OSGi Class Loading

makewave

Bundle Classloader

- An OSGi framework must create one class loader per bundle that is resolved and that is not a fragment bundle
- The creation of the class loader may be delayed until it is actually needed

Parent Delegation

- The OSGi framework must always delegate classes that start with `java.*` to the Parent Classloader
- The framework can be instructed to delegate additional classes with the *`org.osgi.framework.bootdelegation`* system property
- Examples:
  - `org.osgi.framework.bootdelegation=*`
  - `org.osgi.framework.bootdelegation=sun.*,com.sun.*`

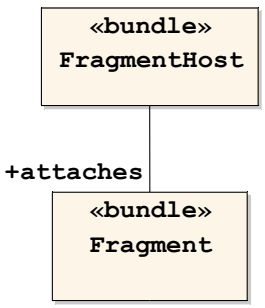
Parent Classloader

- The framework must explicitly export packages other than `java.*` from the System Bundle
- The property *`org.osgi.framework.system.packages`* contains those exports in the same format as the Export-Package manifest header
- Framework implementations must take extra care to make sure that classes exported from the boot classpath are loaded from the boot classpath because some classes on the boot class path assumes that they can be loaded by any classloader because of the prevalent hierarchical model

Wires

- A bundle that has been resolved is said to have a wire to the class loader of the exporting bundle for each of its imported packages

Fragments



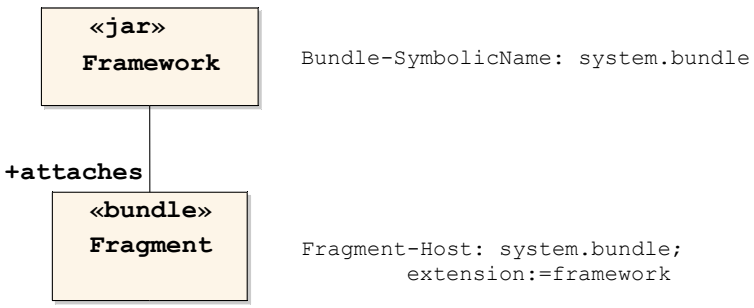

```
Bundle-SymbolicName: example.host
Bundle-Version: 1.0.0
```

```
Fragment-Host: example.host;bundle-
version="[1.0.0,2.0.0)”
```

Fragments

- A fragment
  - must no have a `Bundle-Activator` header,
  - can only attach to one host,
  - is in the state Resolved when attached.
- A fragment host bundle
  - can have **0...n** fragments attached,
  - will **export** all packages exported by attached fragments,
  - will **import** all packages imported by attached fragments,
  - will **require** all bundles that attached fragments requires.
- Fragments are
  - **attached** to the host when it becomes resolved,
  - **detached** from the host when it becomes unresolved.
- Attached fragments are search in bundle id order (lowest first).

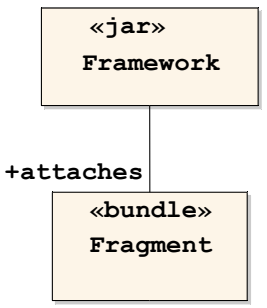
Extension (framework)



Extension (framework)

- Extension bundles are fragments that attaches to the system bundle.
  - Delivers optional parts of the framework implementation.
- The entire framework must be re-launched if an extension bundle is refreshed.
- If an extension bundle is updated the new version must not resolve unless a refresh on it have been made (i.e., the entire framework must be re-launched).
- An extension bundle
  - must not specify `Import-Package`, `Require-Bundle`, `Bundle-NativeCode`, `DynamicImport-Package`, `Bundle-Activator`,
  - exports packages via the system bundle.
- Extension bundles must be appended to the class path of the class loader of the framework implementation in the order in which the extension bundles are installed: that is, ascending bundle ID order.

