

# Red Hat JBoss Fuse

## Application Messaging and Data Integration

by

**Brad Powell**

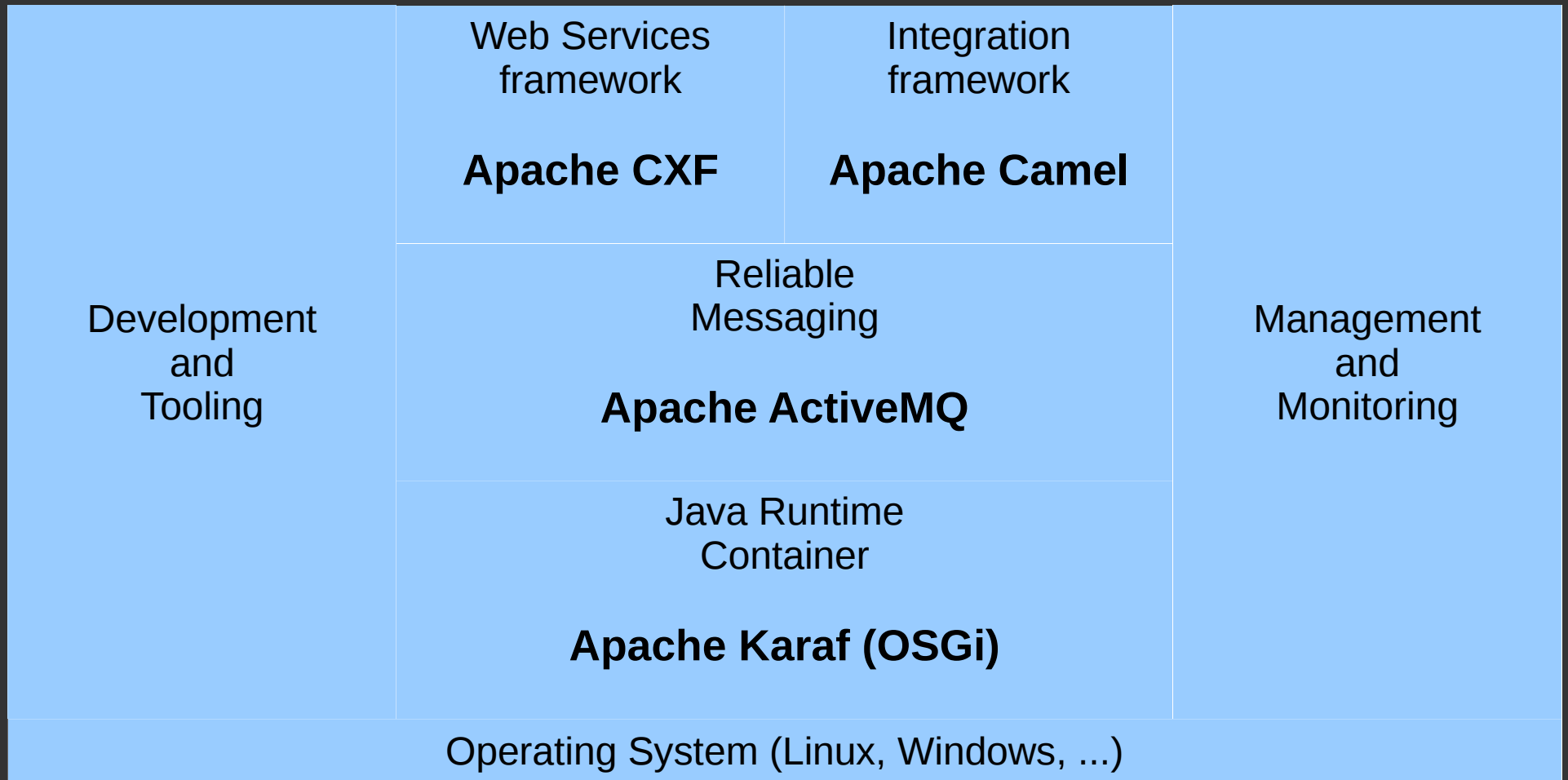
[brad.powell@spark14.com](mailto:brad.powell@spark14.com)

