Jigsaw and OSGi:

What the Heck Happens Now?

Neil Bartlett
neil.bartlett@paremus.com

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Jigsaw and OSGi: WTF Happens Now?

Neil Bartlett
neil.bartlett@paremus.com

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Agenda

- WTF is a Module System?
- How do OSGi and Jigsaw Compare?
- Is OSGi Still Useful?
- Can OSGi and Jigsaw Work Together?

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

WTF Is a Module System?

- Isolation of Module Internals
- Controlled & Scoped Dependencies
- Defined Modes of Interaction (Contracts)
- Lifecycle

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

ISOLATION

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Isolation

- Prevent leakage of implementation details
- Enable public vs private distinction
- Avoid supporting private internals as API

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Java Refresher: Visible vs Accessible

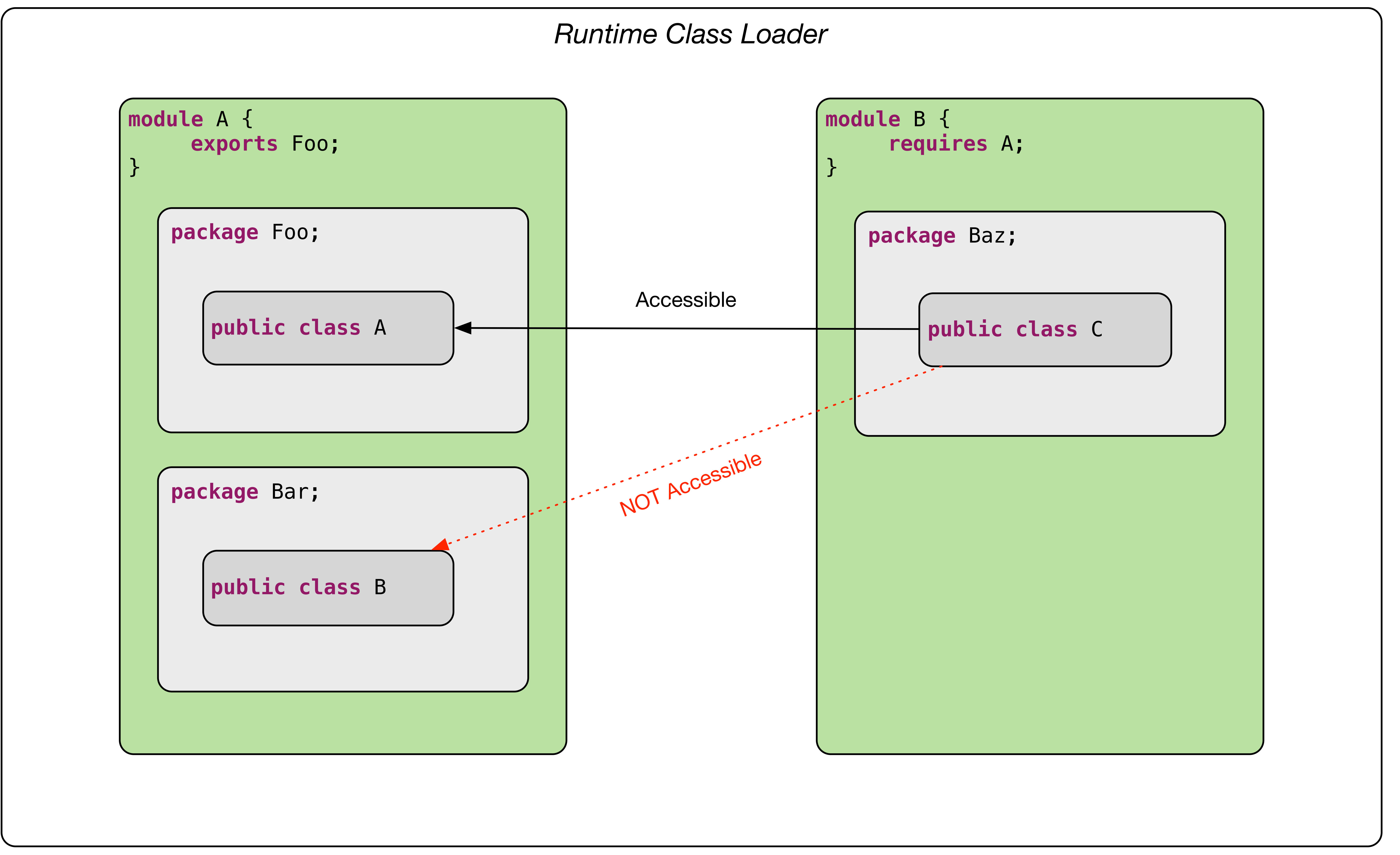
- Accessibility is checked by the compiler…

- Though not always accurately ("runtime packages", JVM Spec 5.4.4)

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Jigsaw Isolation

- Jigsaw redefines the meaning of “public class” in Java.



Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Audience Participation

- **QUESTION**: Can both of these modules be present together on the module path?

```
module A {
  exports foo;
}
```

```
module B {
  exports bar;
}
```

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Audience Participation

- ANSWER: Maybe! Not enough information.

- Modules cannot contain the same packages, even private packages.
- Jigsaw isolation is incomplete: internal details exposed.
- module-info.java is incomplete as a module descriptor.
- Modules on the module path still exist in ONE class loader.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

DEPENDENCIES

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Dependencies

- Isolation is about building walls.
- Any damn fool can build a wall…

- *Almost any damn fool can build a wall.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Dependencies

- Much harder to actually work together.
- Punch holes in the wall.
- Control access to module features, i.e. API.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

OSGi: Requirements & Capabilities

- **Capability**: What I can do. E.g.:

```
Provide-Capability:
   com.acme.display; width:Long=1920; height:Long=1080
Export-Package: org.example.foo; version=1.2.0
```

- **Requirement**: What I need. E.g.:

```
Require-Capability:
   com.acme.display; filter:="(width>=1024)"
Import-Package: org.example.foo; version='[1.2,2.0)'
```

- Depend on what a bundle can do, not its identity.
- Different bundles can implement same idea differently.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Jigsaw Dependencies

- Jigsaw dependencies are entirely based on identity.

```
module B {
  exports bar;
  requires A;
}
```

- No substitution possible.Same module identity for build & run.
- Module identity is API!
- Two artefacts providing the same API **must** have same module identity.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Versions

- Versions in Jigsaw are officially **Somebody Else's Problem.**
- ¯_(ツ)_/¯

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

COMPARISON

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Why is Jigsaw Like This

- It solves the problem it set out to solve: modularise the JDK.
- JDK is a single vendor, closely managed product.
- Not assembled from multiple 3rd party components.
- Released all-at-once, infrequently.

- An API used by 9 MILLION developers, BILLIONS of apps.
- Can’t just refactor into better modular structure!

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

“Typical” Java Applications

- Assembled from many sources, from many organisations.
- Multiple component release cycles.
- Frequent releases… time-to-market critical!

- OSGi has been addressing these challenges for 18 years.
  - Versioning
  - Requirement/Capability
  - Resolving
  - Services
  - FLEXIBILITY

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Ideal Scenario

- We want the best of all worlds.

- Use a modular JDK, provided by Jigsaw…
- … in a modular application, powered by OSGi.

- This will be great!
- Let’s just check it actually works…

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

COMPATIBILITY

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Java 9 Migration

- Jigsaw implements strategies for compatibility and migration to Java 9:
  - Classpath & Unnamed Modules.
  - Automatic Modules.
  - Illegal access not yet enforced in Java 9.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Unnamed Modules

- The classpath still exists in Java 9. Types are loaded into an unnamed module.
  - Unnamed modules implicitly require (“read”) ALL named modules on the modulepath.
  - Named modules cannot read unnamed modules.
- In theory, running any standard Java app on Java 9 should Just Work™.
- “In theory, there is no difference between theory and practice. But in practice, there is.”

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Automatic Modules

- A named module with an implicit definition.
- Drop a plain JAR on the modulepath, it’s now a module.
- Reads every other named module, exports all packages.
- Named modules can depend on automatic modules.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

INTEROP
Level 0

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Level 0: Running

- Felix, Equinox and Knopflerfish all run on Java 9.
- Classpath compatibility mode.
- OSGi runs in an unnamed module.
- Felix Framework requires only java.base module.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Level 0: Challenges

- **Illegal reflective access warnings… can be ignored in Java 9**

- **Hard-coded package lists:**
  - **System Bundle exports *inter alia*** `javax.swing`
  - `javax.swing` **is exported by module** `java.desktop`
  - `java.desktop` **module can now be removed from the platform. Oops!**