Extension (bootclasspath)



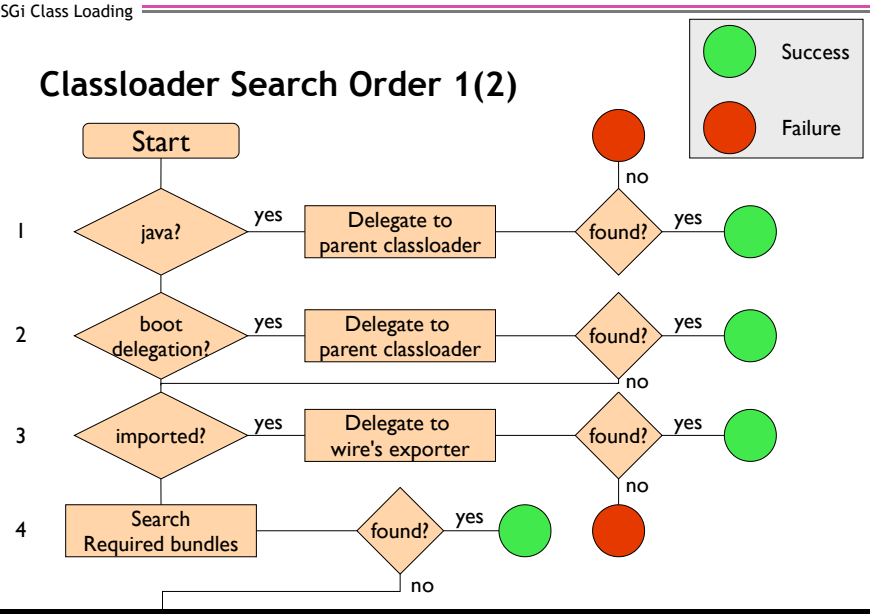

```
Bundle-SymbolicName: example.framework
```

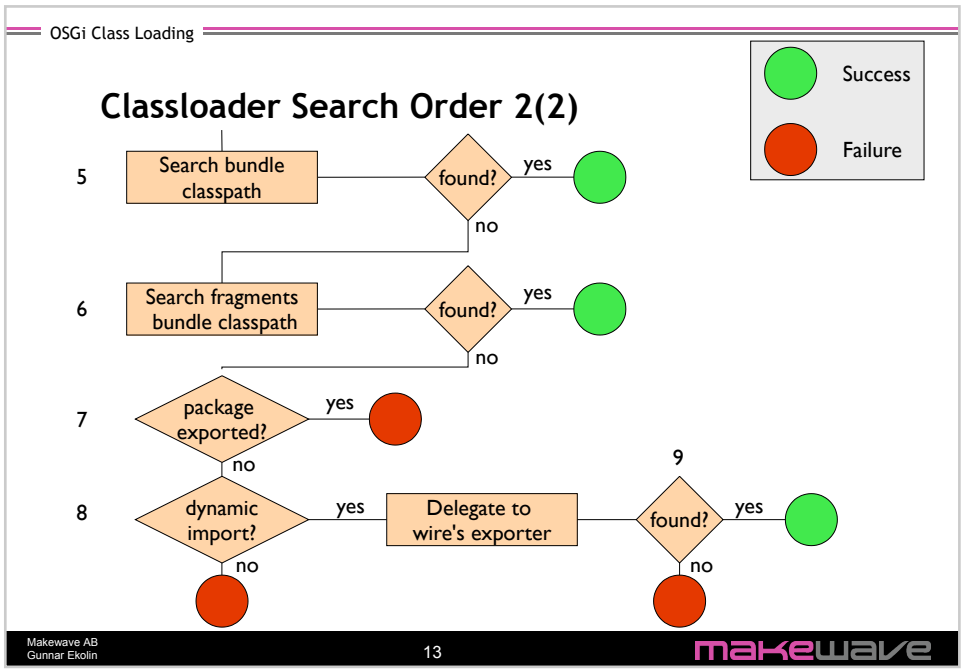
```
Fragment-Host: system.bundle;
        extension:=bootclasspath
```

Extension (bootclasspath)

- Same rules as for extension bundles with a single exception:
  - The jar-files of boot class path extension must be added to the boot class path of the JVM executing the framework.
- Used for JDBC drivers and other legacy code that must be on the boot class path to work properly.

Classloader Search Order 1(2)




OSGi Class Loading
Classloader Search Order 2(2)
Success
Failure
5
Search bundle classpath
found?
yes
no
6
Search fragments bundle classpath
found?
yes
no
7
package exported?
yes
no
8
dynamic import?
yes
no
Delegate to wire's exporter
9
found?
yes
no
Makewave AB
Gunnar Ekolin
13
makewave


Thank You!
Gunnar Ekolin
www.makewave.com
ekolin@makewave.com
© Bruce Teleky
Makewave AB
Gunnar Ekolin
14
makewave