JBoss Fuse Overview

Presented by Matt Pavlovich
matt@mediadriver.com
January 27, 2015
Agenda

• Overview of JBoss Fuse
• JBoss Fuse Subsystems
  • Container and Fabric
  • Reliable Messaging
  • Integration Framework
  • Web Services Framework
  • Tooling and Monitoring
• Why JBoss Fuse?
• Beyond ESB
• Best Practices
• Demo
• Q & A
About Your Presenter

• Technical Practice Lead at Media Driver
  • Leading Consulting Services and Product company for JBoss Fuse
  • Providing technical solutions for Fortune 500 companies
• Over 8 years of professional experience with JBoss Fuse Technologies
  • Apache ServiceMix, ActiveMQ, Camel, CXF, Karaf and others
• Apache Committer on ActiveMQ
• Long history of Open Source
  • Debian GNU/Linux, Linux Video and DVD, UTAH-GLX, Samba, OpenLDAP, Courier MTA, others
EIP Background

- Enterprise Integration Patterns by Gregor Hohpe and Bobby Wolfe
- Considered a seminal tome on EIP
- Collection of 65 patterns
- Patterns grouped into 8 categories
  - Integration Styles
  - Messaging System
  - Message Channels
  - Message Construction
  - Message Routing
  - Message Transformation
  - Message Endpoints
  - System Management
- http://www.enterpriseintegrationpatterns.com
Overview JBoss Fuse

Development & Tooling
- JBoss Developer Studio including JBoss Fuse IDE
- Develop, test, debug, refine, deploy

Web Services Framework
- Web Services Standards
- SOAP, XML/HTTP, ReST HTTP
- Apache CXF

Integration Framework
- Transformation, mediation, Enterprise Integration Patterns
- Apache Camel

Reliable Messaging
- JMS/STOMP/NMS/MQTP
- publish/subscribe, point-to-point, store and forward
- Apache ActiveMQ

Container
- Life cycle management, resources management, dynamic deployment, security and provisioning
- Apache Karaf and Fuse Fabric

Red Hat Enterprise Linux
- Windows, Unix, other Linux

Management & Monitoring
- System and web services metrics, automated discovery, container status, automatic updates
- JBoss Operations Network (JON)
- JBoss Fabric Management Console
Container

- Based on Apache Karaf
  - OSGi-based runtime
  - Provides a lightweight container onto which various components applications can be deployed
    - Logging, Authentication, Configuration, Monitoring, Deployment
  - Supports Apache Felix and Eclipse Equinox
Container

• Features include
  • Hot deployment of OSGi bundles, jar’s, war’s, Spring and Blueprint XML files
  • Dynamic configuration of bundles
  • Comprehensive Logging System
  • Native Maven-based Provisioning
  • Native OS integration
  • Extensible Shell console via SSH
  • Remote access via JMX
  • Security framework based on JAAS
  • Multi-Instance Management
Fabric

- Leverages Apache ZooKeeper
  - A distributed registry service
- Features include
  - Simplifies A-MQ and Fuse configuration
    - Do once and then leverage
  - Support for hybrid deployments (on premise / on cloud / both)
  - Endpoints can be relocated / load balanced / elastic / highly available
  - Distributed Configuration
    - Configuration may be accessed across multiple domains
    - Configuration is highly available
  - Distributed Management
    - Easy elastic scaling of services
    - Monitoring and control of resources
Reliable Messaging

• Based on Apache ActiveMQ
  • Standards-based messaging server
  • Allows for point-to-point, publish/subscribe, store and forward messaging

• Features include
  • Supports JMS 1.1, J2EE 1.4 and JCA 1.5
  • Multiple Language Clients and Protocols
    • Java / C# / C++ / Perl / Ruby
    • OpenWire / STOMP / ReST / MQTT / AMQP / HTTP
  • Advanced capabilities
    • Message Groups / Virtual Destinations / Composite Destinations
  • Fast, reliable persistence store
  • High performance
  • Extensible via plugin architecture
Integration Framework

• Based on Apache Camel
  • Integration framework based on popular Enterprise Integration Patterns
  • Includes routing and mediation rules
• Features include
  • Routes can be configured via DSLs (Java/Scala) or Spring/Blueprint XML Configuration
  • Extensive Component Library
    • FTP, File, Exec
    • HTTP, SOAP, ReST
    • JMS, STOMP
  • Multiple Language Support
    • Constant
    • Simple
    • XPATH
    • Various scripting languages (JavaScript / Groovy / Ruby / PHP)
Web Services Framework