for

**Bartlesville User Group**

June 18<sup>th</sup>, 2015

# Fuse Overview

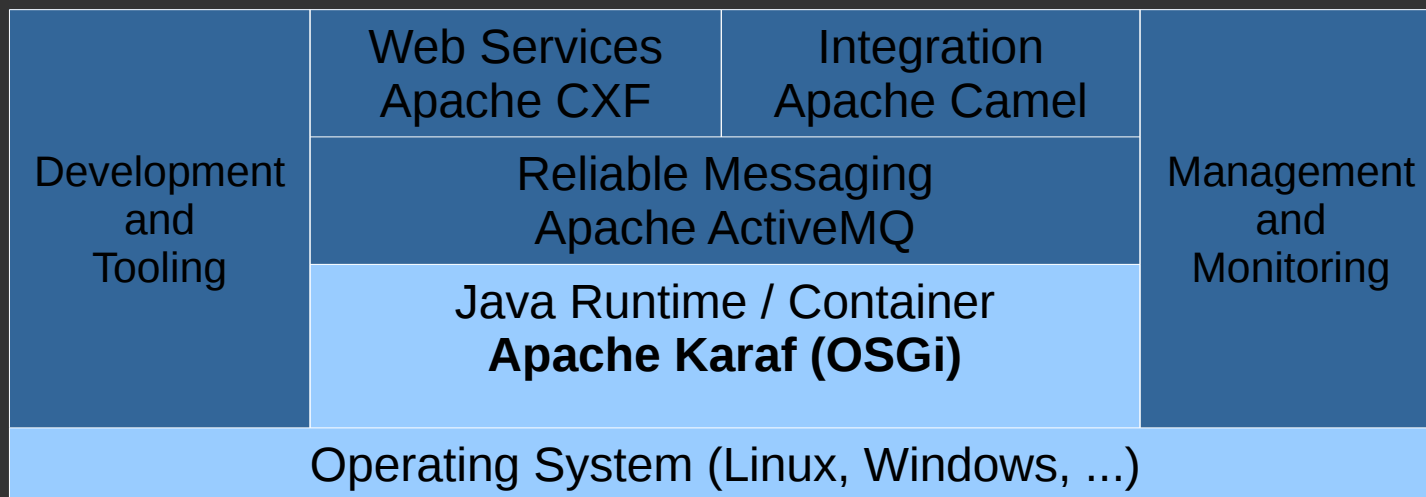


# Red Hat and Open Source

- Red Hat:
  1. Combines the open source pieces into a “release”
  2. QA's the release
  3. Offers support for the release
- Everything is fed back to the upstream open source projects.

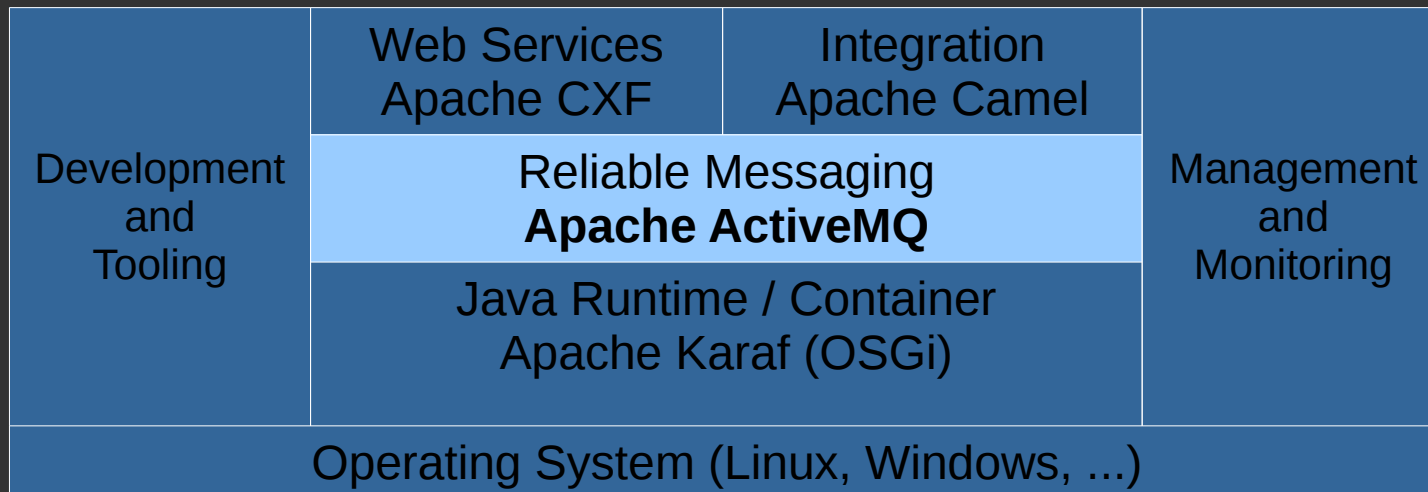
# Java

- Cross-platform, including cloud
- OSGi (Apache Karaf)
  - Dynamic configuration
  - Deploy/update modules live



# ActiveMQ

- Messaging server
- aka JBoss A-MQ

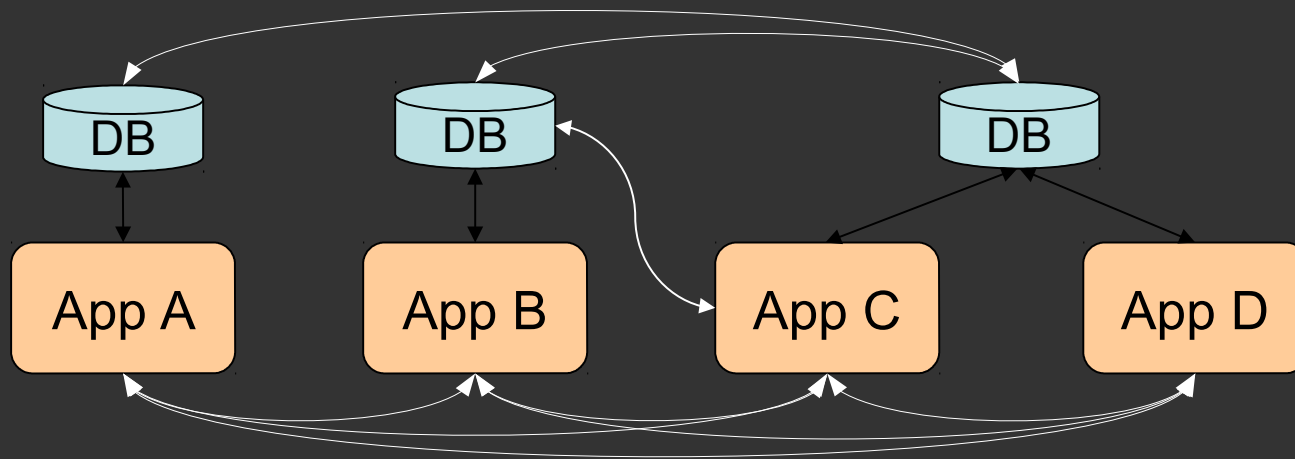


*detour...*

Why Messaging?

