Project Fuji - Taking OpenESB to the Next Level

Andreas Egloff
Lead Architect SOA / BI
Sun Microsystems, Inc.
Overall Presentation Goal

Learn how Project Fuji combines OSGi, JBI and Maven to create a highly flexible and productive services platform
Speaker’s qualifications

- Andi is the lead architect for SOA / business integration at Sun Microsystems
- Andi is a regular presenter at JavaOne and other conferences on topics ranging from SOA and integration to Ajax Push / Comet
- Andi is a member of the JSR 312 (JBI 2.0) expert group
Where is OpenESB at Now?

- OpenESB is a platform for SOA and Integration. OpenESB is 100% Open Source and is licensed under CDDL.

  An active, responsive and growing community
e.g. mails on the users list:

  - September 2008  - 723
  - October 2008    - 1018
  - November 2008   - 1132

- Vibrant component ecosystem with components by many contributors and partners

- With enterprise quality distributions and support offerings
  - GlassFish ESB v2 Product (Brand new!)
  - Part of Sun ESB suite and Java CAPS “super suite”

- Project Fuji forms the basis of the next generation platform
Growing Component Ecosystem

- Logic and orchestration
  - BPEL, Camel, IEP, Java EE, POJO, Scripting, WLM

- Other Interfaces
  - eMail, File, FTP, HTTP, JMS, JMSJCA, Notification, Scheduler, Asterisk, Exec, HL7, LDAP, RSS, SIP, SNMP, TCP/IP, UDDI, XMPP

- EIS Interfaces
  - CICS, CORBA, DCOM, EJB, IMS, MSMQ, MQSeries, Screen Scraping, SWIFT, SAP

- Databases and data manipulation
  - Database, ETL, Data mashup, Encoding, Mural (MDM), XSLT
The Challenge...

Traditionally commercial integration and Enterprise Service Bus (ESB) platforms have a reputation of being...

big, heavy, expensive,
difficult to install,
difficult to set up,
complicated to learn and use,
a haven for lock-in
“Hello World” demo
Faces of Fuji: The Revolutionary

• Radically light weight and modular
  - Leverage OSGi at its core
• Simplify for productivity at every layer
  - New workflows and tools to define applications
• Capabilities targeted at new communities
  - Scripting, Spring, OSGi, POJO developers ...
  - Leverage from Web 2.0 applications
  - REST and Web Oriented Architecture (WOA / ROA)
  - Foundation for Cloud computing, Appliances
Faces of Fuji: The Evolutionary

• Open ESB v2 applications compatibility
  - Existing applications continue to work
  - v2 tooling can deploy to a Fuji runtime

• Leverage the OpenESB component ecosystem
  - 40+ plug-in adapters and containers and counting
  - Shared code base for components in v2 and Fuji

• Enterprise features and systemic qualities
  - Take advantage of heavy investment by many contributors in v2 and components in enterprise ready solution
  - ... but enhance and simplify how they are used
Project Fuji - Key Themes

- Innovate in
  - Ease of use
  - Agility
  - Flexibility

- All with the goal of
  - Increasing Productivity
  - Making it relevant to more technologies

- Align with GlassFish v3 application server (OSGi based)
  - Take advantage of modularity and size-to-fit
  - Easily leverage Fuji capabilities with any v3 container
Ease of Use

• Do more with simple tools; reserve sophisticated tools for real complex scenarios
  - Even productive from a text editor and command line
  - Leverage Maven support of IDEs like Netbeans, Eclipse

• Web based service composition tooling option
  - No need for a full blown IDE just to (re-)use services
  - Brings traditional integration closer to Web 2.0 development

• Domain Specific Language (DSL) for
  - Enterprise Integration Patterns
  - Integration Flow Language (IFL) defines the routing of messages between services
Web-based Composition Example
Text-based Composition

• Simple but expressive Domain Specific Language (DSL) for defining message flows
  - Integration Flow Language (IFL)
  - Makes implementing Enterprise Integration Patterns simple

• Complements richer orchestration languages
  - Where more complex composition is required

```ruby
rss  "cnn"
xmpp "im"
jruby "filter"
file "archive"

route do
  from "cnn" to "filter"
  broadcast do
    route to "im"
    route to "archive"
  end
end
```
Agility

- Reactive Runtime to increase productivity and minimize downtime
  - In-place update of artifacts resulting in dynamic service update
  - Ability to dynamically rewire services in an application
  - Dynamically add and remove interceptors at runtime

- More dynamic, more productive development
  - Goal is to eliminate the compile-deploy-test overhead
  - Deeper insight into live application to quickly see correct application behavior as the application is developed
Flexibility

• Radically light weight
  - Micro kernel architecture (~ 300 KB) that runs on any R4 OSGi container such as Felix, Knopflerfish, Equinox and including GlassFish v3
  - Can be sized to your needs
  - Features and components can be added dynamically via Update Center or OSGI repositories

• Choice of tools
  - Any IDE with Maven support (Netbeans, Eclipse)
  - Web based tooling
  - Even command line and text editor
Flexibility Continued

• Use with more technologies
  - Implement and call services in scripting languages (e.g. Groovy, JavaScript, JRuby) and additional application frameworks (Spring DM, OSGi, POJOs)
  - Leverage with any GlassFish v3 container (EJB, web...)

• Enhanced Mediation
  - Interceptors for easy and powerful aspect injection on message flows, including Security and Policy

• More topology options
  - Easy to set up, pluggable distribution mechanism
  - Stand-alone, Explicit distribution (SOA style), Transparent distribution (Federation), Homogenous clustering, Heterogeneous topologies, including linking of clusters
“Traditional Integration” Shipping Demo
Innovating At Every Layer
Peeking Under the Covers

- Everything under the covers is an OSGi bundle
  - The ESB framework itself (adds messaging layer to OSGi)
  - Every container and adapter (standard JBI component with additional OSGi entries in the manifest)
  - Every application
  - Every interceptor / aspect
  - Extensions to the framework, e.g. command line
- The user is NOT expected to touch or develop in OSGi directly – unless they want to
- OSGi is great for developing containers and frameworks, but often too involved to directly develop the applications
Summary of Project Fuji

• Build a platform people “love” to use
  • Ease of use
  • Agility
  • Flexibility

• All with the goal of
  • Increasing productivity
  • Making it relevant to more technologies, e.g. Web 2.0, POJO, Scripting developers...

• Leverage our strengths in OpenESB components, systemic qualities and enterprise features
  • By staying compatible with v2 components
Concluding statement

Project Fuji aims to simplify all aspects of developing, using and combining services, including treating external systems as services.

Help us improve it by trying it out and giving feedback.
• Screencasts, more info @ fuji.dev.java.net
Thanks for your attention!

(references)
fuji.dev.java.net
open-esb.dev.java.net
users@open-esb.dev.java.net
dev@fuji.dev.java.net
dev@fuji.dev.java.net