Tales from the front line: how ActiveMQ, ServiceMix and Camel are used to solve real world problems

Adrian Trenaman
Distinguished Consultant
http://trenaman.blogspot.com
Ade's consultancy map...
Agenda

- Brief introduction to Apache CXF, ServiceMix and ActiveMQ
  - From a functional and deployment perspective.

- Use cases and stories
  - Health-care: PDA-enable a government health system
  - Retail Pharmacy: warehouse / distribution management solution.
  - Document Storage: built on ActiveMQ
  - Telco Web Services: BSS systems

- This stuff works!
  - Highly available, clustered, flexible, extendible, ...
  - Focussed on enterprise integration & service enablement
ActiveMQ, ServiceMix, CXF & Camel
ActiveMQ


Clients connect to the broker using connectors with simple URIs.

Network connectors control how the broker interacts with other brokers for discovery, replication, failover, clustering and store & forward.

Topics and queues created dynamically.

File & JDBC-based message stores supported.
Networked brokers & distributed queues

- Brokers use network connectors to share consumer information and make routing decisions using “store-and-forward”
  - JMS clients use failover URLs or auto-discovery to connect to a live broker.

1: Producer sends a message \(m\) to the broker.

2: Broker ‘frodo’ decides to route message via broker ‘gandalf’.

3: Message is delivered.

© 2009 Progress Software Corporation
ActiveMQ replication, clustering and failover

Broker replication

Master state replicated to slave using network connector.

Shared message store

Brokers compete for file system or database lock.
Networks of master-slave pairs

- A highly available, clustered approach.
Apache ServiceMix – JBI-based integration

- A standards-based framework for deploying integration solutions.
  - Solutions are deployed as service assemblies (SAs), containing service units (SUs) that initialize service endpoints.
ServiceMix acts as a container for “components”, communicating with each other using the XML-based Normalized Message Router.

Use well-known components like JMS, HTTP, CXF, Bean, FTP, FILE, or write your own.
Build an application by configuring and wiring endpoints as SUs, combining them into SAs that can be deployed atomically.

Endpoints are configured using xbean (Spring) configuration; deployment artifacts are created using maven plugins.
JBI packaging

Each component is specialized using a SU.

In ServiceMix, the endpoint(s) are defined in an xbean.xml file.

Maven plugins are used to generate jbi.xml file.

SUs can optionally contain support classes and resources such as WSDL & XSD.

SUs are bundled together into an SA to be deployed atomically.
Hidden gem: “message flows”

- Internally, the NMR uses different approaches (“flows”) to deliver messages between components:
  - ST - straight through on the same thread
  - SEDA - intermediary in memory queue
  - JMS - using ActiveMQ
  - JCA - using XA-transactional queues with ActiveMQ

- The flow is chosen dynamically at runtime:
  - ServiceMix chooses the flow according to the quality of service of the message exchange.
Apache ServiceMix 4.0 - OSGi

- ServiceMix 4.0 provides a new OSGi-based core
  - OSGi provides a simpler deployment artifact - the bundle - with versioning, class-path isolation & service life-cycle.
  - JBI functionality is implemented as a plug-in 'feature'

- Easily deploy CXF services, Camel routes, ActiveMQ, JBI components, Spring configuration, bundles, …
ServiceMix4: OSGi-based integration
FUSE Services Framework (Apache CXF)

- Flexibly create & deploy code-first or WSDL-first Java web services.

```
Invoice inv = new Invoice();
invoiceSvc.process(inv);
```

Choose the payload (XML, SOAP, JSON) or transport (HTTP, JMS) declaratively with no impact on code.

```
@WebService
Class MyImpl implements Invoicing {
    public void process(Invoice inv) {
    }
}
```

FUSE Services Framework (Apache CXF)

- Flexibly create & deploy code-first or WSDL-first Java web services.

Choose the payload (XML, SOAP, JSON) or transport (HTTP, JMS) declaratively with no impact on code.

```
Invoice inv = new Invoice();
invoiceSvc.process(inv);
```

Separation of concerns

```
@WebService
Class MyImpl implements Invoicing {
    public void process(Invoice inv) {
    }
}
```
Cool things about FUSE Service Framework

- Code RESTful services using JAX-RS

- Generate client-side JavaScript code on the fly for browser-based WS clients:

- Deploy anywhere:
  - J2EE: WebLogic, WebSphere, Tomcat, Jboss
  - OSGi: Servicemix 4
  - JBI: ServiceMix 3
  - SpringFramework
  - Lightweight FUSE Spring Container
  - Standalone: `public static void main(...) { }`
Integration for Mobile Healthcare
Aim: expose patient care data from the national healthcare systems to health professionals using PDAs.

