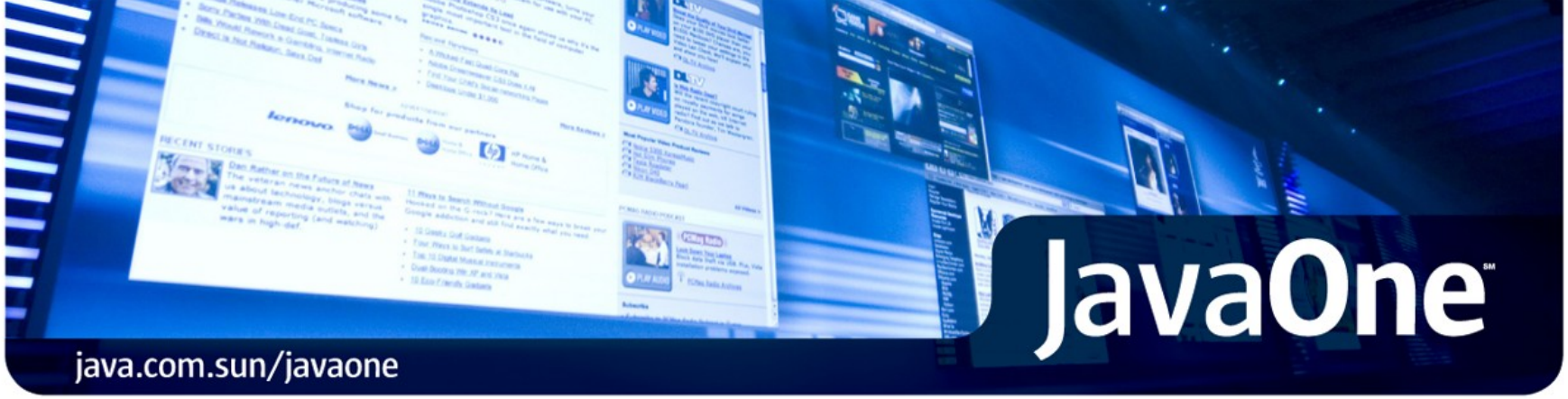


Developing Service-Oriented Architecture Applications with OSGi

Dr Mark Little, Kevin Conner (Red Hat), Keith Babo (Sun), Alexandre Alves (BEA)

BOF-5846

Java

Sun microsystems

Overview

- Using OSGi in real-world Service Oriented Infrastructures
  - Some use cases
  - What's good and bad about OSGi in these situations
  - What could be improved
- Q&A

Speaker Background – Mark Little

- Director of Engineering for JBoss SOA Platform
  - Ex-lead of JBossESB and JBossTS
- Co-author of many WS-* specifications and standards
  - WS-Context, WS-Addressing, WS-Transactions
  - SCA
- Member of JBI 2.0
- OSGi expert group
- 20+ years working on architecting and building reliable distributed systems

Speaker Background – Keith Babo

- Architect and developer in SOA / Business Integration organisation at Sun
  - Java CAPS
  - Open ESB
- Member of JBI 1.0 (JSR 208) and JBI 2.0 (JSR 312) Expert Groups
- Exposure to OSGi through Project Fuji
  - JBI framework implementation based on OSGi
  - Core SOA runtime for Open ESB v3 and Glassfish ESB
  - http://fuji.dev.java.net

Speaker Background – Kevin Conner

- Project Lead and Architect for JBoss ESB
- OSGi development as part of our next major ESB release
  - OSGi is a core framework in our ESB implementation.
  - Will be included in the JBoss SOA Platform.
  - http://www.jboss.org/jbossesb/

Speaker Background – Alexandre Alves

- Representative at the BPEL TC at OASIS
- Architect for BEA Weblogic Event Server (WL EvS)
- WL EvS
  - *Domain-specific* application server for real-time event processing
  - Completely built on top of OSGi technology
  - All components in WL EvS, both system and application components, are OSGi bundles

Requirements

- Service as the unit of re-use
  - Loose coupling
  - Cohesive services
  - Interoperability
  - Portability
- Deployment
  - One service per process (VM)
  - Multiple services per process

More requirements

- Versioning of services
  - Not necessarily in the same VM
- Enterprise features
  - transactions, security, reliability, etc.
- Widely deployable environments
  - Not just “fat” clients or services, e.g., mobile phones
  - Secret of cloud computing
- Language agnostic
  - Well ...

Q&A

- How are these requirements realised in OSGi?
- Lessons learnt from using OSGi for SOA?
- Pros and Cons of using OSGi for SOA?
  - What would you like to see improved?
- Questions from the audience ...

