MELD OSGI BUNDLES WITH JAVA MODULES
OSGI RFP 190

Udo Hafermann
Todor Boev
MOTIVATION

Example: My bundle requires AWT—what is the minimum Java module set?
OVERVIEW

This talk presents RFP 190: Resource Encoding of Java Modules

1. Application Domain and Problem Statement
2. Use Cases
3. Requirements
4. Proof of Concept
5. Open Ends
PROBLEM STATEMENT

• We want self-contained Java applications with minimal footprint
• For OSGi, we can resolve the set of required bundles
• But JRE can easily dominate the overall size
• With JPMS, we can strip down to the modules required

• How to determine which ones are required?
• R7 provides the flat list of all boot layer packages
• Removing modules from the boot layer is a trial and error process
• We need the modules as such to be visible to tools and runtime
USE CASES

- Build a compact Docker image
- Build a self-contained, no-installation tool
- Analyze JRE footprint, bundle by bundle
- Introspect JRE at runtime
- Create hybrid app of OSGi bundles and JPMS modules
GENERAL REQUIREMENTS

• Resolver can still be used to build consistent set of modules and bundles
  – Should not require a new tool or component
• Bundles are not forced to explicitly reference modules
  – Just as we don’t want bundles to explicitly reference bundles
• A bundle may function as a module
  – OSGi framework as a special case
• Includes third-party modules
• Best-effort JPMS validation
  – Rejects duplicate and (optionally) conflicting and cyclic modules
RESOLVING DEPENDENCIES

- Module identity and version
- Bundle package imports to module exports
- Modules to modules
- (Optional) Module services as resources
- (Optional) Requirements to module services
ENHANCING METADATA

• Add version range requirements for modules
• Use bundle annotations for additional JPMS module dependencies
• Replace org.osgi.framework.system.packages with module resource representations
PROOF OF CONCEPT

• Modeling
  – Module as special kind of bundle
  – Require module as require-bundle
  – Module exports as export-package
  – Module provides as service-loader capability
  – Module uses as service-loader requirement

• Extracts of actual metadata
• Run resolver on “awt” example
DEMO TEST

- org.example.awtclient 1.0.0
- osgi.wiring.package; filter:=(&(osgi.wiring.package=java.awt)(version>=0.0.0))
THANK YOU

• Feedback welcome on https://github.com/osgi/design/blob/master/rfps/rfp-0190-Resource%20encoding%20for%20Java%20Modules.odt