Microservices with Red Hat JBoss Fuse

Alexey Brook
CTO, Head of Integration & SOA Center of Excellence
Matrix
Agenda

- Who we are – Integration and SOA Center of Excellence
- Microservices – the right way to do SOA
- Why ESBs are still around
- JBoss Fuse: benefits for Microservices with JBoss Studio, CXF, Camel, and Fabric8
- Red Hat Microservices platform – end to end
MATRIX INTEGRATION AND SOA COE
Matrix CoE Concept

- Non-Functional
- Software
- Consulting
- Experience
- Architects
- Business Partner
- Expertise, Knowledge
- Delivery

Center Of Excellence
Red Hat Partnership

RED HAT
PREMIER BUSINESS PARTNER

MATRIX IT LTD
ISRAEL

TERM OF VALIDITY: 5/20/2016 - 5/19/2017

PETRA HEINRICH
VP, Partners & Alliances EMEA
MICROSERVICES – THE RIGHT WAY TO DO SOA
Microservice definition

- **Microservices** is an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other modules (What.com).
Microservice definition

- **Microservices** is an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other modules (What.com).

- **Microservices** is a specialization of and implementation approach for service-oriented architectures (SOA) used to build flexible, independently deployable software systems (Wikipedia).
Microservice definition

- **Microservices** is an approach to application development in which a large application is built as a suite of modular services. Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other modules (What.com).

- **Microservices** is a specialization of and implementation approach for service-oriented architectures (SOA) used to build flexible, independently deployable software systems (Wikipedia).

- The common definition of **microservices** generally relies upon each microservice providing an API endpoint, often but not always a stateless REST API which can be accessed over HTTP(S) just like a standard webpage (opensource.com).
Service Oriented Concepts

- Independent, reusable, “atomic” building blocks
- Decoupled architecture
- Standard protocols and integration techniques
- Management / Governance
- Centralized integration efforts, let service be “business services”
### Microservices – pragmatic SOA

<table>
<thead>
<tr>
<th>Independent</th>
<th>Manageable</th>
<th>Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure</td>
<td>Reusable</td>
<td>Atomic</td>
</tr>
<tr>
<td>Decoupled</td>
<td></td>
<td>Standard</td>
</tr>
<tr>
<td>Integrated</td>
<td></td>
<td>Scalable</td>
</tr>
<tr>
<td>Testable</td>
<td></td>
<td>Simple</td>
</tr>
</tbody>
</table>
WHY ESBS ARE STILL AROUND?
Generic application architecture

- Web Client
- Mobile Apps
- Partners Apps

JS Frameworks
AJAX Layer

APIs Layer (API Management)

DB

Composite Services
Service proxies

Micro-Services

services platform

micro-services on the platform ? !!
Microservices platform

- Service Contract – defined operations (WSDL, Swagger)
- Service Binding – use standards (SOAP, REST, JMS, …)
- Service Implementation – patterns, faults, business
- Service lifecycle – testing, packaging, deployment
- Testing, visibility, security, …
Microservices platform

- Service Contract – defined operations (WSDL, Swagger)
- Service Binding – use standards (SOAP, REST, JMS, …)
- Service Implementation – patterns, faults, business
- Service lifecycle – testing, packaging, deployment
- Testing, visibility, security, …
- You can code it all, or …
Microservices platform

- Service Contract – defined operations (WSDL, Swagger)
- Service Binding – use standards (SOAP, REST, JMS, …)
- Service Implementation – patterns, faults, business
- Service lifecycle – testing, packaging, deployment
- Testing, visibility, security, …
- You can code it all, or …

Use JBoss Fuse to do the job
JBoss Fuse: Benefits for Microservices
JBoss Fuse (v6.3) Components

- **JBoss Dev Studio** 8.x-10.x - Makes things visual
- **Apache Camel** 2.17.0 - Bindings, routing, DSLs
- **Apache CXF** 3.1.5 - SOAP/REST full-featured
- **Fabric8** 1.2 - Platform management
- **Apache ActiveMQ** 5.11.0 - JMS, supported in Camel
- **SwithYard** 2.1.0 - SCA implementation
- **Apache Karaf** 2.4 - OSGi container

+ **JBoss EAP** as your JEE Application Server
Visual debugger for Camel routes:
- Break points
- Tracing
Visual data mapping, included in Camel routes
- XML / JSON / Java
- Custom formats
Service implementation

- Apache Camel framework is fully supported within Fuse product.
- Camel itself is a powerful integration framework, and allows to define full service implementations using Java language.
- Camel users Java/XML to define a “map” between interfaces and implementations.

Camel Java DSL

```java
import org.apache.camel.builder.RouteBuilder;

public class MyRoute extends RouteBuilder {
    
    public void configure() throws Exception {
        from("activemq:queue:newOrder")
            .choice()
                .when(xpath("/order/product = 'widget'"))
                    .to("activemq:queue:widget")
                .otherwise()
                    .to("activemq:queue:gadget")
            .end();
    }
}
```

Camel XML DSL

```xml
<route>
    <from uri="activemq:queue:newOrder"/>
    <choice>
        <when>
            <xpath>/order/product = 'widget'</xpath>
            <to uri="activemq:queue:widget"/>
        </when>
        <otherwise>
            <to uri="activemq:queue:gadget"/>
        </otherwise>
    </choice>
</route>
```

Camel can use several languages to define routes and patterns.

Service logic can be provided with POJOs and/or Spring beans.
Camel java DSL fully supports REST/JSON bindings in Java code, and allows to simply route HTTP API to the Java implementation.

And also helps to generate Swagger documentation.
Service Bindings - REST

And similar with XML DSL
Using CXF & Maven with “contract-first” archetype allows to automatically generate endpoint definition from a WSDL.

```xml
<cxf:cxfEndpoint id="reportEndpoint" address="/report/
    serviceClass="org.apache.camel.example.reportincident.ReportIncidentEndpoint"
    wsdlURL="wsdl/report_incident.wsdl"/>
```
Service deployment and scaling

- Microservices are a system backbone, behind the APIs and GUI – scalable environment is a must.
- **Docker** container with **OpenShift** provides a manageable environment for JBoss products.
- JBoss Fuse (with JBoss EAP) images can be managed using Fabric8 and Kubernetes
- Integrated Fabric8 makes it simple to manage large and distributed JBoss Fuse deployments
RED HAT MICROSERVICES PLATFORM – END TO END
Generic application architecture

Web Client

Mobile Apps

Partners Apps

JS Frameworks
AJAX Layer

APIs Layer (API Management)

DB

Composite Services

Service proxies

Micro-Services

services platform
RED HAT based architecture

Web Client

Mobile Apps

Partners Apps

JS Frameworks
AJAX Layer

3scale API Management

RED HAT
Linux
OpenShift

Camel

RED HAT JBoss Fuse

CXF

runs on JBoss EAP

Micro-Services

JBoss Fuse gives us an end to end service platform
The Microservices principles are new, and better pattern for doing SOA the right way. And RED HAT products can give you an end to end platform for Microservices implementation, management and run-time.
Alexey Brook
Head of Integration & SOA Center of Excellence
alexeybr@matrix.co.il

Thank You