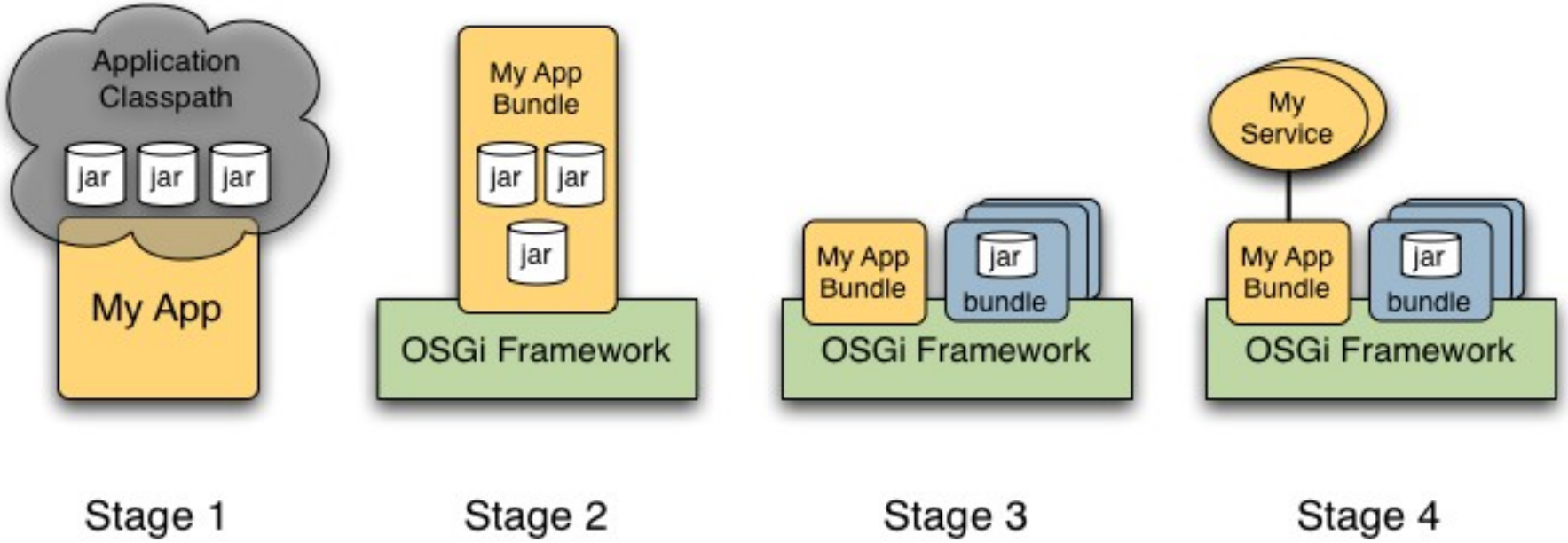
Q&A

- How are these requirements realised in OSGi?
- Lessons learnt from using OSGi for SOA?
- Pros and Cons of using OSGi for SOA?
  - What would you like to see improved?
- Questions from the audience ...