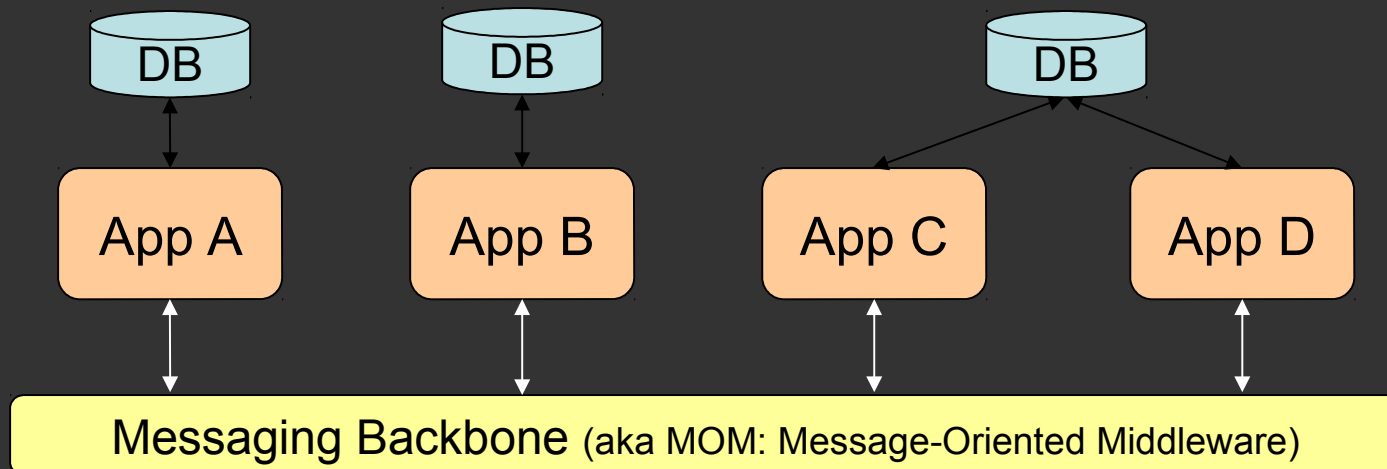
# Traditional Point-to-Point Integration

- Synchronous
- RPC
- Application guarantees delivery
- Tightly-coupled



# Integration with Messaging

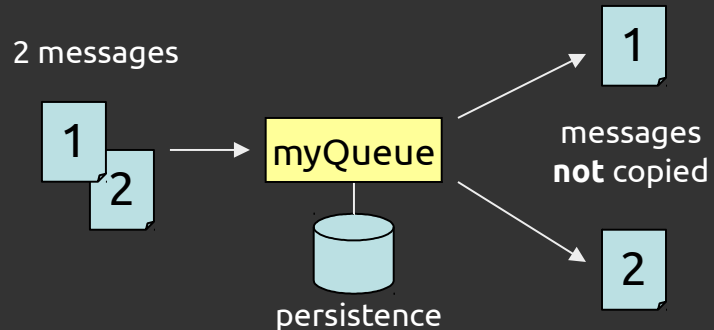
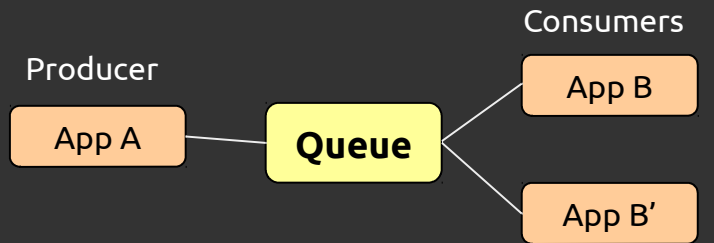
- Asynchronous, event-driven
- Fire-and-forget
- Guaranteed delivery
- Loosely-coupled



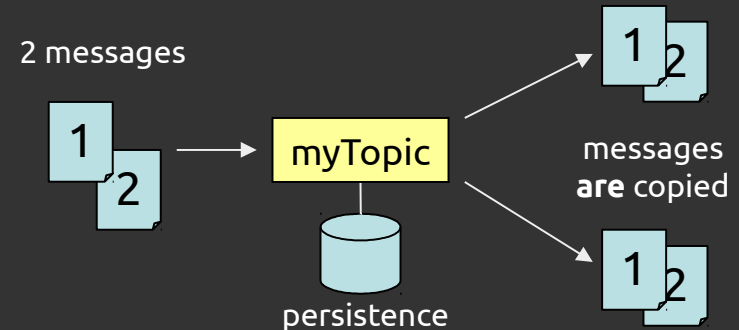
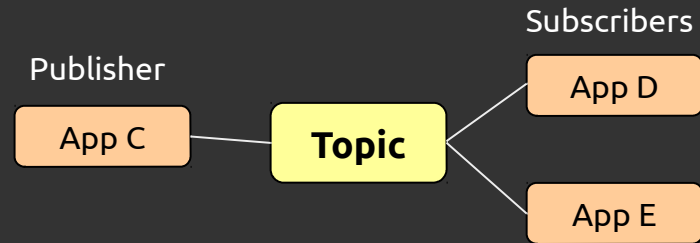