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Level 0: Challenges

- OSGi bundles cannot import `java.*`
- Therefore cannot have versioned Import-Package.
- How to check compatibility, e.g. can I call `String.isEmpty`?
- Execution Environment requirement (generated):

```
Require-Capability:
  osgi.ee; filter:="(&(osgi.ee=JavaSE)(version=1.8))"
```

- Execution Environment describes a monolithic platform.
- `java.*` packages can now be removed! Oops.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

INTEROP Level 1

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Detect Platform Packages

- Calculate System Bundle exports using the JPMS Module API.
- Better… filter existing package list through calculated platform packages.
  - Runtime introspection does not show uses constraints.
  - MANY more packages exported than Java SE.
- Working (experimentally) in Felix now.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

DEMO

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Analysis

- Fixes the problem of Java SE packages such as `javax.*`.
- Doesn't fix the problem of `java.*` dependencies
- Exec Env is no longer sufficient.
- Bundles need to indicate their Jigsaw module dependencies.
- Invent a new capability namespace: `jpms.module`

```
Require-Capability:
  jpms.module; filter:="(jpms.module=java.sql)"
```

- Get bnd to generate this obviously.
- Make sure OSGi Framework provides it.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

DEMO

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

JPMS Module Dependencies

- Effectively, OSGi bundles can now depend on JPMS modules.
- Considered for inclusion in OSGi Core Spec?
- However!
  - Many legacy bundles… all existing bundles today…
  - Without the `jpms.module` requirement, we don't know whether a bundle should resolve.
  - With the `jpms.module` requirement, the bundle will never resolve on Java 8 or below :-(
- Maybe run jdeps on-the-fly rather than build time?

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

INTEROP
Level 2+

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Full Bidirectional Interop?

- Very unlikely ever to work.
- Jigsaw modules can only depend on Jigsaw modules.

- No dynamics…
- No split packages…
- No cycles…
  - NB split packages are bad, and cycles are often bad.
  - Unfortunately they are sometimes necessary.
  - Backwards-incompatible to remove from OSGi.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Leverage Jigsaw Access Controls

- Non-exported types in OSGi can still be loaded w/ reflection.
- Use Java 2 Security to prevent this.
- Might be nice to use Jigsaw access control as an alternative?
- Can’t be done in Java 9. Have to go all-in on modules to use this feature (not very modular…).
- Maybe possible in future Java?

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Load OSGi in a Jigsaw Module?

- Run OSGi and all bundles inside a named module.
- Might allow more control over access to the platform.
- Can't be done: module content is immutable.

```
void loadBundle(URL[] bundleClasspath) throws Exception {
  URLClassLoader loader = new URLClassLoader(bundleClasspath);
  Class<?> clazz = loader.loadClass("org.example.MyBundleActivator");
  Module module = clazz.getModule();
  // ...
}
```

- This class will always be the unnamed module, even if the above code is running in a named module.
- We asked for but did not get an `addPackages` method on Module.

