How to Make Your Code “OSGi-Friendly”... Without Depending on OSGi!

Neil Bartlett – Paremus Ltd
Target Audience
Java Library Author?
Want your library to be widely used?
...by OSGi users and non-OSGi users?
Ubuntu Bug #1

Proprietary operating systems have a majority market share

Reported by Mark Shuttleworth on 2004-08-20

This bug affects 1722 people

<table>
<thead>
<tr>
<th>Affects</th>
<th>Status</th>
<th>Importance</th>
<th>Assigned to</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clubdistro</td>
<td>Confirmed</td>
<td>Critical</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>Computer Science Ubuntu</td>
<td>Confirmed</td>
<td>Critical</td>
<td><a href="https://launchpad.net/ubuntu/+bug/1">Computer Science Ubuntu Bugs</a></td>
<td></td>
</tr>
<tr>
<td>EasyPeasy Overview</td>
<td>Invalid</td>
<td>Critical</td>
<td>Jon Ramvi</td>
<td></td>
</tr>
<tr>
<td>Ichthux</td>
<td>Invalid</td>
<td>Critical</td>
<td>Raphaël Pinson</td>
<td></td>
</tr>
<tr>
<td>JAK LINUX</td>
<td>Invalid</td>
<td>Critical</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>LibreOffice Productivity Suite</td>
<td>New</td>
<td>Undecided</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>New</td>
<td>Undecided</td>
<td>Bruno</td>
<td></td>
</tr>
<tr>
<td>Linux Mint</td>
<td>In Progress</td>
<td>Undecided</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>OpenOffice</td>
<td>In Progress</td>
<td>Undecided</td>
<td>maviya</td>
<td></td>
</tr>
</tbody>
</table>

[https://launchpad.net/ubuntu/+bug/1](https://launchpad.net/ubuntu/+bug/1)
OSGi Bug #1

Some Java developers are still not using OSGi!

(yet)
Some Java developers are creating libraries that violate modular principals.
• Jigsaw will break lots of libraries too.

• Awesome!!!

• Developers finally stop blaming OSGi for the failure of their own shitty code?
Class Loading
Classpath
Class Identity  
=  
Name + Loader
Class Identity
=
Name + Module
Library:

“Give me class org.example.Foo”
OSGi:

“Duh, which one??”
Developer:

“OSGi was mean to me, waaah!!”
ClassLoader (Ab)use

- Poor-man’s extension framework
  - e.g. Log4J Appenders
- Mapping data to objects
  - e.g. Hibernate
- Lazy initialization, lifecycle control
  - e.g. JAX-RS
Class.forName((String); :

:-(
Class.forName(String, boolean, ClassLoader);
ClassLoader.
loadClass(String);
:-)
Thread.getContextClassLoader().loadClass(String);
I really need to load classes by name...
• Always allow clients to specify ClassLoader

• Can you take a java.lang.Class instead?

• Use your own ClassLoader and/or TCCL as last resort.

• Document what you did!
Laziness/Lifecycle

- Classes can be used as a cheap factory.
- Example: JAX-RS resource registration.
• Classes suck as factories!

• Real factories can:
  • Reuse from a pool
  • Create with context
  • etc

• Even Eclipse (pre 4.0) got this wrong.
ServiceLoader
Okay idea... shame about the implementation
Provide Services via META-INF/services
Consume Services via ServiceLoader.load
Classpath Scanning

- Module A
  - Scanned
- Module B
- Module C
  - IGNORED
- Module D
- Module E
java.util

::

Boot Class Loader

::

Neither overridable nor extensible
• OSGi R5 Enterprise: Service Loader Mediator Spec.

• Bytecode munging of ServiceLoader.load() invocations, at install time.

• You do what you gotta do...
• No lifecycle... what if service provider goes away?

• Mediator forced to refresh the consumer (new wiring, new class loader)

• Hope the consumer doesn’t create any long-lived threads...
Suggestions

- Avoid calling ServiceLoader directly.
- Dependency Injection is your friend.
- Consider PojoSR / OSGi Connect.
- OSGi service registry without the module layer.
Dynamics
Who wants to live forever??
Traditional Java App

Lifetime of JVM

Lifetime of Library
• Clean up those threads, open streams, sockets, windows...

• Careful with Observable Pattern (memory leaks).

• Die nicely when asked to.
Configuration
Whence to load your Configuration?
Config File in
$HOME/.myjunk
• Application config now in 20 different places!

• ... and in 20 different formats including properties, XML, JSON...

• What if config needs to be dynamically changed?

• What if there’s no filesystem??
System Properties
• Also known as GLOBAL VARIABLES

• Not modular!

• Yeah I really want to type War & Peace on the command line...
Suggestions

• Libraries need to fit into the application’s config system...

• ...but they don’t know what that will be!

• Support programmatic configuration.
Recommendations
1. If you need an Object, ask for an Object, not a Class.
2. If you must control the lifecycle, define a factory interface.

(java.lang.Class is a crappy Factory!)
3. If you really need a Class, let me pass a java.lang.Class.

(a String is NOT enough!)
4. If you REALLY need a class name, let me pass you the ClassLoader.

(I know better than you do where to find that class!)
5. Die nicely.

(only use “well-known” config locations, system props etc as a last resort)
Conclusion
KISS

(Keep It Simple, Stupid!)
YANIC
(You Are Not In Charge)