# Messaging Semantics

## Queues: point-to-point



## Topics: publish/subscribe

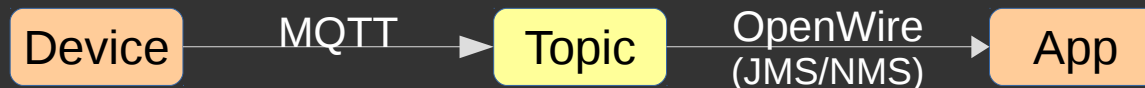


*...back to Fuse*

# ActiveMQ

## Protocols & Transports

- Multi-protocol
  - Default:
    - OpenWire over TCP/SSL/HTTP(S)/UDP/IP
  - Also:
    - STOMP, AMQP (RabbitMQ), MQTT, REST
  - Can *bridge* protocols



- WebSocket
  - STOMP and MQTT over WebSockets

# ActiveMQ Clients

- Multi-platform
  - Clients for just about any platform
    - Native:
      - JMS, .NET, C++
    - Other:
      - JavaScript, Ruby, Perl, Python, PHP, Smalltalk, ...

# ActiveMQ

## Code Example – Java/JMS

<http://activemq.apache.org/hello-world.html>

```
// Create a ConnectionFactory
ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory("vm://localhost");

// Create a Connection
Connection connection = connectionFactory.createConnection();
connection.start();

// Create a Session
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

// Create the destination (Topic or Queue)
Destination destination = session.createQueue("TEST.FOO");

// Create a MessageProducer from the Session to the Topic or Queue
MessageProducer producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

// Create a messages
String text = "Hello world! From: " + Thread.currentThread().getName() + " : " + this.hashCode();
TextMessage message = session.createTextMessage(text);

// Tell the producer to send the message
System.out.println("Sent message: " + message.hashCode() + " : " + Thread.currentThread().getName());
producer.send(message);

// Clean up
session.close();
connection.close();
```

# ActiveMQ

## Code Example – .NET/NMS

<http://activemq.apache.org/nms/nms-examples.html>

```
Uri connecturi = new Uri("activemq:tcp://activemqhost:61616");
IConnectionFactory factory = new NMSConnectionFactory(connecturi);

using(IConnection connection = factory.CreateConnection())
using(ISession session = connection.CreateSession())
{
    IDestination destination = SessionUtil.GetDestination(session, "queue://FOO.BAR");

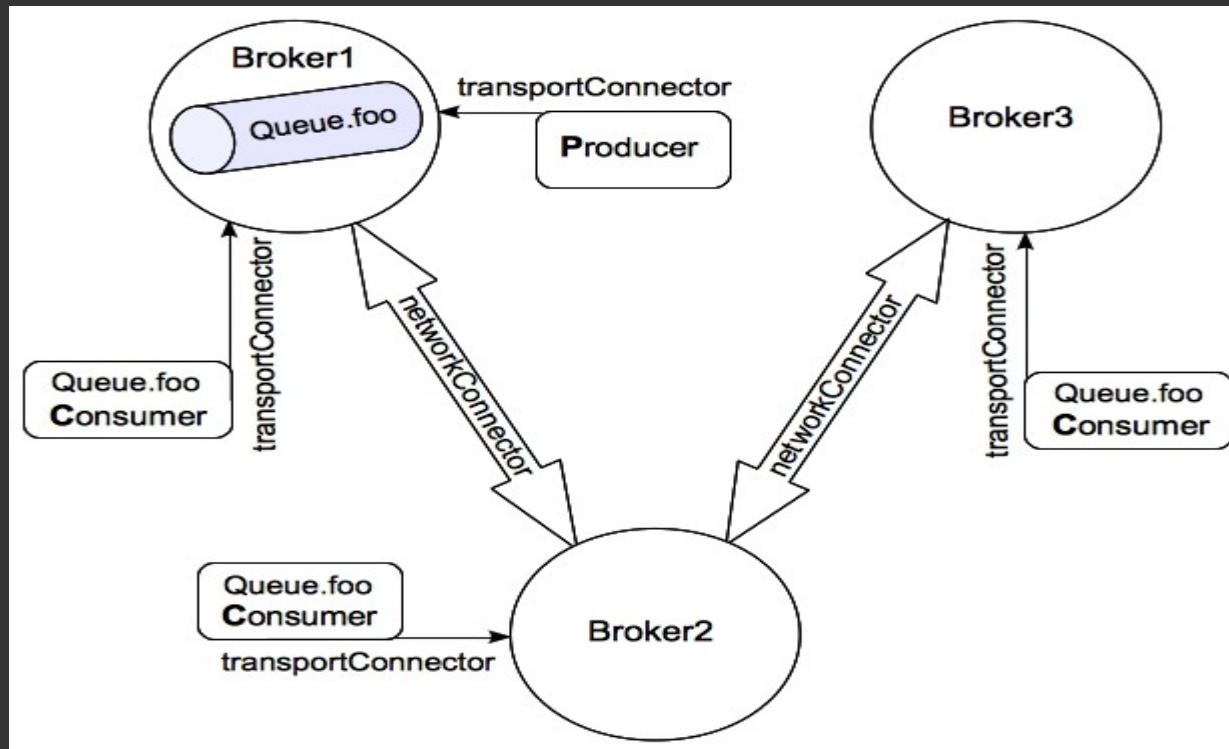
    // Create a consumer and producer
    using(IMessageConsumer consumer = session.CreateConsumer(destination))
    using(IMessageProducer producer = session.CreateProducer(destination))
    {
        // Start the connection so that messages will be processed.
        connection.Start();
        producer.DeliveryMode = MsgDeliveryMode.Persistent;

        // Send a message
        ITextMessage request = session.CreateTextMessage("Hello World!");
        request.NMSCorrelationID = "abc";
        request.Properties["NMSXGroupID"] = "cheese";
        request.Properties["myHeader"] = "Cheddar";
        producer.Send(request);

        // Consume a message
        ITextMessage message = consumer.Receive() as ITextMessage;
        if(message == null) {
            Console.WriteLine("No message received!");
        }
        else {
            Console.WriteLine("Received message with ID: " + message.NMSMessageId);
            Console.WriteLine("Received message with text: " + message.Text);
        }
    }
}
```

