OSGi in Java EE servers: sneak peek inside

Krasimir Semerdzhiev (SAP AG)
EclipseCon Europe 2012
Why care at all?

Most common application scenarios:

- Reuse open-source components:
  more than 80% percent of enterprises embed **open source** components in software /Source: SonaType/
  - Eclipse ones are (typically) OSGIfied
  - Apache/GitHub/GitHub/GoogleCode/SourceForge/etc mostly have no OSGi packaging

- Modularize/”OSGIfy” Java EE applications

- Getting rid of proprietary module systems in favor of standard one(s). Anyone?

- Software refactoring and evolution
General approach – That’s our starting point!
General approach – How to eat an elephant?

Java EE monolithic app

Libraries in:
\lib
\META-INF\lib
\WEB-INF\lib

Java EE container
General approach – Peter Kriens: One piece at a time!

Java EE monolithic app

- Libraries in:
  \lib
  \WEB-INF\lib

Java EE container

Java Web Bundle

- Shared libraries

OSGi-enabled Java EE container
General approach – Peter Kriens: One piece at a time!

Java EE monolithic app

Libraries in:
\lib
WEB-INF\lib

Java EE container

Java Web Bundle

Shared libraries

OSGi bundle  OSGi bundle  OSGi bundle  OSGi bundle

OSGi-enabled Java EE container
General approach – Peter Kriens: One piece at a time!

Java EE monolithic app

Libraries in:
\lib
\WEB-INF\lib

Java EE container

Java Web Bundle

Shared Libraries

OSGi bundle

OSGi bundle

OSGi bundle

OSGi bundle

OSGi-enabled Java EE container
General approach – Peter Kriens: One piece at a time!

Java EE monolithic app

Libraries in:
\lib
\WEB-INF\lib

Java EE container

Web Bundle

Shared Libraries

OSGi bundle

OSGi bundle

OSGi bundle

OSGi bundle

OSGi-enabled Java EE container

Projects typically stop at that state…
General approach – Peter Kriens: One piece at a time!

Very unlikely to reach this state! Don’t worry! That’s OK!
General approach – Keep the goal in sight!

It’s **not** for those reasons:

- Single bundle can’t be reliably stopped!
- Single bundle can’t be reliably updated!
- Reference caching further spoils the party!
- Random startup order as a surprise for non-aware modules.
- Reference resolution for naïve package imports without a version might become tricky.
We go for OSGi - which server to run on?

Java EE 6 compatibility page (23.Oct.2012)

Java EE 6 Web profile

- Cauch Resin 4.0.17
  - Tested Configuration
- Apache TomEE 1.0
  - Tested Configuration
- JBoss Enterprise Application Platform 6
  - Tested Configuration

Java EE 6 Full profile

- Oracle GlassFish Server 3.x
  - Tested Configuration
- TMAX JEUS 7
  - Tested Configuration
- Cosminexus
- JBoss Application Server 7.x
  - Tested Configuration
- Hitachi uCosminexus Application Server v9.0
- IBM WebSphere Application Server 8.x
- Fujitsu Interstage Application Server
  - powered by Windows Azure
  - Tested Configuration
Tested scenario: Granny’s address book

Source: Original sources + packaging on GitHub
Apache TomEE 1.0

OSGi support

- None whatsoever
  - Neither uses, nor provides any
- Modularity: none – stick to Tomcat webapps
  - Libs can be put in [tomee\lib] folder
  - No activators/resolution. Works like classpath.

Overall experience

- Feels very lightweight even with a bigger WAR file.
- TomEE doesn’t currently support web.xml <resource-ref> definitions in web.xml. To be fixed soon.
- Doing partial application update is not supported by the server. Requires 3-rd party tools like LiveRebel for example.
OSGi support

- None whatsoever (but [Why OSGi is cool, but not for enterprise apps blog](#))
- Modularity: Own “pomegranate” module system, dependencies in pom.xml
  - Additional JARs can be put in [resin\ext-lib](#) folder
  - No activators/resolution. Works like classpath.

Overall experience

- Easy to handle
- No default datasource is bound in JNDI. Therefore app startup fails if not created manually.
JBoss Application Server 7(web)

OSGi support

- JBoss own OSGi impl.
  - Tutorial available over here
- Additional bundles handling
  - Transform the WAR as a Web Bundle Archive (WAB) JAR.
  - Add the WAB + all related bundles in standalone\deployment “auto-deploy” folder

Overall experience

- Easy to use! A bit hard to find the application web alias…
- Both web console + CLI admin deploy only single archives. No batching.
- Deployment passes successfully although bundles can’t be resolved at that time.
- Startup fails silently afterwards.
OSGi support

- HK2 is based on Apache Felix
- Additional bundles handling
  - Deploy bundles one by one via asadmin CLI tool.

Overall experience

- Deploy fails correctly on failed resolution. Have to handy-pick all dependencies one by one.
- Web console seems to cautiously ignore OSGi :-(
Apache Geronimo 3.0

OSGi support

- Based on Apache Felix
- Additional bundles handling
  - Deployment via the Eclipse development tools

Overall experience

- Build as both Web Profile and Full Profile, but never cared to certify against the Web Profile
Other projects to watch out for

WebSphere Application Server V8.5 Liberty Profile

Eclipse Virgo
You still want to take the journey?

Take a look at @glynnormington “Is OSGi modularity always worth it?” session.

Management is very susceptible to “costs” and “ROI” arguments.

Make the cost calculations carefully and use the numbers wisely!
Thank You!
krasimir.semerdzhiyev@sap.com