- Based on Apache CXF
  - Framework that allows development/deployment of web services
  - Essential component in build Services Oriented Architectures (SOA)
- Features include
  - Variety of protocol support: SOAP / ReST / POX
  - Variety of transport support:
    - HTTP / JMS / JBI
    - SMTP/POP3-Jabber*
  - Rich standards support: JAX-WS / JAX-RS / WS-*
  - Numerous Data Bindings: JAXB / XML Beans / Service Data Objects / JiBX
  - Flexibility of data representation: XML / JSON / FastInfoset
  - Extensible platform

* via Integration Service transport for CXF
Tooling and Management

- FuseIDE
  - Eclipse-based
  - Fuse Fabric Perspective
  - Fuse Route Editor (route constructive via “blocks and lines”)
- Management
  - Fuse Management Console
  - JBoss Operations Network (JON)

* via Integration Service transport for CXF
Why JBoss Fuse?

- Best of breed open source technologies
  - Vast open source support community
  - Red Hat certification and enterprise support
- Standards based platform
- Small, elastic footprint
  - Ability to obtain pre-packaged profiles or tailor to your needs
  - Micro-services ready
- High degree of integration capabilities out of the box
- High performance and low total cost of ownership
- OSGi provide high level of code project density
- Supports legacy technologies to enable transition

*JBoss Fuse is an End State Technology*
Beyond ESB

- Originally developed as an ESB solution due to need to support 100’s of technologies simultaneously
- OSGi is more than just modules
  - Managed configuration
  - Services
  - Managed Service Factories
  - Enables transition from legacy technology
- Next push: Eliminate “Single Purpose” technologies
  - Single SDLC – jobs, integration, services, web, BPM, cache, messaging, etc
  - A few things just need to be layered on top of Fuse to enable BPM vs using a separate technology
- Best Web Container no one is using as a web container
Best Practices

• Break out all DB access (as data sources) into separate OSGi bundles
  • One bundle per DB data source
  • Allows for granular management
  • Promotes reuse by other applications
• Register Data Sources as services in OSGi Registry
  • Decouples concrete implementation from service consumers
  • Provides for filtered-based lookups
• Take similar approach for ConnectionFactories to messaging systems
  • A-MQ / WebLogic / WebSphere MQSeries
• Utilize Built-in Property Management capabilities of Fuse ESB
  • Allows for the definition of default properties
  • Flexible means to override
  • Very useful for deployment between environments (DEV/TEST/PROD)
Demo – EIP Diagram
Extending the Integration Framework

• Despite having a robust set of components to leverage, occasionally you’ll come up short
  • Factors could include no component at all or commercial options only
• Integration Framework offers a platform that can be extended
• Key Drivers for implementing custom components
  • You’re writing and using processors a lot to bridge to a system
  • The inbound (consumer) characteristics tend to drive an event driven endpoint
• “Ground Rules”
  • Thoroughly research the availability of components
  • Cost benefit analysis of development time versus commercial component cost
  • Perform crisp due diligence on the needs to truly integration to system in question
Extending the Integration Framework

• Creating a custom component involves 4 major implementation activities
  • Component
  • Endpoint
  • Consumer
  • Producer
• Not all components will have both consumers / producers
• Additionally, if there are complex data conversion involved, consider development of a TypeConvertor to accompany your component
  • Makes use of component much easier and cleaner in routes

Disclaimer: the next series of slides is not intended to be a full overview of component development; it will cover high points and hopefully pique your interest in the power of JBoss Fuse
Key Elements of a Custom Component

Component

Endpoint

creates

Consumer

creates

Producer
**URI Notation**

activemq:topic:quotes?explicitQosEnabled=true&priority=9

<table>
<thead>
<tr>
<th>scheme</th>
<th>context</th>
<th>options</th>
</tr>
</thead>
</table>

- **Scheme**
  - Identifies the component that handles this type of endpoint
- **Context / Path**
  - Identifies the location of the source of messages
- **Options**
  - Optional details that characterize the message handling
- **You will define the URI notation for custom component**
  - May ultimately drive design decision, so it’s important to understand
Custom Component Maven Project
Select Resources