# ActiveMQ Architectures

- Simple, stand-alone
- Master/Slave, with failover protocol
- Network-of-Brokers:



ActiveMQ

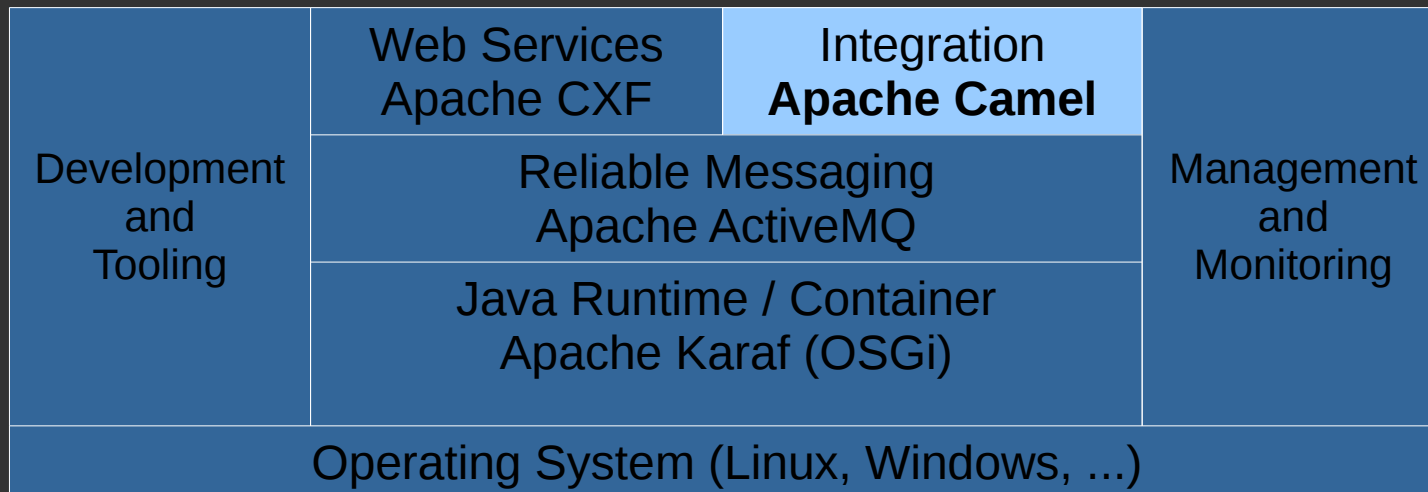
# Marketplace Comparison

- RabbitMQ (AMQP)
- MSMQ
- Mosquitto (MQTT)
- Amazon SQS
- Kafka
- ZeroMQ (library)
- ... many others



# Camel

- Data integration framework

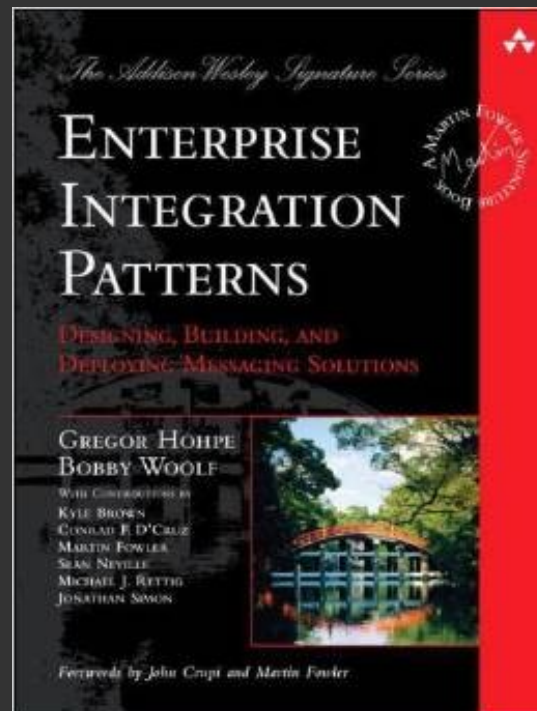


# Camel Overview

- Lightweight and embeddable
- Components (i.e. connectors)
  - 150+ out-of-the-box components
  - The usual:
    - DB, File, (S)FTP, HTTP(S), JMS, etc.
  - And many more:
    - SAP, SalesForce, HDFS, HBase, MongoDB, Dropbox, GMail, Facebook, Twitter, ...
- Domain Specific Language
  - Based on **Enterprise Integration Patterns** (EIPs)
  - DSL implementations: Java, XML, Scala, ...




