growing)
  • One for platform
  • Rest for solutions
• Want features, we have URLs…
What’s the takeaway here?

- You have no excuse now, you have seen this
- Design your feature repos early
- The more automation, the better
- One size does not fit all
- Using memes to enhance content delivery is completely acceptable
Evaluate the sessions
Sign in: www.eclipsecon.org