Key points:
- Four month project.
- Integration using SOAP/HTTP and the healthcare HL7 XML schema library.

Solution built using ServiceMix (Fuse ESB).
- Architecture: SEDA based; use embedded transactional queues to achieve synchronous, synchronous reliable, and asynchronous reliable flows.
- Standards: SOAP, JBI, HL7, XML, XSD, XSLT
- Tools: JDK, Eclipse, Maven, Subversion, Fuse ESB, CXF

Methodology: agile methods, war room, stand-ups, XP metrics, all in the context of a highly formal project methodology.
Architecture

- Integration bridge between national health-care system and mobile devices
  - Providing patient care data to medical professionals on PDA.
  - Supporting synchronous, asynchronous and reliable asynchronous message exchange patterns.
Presentation layer makes a blocking call on the integration service, which in turn blocks as it accesses the back-end.

- ... with some transformation to/from HL7 XSD using XSLT stylesheets.
- Easy peasy, lemon squeezy.
Presentation layer makes a blocking call on the ESB
- Which uses an 'asynch bridge' pattern to send a message to the back-end...
- And correlate a subsequent message from the back-end as a the response.
Unreliable? What do you mean, UNRELIABLE?

The client is blocking for a response, so
- If there's an error in the flow, then the client will receive an exception or timeout...
- ... so they can retry.
- ... so this means there's no need to make the flows persisted or transactional.
- ... hence, the term 'unreliable'

Use reliability *when you need it.*
Presentation layer submits a request to the integration service.

- Message is placed on persisted queue; ACK is returned.
- Message read off queue, request is submitted to HL7 back-end.
- Later a 'useless ACK' response is received from back-end and placed on a queue; this is logged for audit.
Lessons learnt

- Open source is geared for partnership
  - Java/XML/Maven/Spring skills readily available.
  - We had some great externals on this project.

- Agile can work well in a formal project environment
  - Need a good project manager to act as a “spring”.

- ServiceMix is solid.
  - Great performance in non-functional test: the 72 hour flat-line of joy.

- ServiceMix is not a silver bullet.
  - Some of those flows were tricky.

- Community contact is good
  - IRC, direct email
  - Don’t get distracted by the dark side of the source.
Retail Pharmacy
... actually, a warehouse scheduling problem

- How to schedule trucks and payload to increase turnaround and save on penalties.
- Application manager at warehouse sketched his design using gregor-grams on Visio.
  - ... and discovered he could implement them using ServiceMix.
- End solution was declarative configuration of EIPs.
  - No code, just config.
  - Maven, xbean, ...
The EIP solution
Aside

- We've seen Apache middleware adopted numerous times in retail sector.
  - No million-dollar budget.
  - Pragmatic, bottom-line approach.
  - Real results, real-time.
Telco
Web Services for BSS

- Mobile operator with a growing suite of web services around customer care and provisioning.
  - CXF used for Web Services
  - Deployed in Tomcat
  - Load-balanced via Apache Proxy

- >50 services:
  - Needed more control over provisioning and better security.
Proposed Architecture

Order Management Operation landscape

- OC
  - OC Web
  - HTTPS (target only TLS)

- De-millerized Zone
  - Web Presentation Layer
    - CXF client: provide caller credentials
      - Authorization & Authentication
  - Authorization & Authentication
  - User+role (envelope)

- Milllerized Zone
  - Monitoring
  - SOA CENTER
    - * delegate role
    - * load balancing
    - Firewall
    - TOMCAT
      - Jetty
      - WS (BPMs, Service Composites etc)

- Escalation
  - ORACLE
  - BSS
  - LEGACY

© 2009 Progress Software Corporation
Another telco use-case

- Implementation of Parlay-X
  - Allowing Telcos to provide services such as MMS, SMS, ALM to partners.
    - For example, how does my SMS vote for X-Factor get counted?
  - Integrates heavily with CORBA Parlay infrastructure
    - Makes use of CXF CORBA binding.

- Services deployed using OSGi framework.
Artix technology used for service infrastructure, provisioning and management.
Documents Storage
Document storage company; terabytes of data per day.

- Moving from an closed-source solution to ActiveMQ.
  - It'll scale in price, but will it scale in deployment??
- ActiveMQ put through rigorous performance tests.
  - And was selected for next generation system :)
  - Get the results from the white paper at fusesource.com.
Summary

- **ActiveMQ, CXF, ServiceMix, Camel:**
  - Focussed on messaging, integration, SOA enablement
  - Reliable, clustered, highly-available infrastructure

- **The Apache Software License is scalable.**
  - This is a major attraction for many adopters.

- **Futures**
  - Movement to OSGi runtime in SMX4
  - Supporting “SOA” & “REST” architectures