Dr Andy Piper | Oracle Corp.

OSGi and Android – or how to train your appserver

Disclaimer

- The opinions represented in these slides are entirely my own not Oracle's
- These slides do not represent any kind of indication of or commitment to Oracle's future products or future product strategy

Me

- Dr Andy Piper, MA, MBA
- Software architect and engineering manager for Oracle Complex Event Processing at Oracle
- ex-BEA, WebLogic Server core architect
- Author “Spring Dynamic Modules in Action”

Background

- Android supports Java
  - But what does this really mean?
- OSGi has been shown to run on Android
  - Equinox on OSGi at EclipseCon '08 (Bartlett/IBM)
  - Felix on OSGi with EZDroid (Luminis)
  - Various commercial offerings (Prosyst mBS Mobile)

Don't cross the streams!

- No-one appears to be trying any full-fat apps on Android (i.e. non-phone apps)
- What happens if you try something big – like an application server, say?
- What problems do you run into?

The challenger

- OCEP
  - OSGi-based appserver (a bit like Virgo)
  - Spring
  - Spring DM
  - Glassfish JAXB
  - Jetty
  - Hessian
  - JCL
  - Commonj
  - And much more …

The challenger at the weigh-in

- Default OCEP installation:
  - 224 OSGi bundles
  - 125 Mb on-disk
  - 162 Mb in-memory
- OCEP re-factored to create a minimal set of bundles
  - Straightforward thanks to OSGi
  - Supports a **helloworld** application, but not much else
  - 113 bundles
  - 24 Mb on-disk
  - 92 Mb in-memory

Android

- Google Phone OS
- Linux-based kernel and utilities
- Applications are written in Java
  - Dalvik VM runs applications
    - One VM per-application - no JIT! (except 2.2)
  - Proprietary bytecode format
    - Optimized for small CPU & memory footprint
    - Java bytecode can be translated
  - Package profile similar to Java ME
    - Many features missing, e.g. JNDI, parts of javax.xml, JMX and other enterprise features
    - Based on Apache Harmony class libraries

Android & OSGi

- Dalvik does not support dynamic class definition
  - Problem for OSGi, but there are some workarounds
- Felix (Apache) now works
  - Work by Luminis
  - EZdroid project based on Apache ACE
- Equinox demonstrated at EclipseCon 08
  - Fixes do not work with latest Android and Equinox
  - But can be made to work
- Other commercial offerings available

OCEP on Android

- OCEP re-packaged as dalvik bytecode jars
- Equinox issues resolved
- Minimal feature-set bundles will all install on Android emulator (256Mb RAM, 256Mb flash)
- But there many problems …

Problems with dalvik profile

- Problem is the missing packages in the dalvik profile
  - For instance OCEP relies on Java SE XML features not present in profile
  - Very hard to simulate since some are in java.*
- In theory this can be worked around by providing the packages
  - Add xerces
  - Add Sun 1.2 JNDI impl
  - Add mx4j
- But the list is quite large

Developing a dalvik profile

- Easy to test without Android!
  - –Dosgi.java.profile allows the profile of any VM to be limited
    - Still does not simulate missing java.* packages
  - We have a Dalvik profile
  - If it works on Windows/Linux it may work on Dalvik
- We have a small footprint profile as a starting point

Really, really small profile

- JMX gone (no RMI)
- RMI gone (no RMI OP implementations)
- JNDI gone (no RMI)
- cglib gone (no dynamic class definition in Dalvik)
- JMS gone
- datasource gone (too many dependencies)
- transactions gone (no datasource)

Even with the small profile

- Missing packages cause problems
  - java.rmi
  - java.beans (some of; related to awt & swing)
  - javax.security.auth
  - javax.management
  - javax.naming
  - javax.annotation
  - javax.xml.validation & friends
  - java.sql.SQLXML
- javax outages can be fixed through apache libraries
- java outages more tricky

Start hacking!

- java.rmi
  - Engine dependent on Remote & RemoteException
    - Package these up, but ideally factor out dependency
  - Config requires java.rmi.dgc
    - Package up, but need to factor out
- java.beans
  - Particularly problematic. Android includes some classes
    - But not the useful ones (heavily used by Spring)!
  - Take harmony impl and hack out awt & swing references
  - Recompile & package up
- Both need to be included on bootclasspath

More hacking

- java.sql.SQLXML
  - Can make the dependency dynamic through Class.forName()
- XML parser problems
  - Cannot use Android and xerces together
  - Have to completely replace implementation
  - Implementation has assumptions about the ContextClassLoader
    - Hack Factory's to fallback to bundle classloader (changes submitted to apache)
  - Compiled versions contain bad bytecode!
    - Recompile

… and more …

- JAXB Problems
  - Contains marshalling support for types that do not exist on Android
    - Hack to remove these dependencies
  - Issues with classloading (JAXB or Harmony bug)
    - DynamicImport-Package: *
  - Dalvik hasn't implemented Package.getDeclaredAnnotations()
    - Get a LinkageError
    - Hack JAXB to remove dependency

… and finally

- Dalvik doesn't like null as a parent classloader
  - Hack Spring DM and Equinox to remove (changes kindly incorporated by Thomas Watson)
- Bundle.getResources() can return null which is not the same semantic as ClassLoader.findResources()
  - Hack Spring DM and Equinox to return empty collection
- Many missing imports when classloading is more strict
- Logging requires boot delegation for META-INF.services to work

But it works!

- 12 extra bundles/jars
  - 3 vanilla SpringSource EBR bundles
  - 9 custom patched bundles/jars
- A few Android specific hooks in the product
  - Disable JMX
  - Disable cglib

But…

- It's really slow
  - The JIT blows up if you try it (not tried later JITs)
  - Java ME JIT is typically 6x faster than interpreted code
- It's really hard to debug
  - You can attach remotely
  - But the emulator blows up a lot
  - Did I mention it's slow?

The good

- If your application runs on OSGi, refactoring for footprint is relatively easy
- Equinox profiles makes testing easy
- Linux/Java on a phone makes for a very low developer barrier to entry (contrast with other platforms)
- You can run a full-fat application on Android/Dalvik
- Non-Dalvik offerings (e.g. Java ME) are in the works

The bad & ugly

- Android class libraries are just broken
  - What were they thinking!
  - Need to support some recognized VM profile or recognized subset
  - Standardization is necessary
- Dalvik VM limitations are a significant hindrance to development

Questions?

- Contact: andy.piper@oracle.com