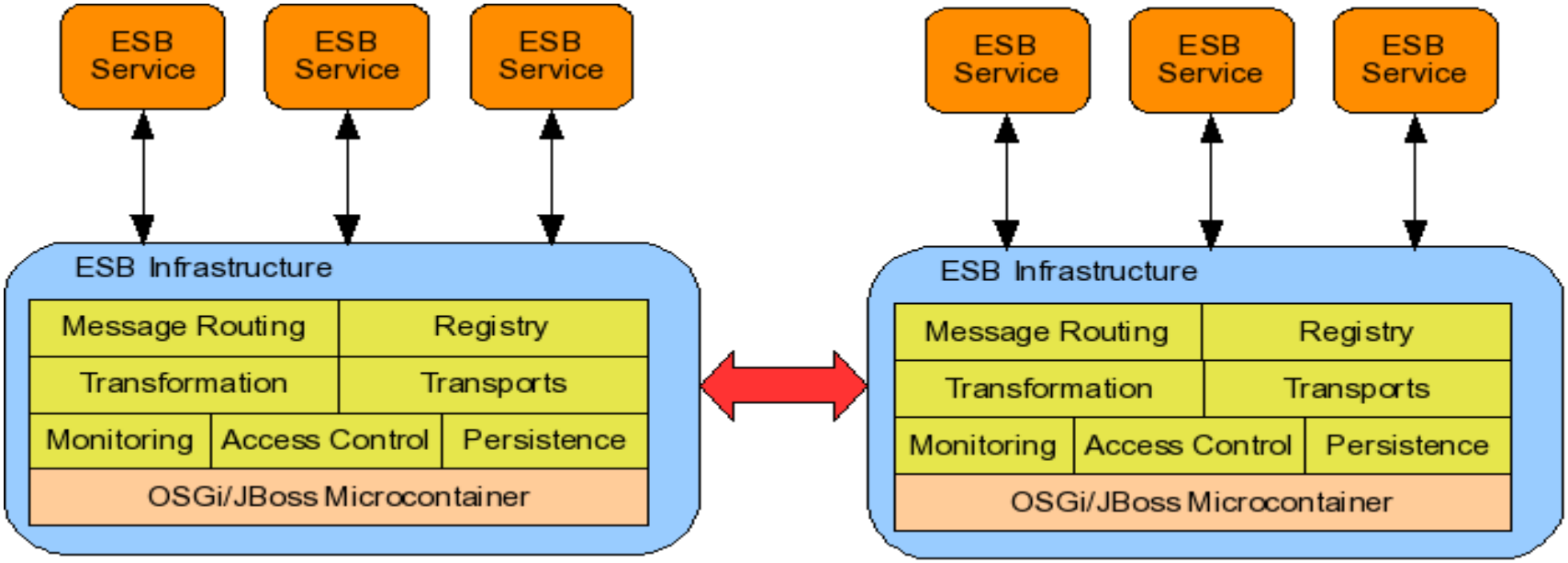
The Role of OSGi in a SOA Runtime

- Pluggability
  - Ability to extend the runtime with additional functionality (services, containers, etc.)
- Isolation
  - Total control over the packages you expose and consume
- Dynamism
  - Bundles have a life cycle independent of the virtual machine
- Dependency Management
  - First and worst form of coupling you encounter
  - Coupling between services
  - Coupling between a service and its underlying runtime

Evolution of a Modular Application



JBoss ESB Architecture



Modularization

- Explicit dependencies
  - Helped with the modularization effort in WLS
  - WLS provides over 100 bundles to other products
- Improves automation of build process
- Versioning and name ratification
  - Helped manage the promotion of individual features, rather than release all-or-nothing
- Feature-Sets
  - Grouping of features (e.g. enterprise, messaging) so that end-user can profile runtime
  - **“Just-Enough App Server for …”**

Services

- De-coupling of interface and implementation allows the selection of different implementation providers
  - Cache
    - Native implementation from WLS, Tangosol, etc
  - Web-containers
    - Jetty, Tomcat
  - Authentication/Authorization providers
    - LDAP, file-system
- Service Management
  - Runtime monitor of services, offending services can be un-registered, and swapped

Q&A

- How are these requirements realised in OSGi?
- Lessons learnt from using OSGi for SOA?
- Pros and Cons of using OSGi for SOA?
  - What would you like to see improved?
- Questions from the audience ...

What have we learned?

- Do not use OSGi directly.
- Be dynamic.
  - Use ServiceTracker.
- Test with multiple implementations
  - Felix.
  - Equinox.
  - JBoss Microcontainer.
- Restrict your execution environment.
- Use 'bnd' to generate bundles
  - We do not use eclipse PDE.

What We Get Out of OSGi

- Completely replaced our own modularity layer with OSGi
  - OSGi R4 API is fantastic!
  - API covered 90% of what we needed, other 10% can be worked around.
- Everything is a bundle
  - Containers
  - Composite Applications
  - Framework
- OSGi service registry was a nice bonus feature
  - Whiteboard pattern

Extensibility and Portability

- WL EvS provides Service Provider Interface
  - Vendors can plug-in different processing engines, cache providers, etc.
- SPI implemented using the Whiteboard pattern
  - Vendors implement service interfaces and register providers in the OSGi registry
  - Infrastructure selects services based upon ranking and properties
  - Simple, easy, and powerful (dynamic)
- WL EvS applications are bundles
  - Standard-based deployment unit (although services being used are not standard today)

Greater Re-use of Core Infrastructure

- HTTP Service
- Service Tracker
- Initial Provisioning
- Declarative Services using Spring-DM
- Start Level Service
- … Event Admin Service, Conditional Permission Admin

Q&A

- How are these requirements realised in OSGi?
- Lessons learnt from using OSGi for SOA?
- Pros and Cons of using OSGi for SOA?
  - What would you like to see improved?
- Questions from the audience ...

Why do we like OSGi?

- Improved modularization
- Service management
- Extensibility and portability
- Greater potential for re-use of infrastructure

Challenges

- Very large *Import-Packages*
  - Hard to get correct, especially when reflection is used (e.g. Kodo)
- Complex class-path resolving
  - *instanceof* fails… Hard to debug and find problem
- Service availability race-conditions
  - Client applications referencing to services that have not been bound it
  - Particularly a problem during start-up

What's Missing?

- No BundleEvent.UNINSTALLING?
  - I need to perform some cleanup before a bundle is removed
  - By the time BundleEvent.UNINSTALLED fires, the bundle is long gone
- Service Provisioning Life Cycle
  - Services life cycle today : registered, unregistered
  - Would like to see a two-phase life cycle that separates consuming and providing services
- Message-based interaction between services
  - Synchronous call-through on a Java interface still couples your service

Why do we like OSGi?

- Class Loading Architecture
  - Explicit control over the classes being consumed and exposed.
  - Dependencies are based on versions.
  - Multiple versions can co-exist.
- Dynamic Bundles
  - A well defined lifecycle.
  - Programmatic control.
- Services & Registry
  - Used for our core infrastructure.
- The adoption of OSGi is growing.
  - Mobile through to enterprise platforms

What do we think is missing?

- Enterprise features
  - Transactions, JCA etc.
- Standardised remote access to services
- Asynchronous invocations

Q&A

- How are these requirements realised in OSGi?
- Lessons learnt from using OSGi for SOA?
- Pros and Cons of using OSGi for SOA?
  - What would you like to see improved?
- Questions from the audience ...