How we use OSGi to build Open Liberty

Alasdair Nottingham - IBM
Background
Project goals

- Implement Jakarta EE
- Small Footprint
- Start fast
- Composable
- Dynamic
- Easy to use
Just Enough App Server

- You control which features are loaded into each server instance

```xml
<feature>jsf-2.3</feature>
```

Features:
- jsp-2.3
- jsf-2.3
- Servlet-4.0
- http-2.0
- appmgr
Server configuration

<server>
  <featureManager>
    <feature>javaee-8.0</feature>
  </featureManager>

  <httpEndpoint id="defaultHttpEndpoint" httpPort="8080"/>

  <webApplication location="myWeb.war" contextRoot="/"/>
</server>

- Metatype describes config
- DS describes config
- CA parses XML
- Subsystem Feature describes bundles for feature
- DS component activated to start http transport
- DS component activated to start application
Jakarta EE on OSGi
Last 12 months
Towards one second start-up

2019 Progression of OpenLiberty+OpenJ9 startup time (seconds)

**Almost halved startup time** due to app server and JVM improvements through 2019

2 hyperthreaded cpus on 2 socket system, each socket containing 24 cores. CPU model: Intel Xeon Platinum 8168 CPU @ 2.70 GHz

Application PingPerf - [https://github.com/HotswapProjects/pingperf](https://github.com/HotswapProjects/pingperf)
Performance Improvements

- Equinox
- Regions (Equinox)
- Metatype (Equinox)
- Declarative Services (Felix SCR)
- Config Admin (Liberty)
- Subsystem Features (Liberty)
Things that are expensive

- findEntries

  ```java
  Enumeration e = b.findEntries("OSGI-INF", "*.xml", true);
  ```

- Searching jar files for sub-paths is expensive
- XML Parsing
- Text file parsing
- Opening and reading many files vs one
- Case insensitive matching in filters
- Reflection to find methods on DS components
Parallel Bundle start

- Equinox activates bundle one by one
- Updated to support starting bundles within start level in parallel
- Turns out a lot of code actually depends on bundle start order when you have a predictable install order
Things we learned

- Shutdown is not as simple as stopping the framework
- Statics and service do not mix & match
- Use the build tools
- Very powerful for large complex software
- DS and ConfigAdmin together are brilliant
- High learning curve
- Java SE classloading assumptions don’t mix well in OSGi
- Shuffle bundle install order to avoid start time dependencies
Thoughts on future
• Can Liberty function without the OSGi Modularity layer?
• Can we choose the modularity layer and swap between OSGi, JPMS and none?
• Stick with DS, CA, services, but not OSGi bundles