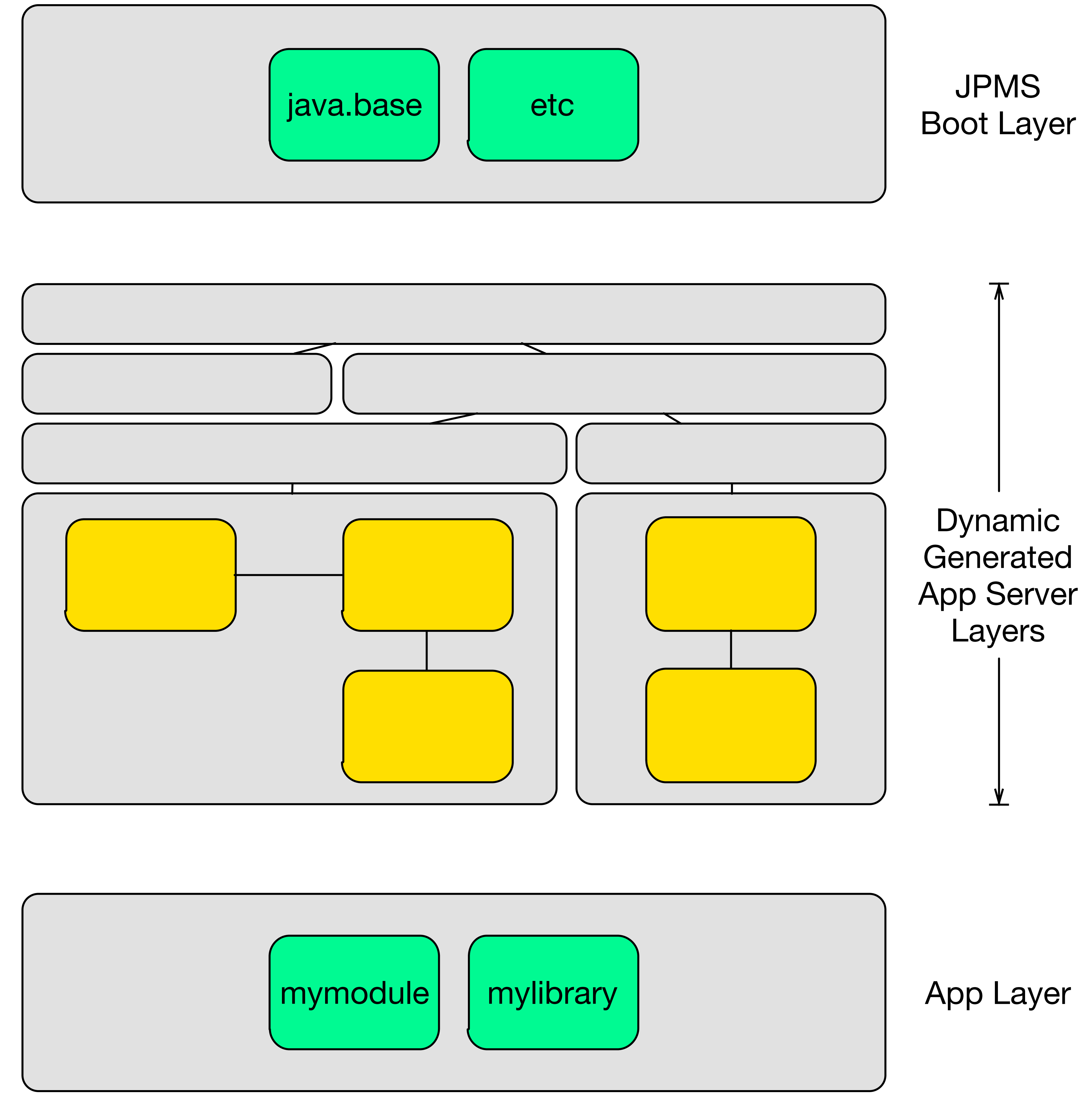
Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

The Tom Watson Experiment

- IBM has App Server products written in OSGi.
- Future customers may need to deploy Jigsaw modules.
- Jigsaw modules can only depend on Jigsaw modules…
- … or Jigsaw Layers.
- Hierarchical groups of modules with inherited readability.
- Dynamically map OSGi Framework subgraphs to Layers.
- Not full bidirectional interop.
- Complex and computationally expensive.
- github.com/tjwatson/osgi-jpms-layer

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

The Tom Watson Experiment: Architecture*



* Approximate, according to my understanding

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Just Pretend It's a Bundle

- Jigsaw metadata is (almost) a subset of OSGi.
- OSGi functionality is (almost) a superset of Jigsaw
- Load a Jigsaw module *as if* it were an OSGi bundle.
- Map `requires` to `Require-Bundle`, etc. Relatively simple to do.
- Huge loss of info:
  - Goodbye Import-Package, Capabilities, Resolver…
  - Goodbye Versions!
- ServiceLoader does not really work… limited value.
- Will discourage open source developers from including OSGi metadata.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Move Up the Stack

- Much of OSGi's value is in the Service Registry…
- … and the fantastic specs that build on top of Services.
- The registry can be factored out of the rest of the framework.
  - "PojoSR" from Karl Pauls, now Felix Connect.
- For some people the module layer simply isn't that interesting.
- App assembly and continuous deployment — enabled by Capabilities & Requirements — are unparalleled.
- SAD.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

The Good News

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

The Good News

- Jigsaw is fantastic for modularising the JDK.
- OSGi is fantastic for modularising applications.
- Already working together.

- Enhanced bnd tooling + jlink = tiny application images.
- Tens of megabytes.

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

The Future Does Not Have to be Scary

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Jigsaw for JDK

OSGi for Apps

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.

Thank You!

Email: neil.bartlett@paremus.com
Twitter: @nbartlett

Copyright © 2017 Paremus Ltd.
May not be reproduced by any means without express permission. All rights reserved.