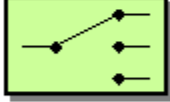
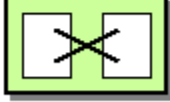

# Camel EIPs

- <http://www.enterpriseintegrationpatterns.com>
- Book by Gregor Hohpe and Bobby Woolf



# Camel EIPs

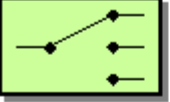

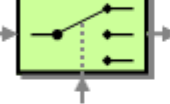
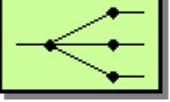


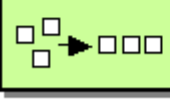
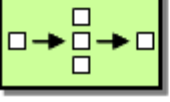
## Messaging Systems

	Message Channel
	Message
	Pipes and Filters
	Message Router
	Message Translator
	Message Endpoint

## Messaging Channels

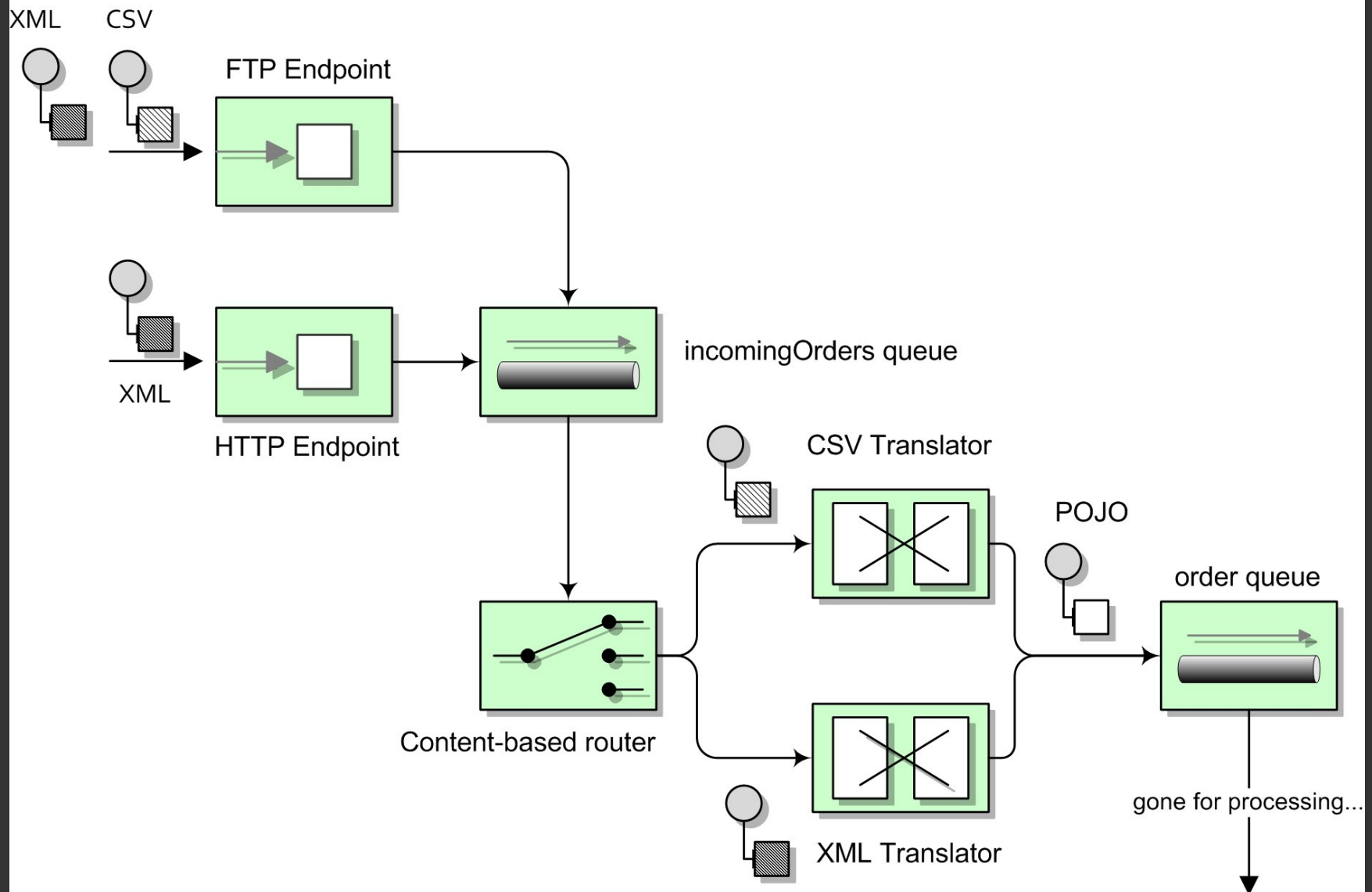
	Point to Point Channel
	Publish Subscribe Channel
	Dead Letter Channel
	Guaranteed Delivery
	Message Bus

## Message Routing

	Content Based Router
	Message Filter
	Dynamic Router
	Recipient List
	Splitter
	Aggregator
	Resequencer
	Composed Message Processor

...and more patterns (65 in total)

# Camel EIP Example



# Camel

## Code Examples

- Java DSL (fluent)

```
from ("file:/tmp").to("jms:aQueue");
```

- Spring XML DSL

```
<route>  
  <from uri="file:/tmp"/>  
    <to uri="jms:aQueue"/>  
</route>
```

- Scala DSL

```
from "file:/tmp" -> "jms:aQueue"
```

# Camel

## Code Examples (cont.)

- Java DSL, with content-based routing (<http://camel.apache.org/java-dsl.html>)

```
// Process input files, leaving them in place ('noop' flag), then  
// perform content-based-routing on the message using XPath.
```

```
from("file:src/data?noop=true")  
    .choice()  
        .when(xpath("/person/city = '&#39;London&#39;"))  
            .to("file:target/messages/uk")  
        .otherwise()  
            .to("file:target/messages/others");
```

Camel

# “Framework vs Platform”

- Camel is an integration “framework”, not a full integration platform, like Talend, Mule, TIBCO, Informatica, PI, BizTalk, etc.
  - *Although*, when combined with all of the associated tooling, such as A-MQ, CXF, Hawtio, JBoss Studio, etc., it’s more-or-less a platform.
- Skills: Java, Maven, Spring
  - Good, because finding “typical” Java developers tends to be easier than finding specialists in the other integration platforms.

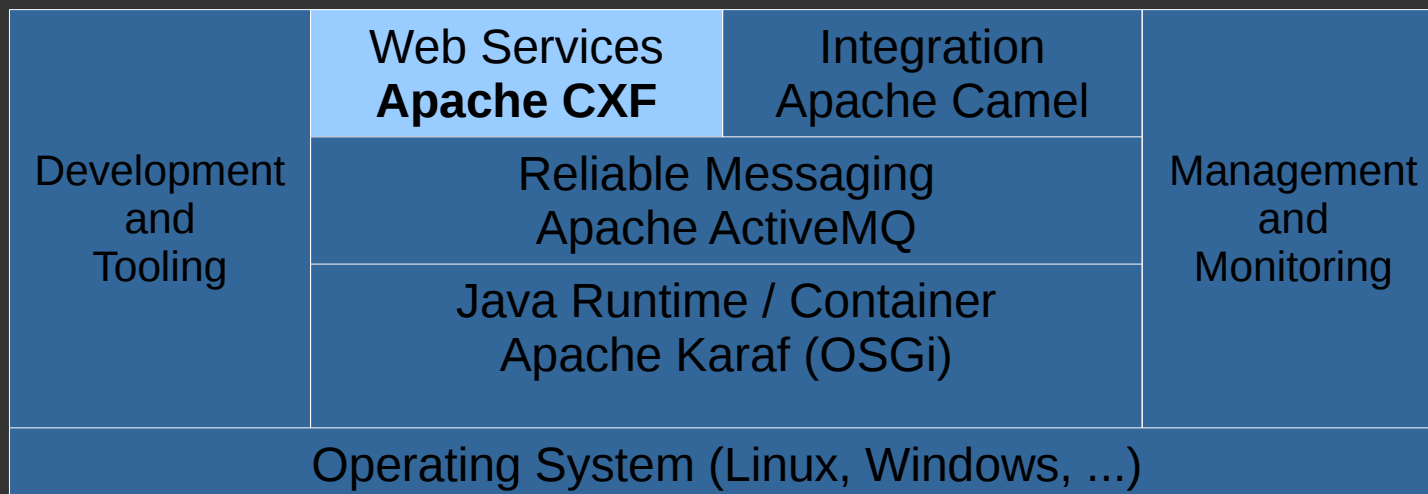


# Camel Marketplace Comparison

- Java
  - Spring Integration
- .NET
  - NServiceBus
  - MassTransit

# CXF

- Services framework



# CXF

## Overview

- Build services using front-end APIs, such as:
  - JAX-WS
  - JAX-RS
- Protocols and transports
  - SOAP
  - XML/HTTP
  - RESTful HTTP
  - JMS
  - ...more

(side note)

## Camel REST DSL

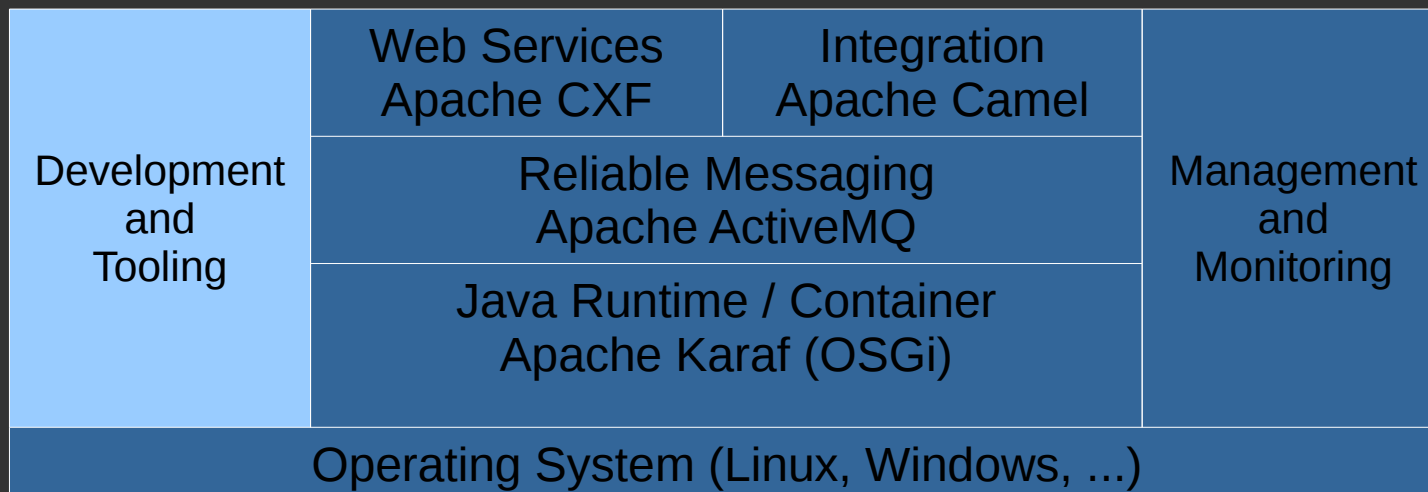
- As an alternative to using CXF for RESTful services, Camel now includes a REST DSL

e.g.

```
rest("/customers/")  
    .get("/{id}").to("direct:customerDetail")  
    .get("/{id}/orders").to("direct:customerOrders")  
    .post("/neworder").to("direct:customerNewOrder");
```

# Development and Tooling

- Text editor
- Eclipse IDE
  - JBoss Developer Studio
- Typical Java build/CI tools (Git, Maven, Jenkins)



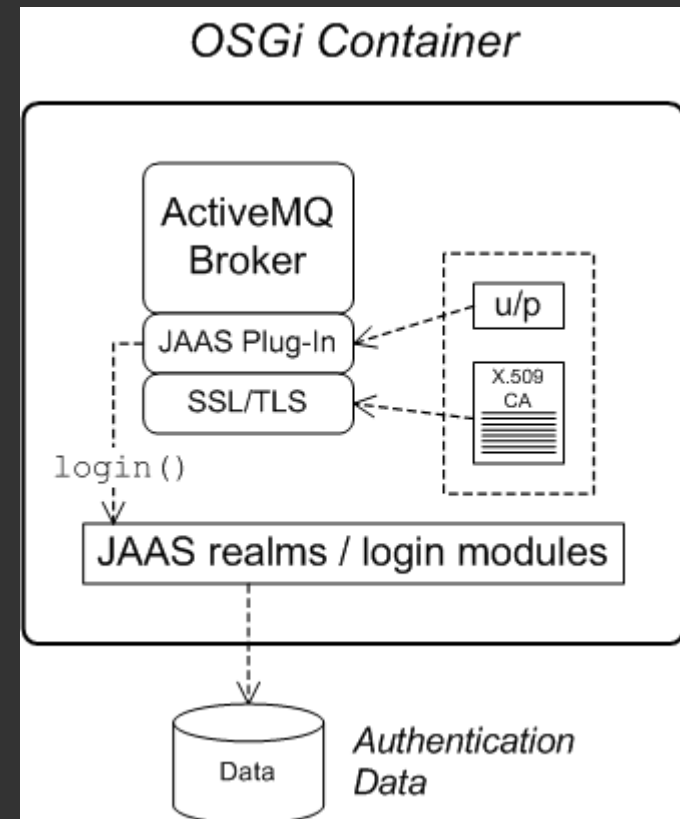
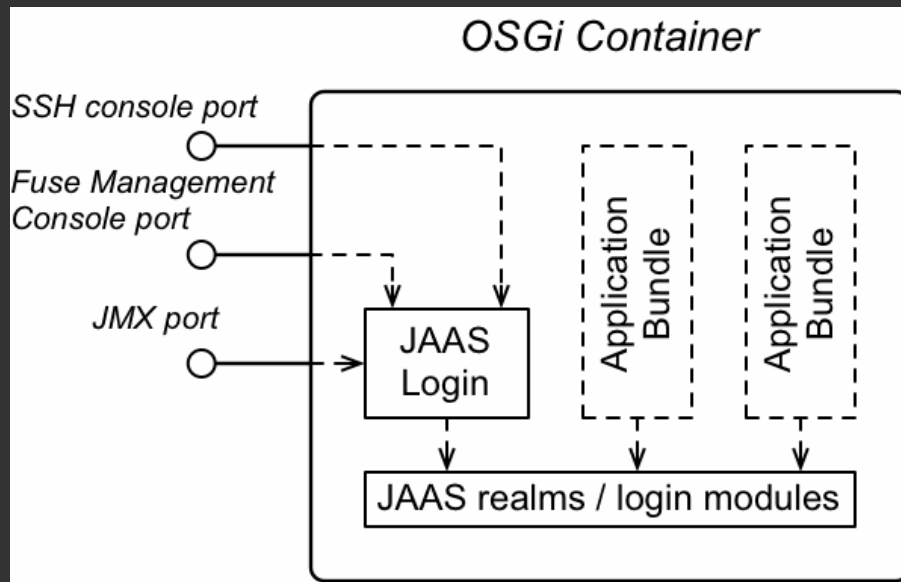
# Management and Monitoring

- JMX
- Hawtio console
- JBoss Operations Network

Development and Tooling	Web Services Apache CXF	Integration Apache Camel	Management and Monitoring
	Reliable Messaging Apache ActiveMQ		
	Java Runtime / Container Apache Karaf (OSGi)		
Operating System (Linux, Windows, ...)			

# Security

- [https://access.redhat.com/documentation/en-US/Red\\_Hat\\_JBoss\\_Fuse/6.1/html/Security\\_Guide](https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/6.1/html/Security_Guide)
- JAAS: Java Authentication and Authorization Service



# Misc

- Message payloads
  - JMS/NMS have common types (binary, text, etc.)
  - MQTT is binary
- BPM/BRMS
  - Often used in conjunction with integration/messaging systems
- Links
  - [https://access.redhat.com/documentation/en-US/Red\\_Hat\\_JBoss\\_Fuse/6.1/](https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Fuse/6.1/)
  - <https://github.com/FuseByExample>
  - <http://activemq.apache.org/>
  - <http://camel.apache.org/>



Questions?