RED HAT
FORUM
Europe, Middle East & Africa

Microservices with Red 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

RED HAT
FORUM
Europe, Middle East & Africa

MATRIX INTEGRATION AND SOA COE

RED HAT FORUM
Europe, Middle East & Africa

redhat.

Matrix CoE Concept

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

Center Of Excellence

RED HAT FORUM
Europe, Middle East & Africa

Integration & SOA CoE

redhat.

Services Security	Services DevOps & Testing	ESB, BPM, SRR, BRM, APIM, CEP, …	Topology, Federation, Clustering
Service platforms	Architecture	Java and Tools	Planning and Scoping

Integration, Middleware, SOA

RED HAT FORUM
Europe, Middle East & Africa

Red Hat Partnership

redhat.

matrix Integration & SOA
Center of Excellence

RED HAT
FORUM
Europe, Middle East & Africa

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”

RED HAT FORUM
Europe, Middle East & Africa

Microservices – pragmatic SOA

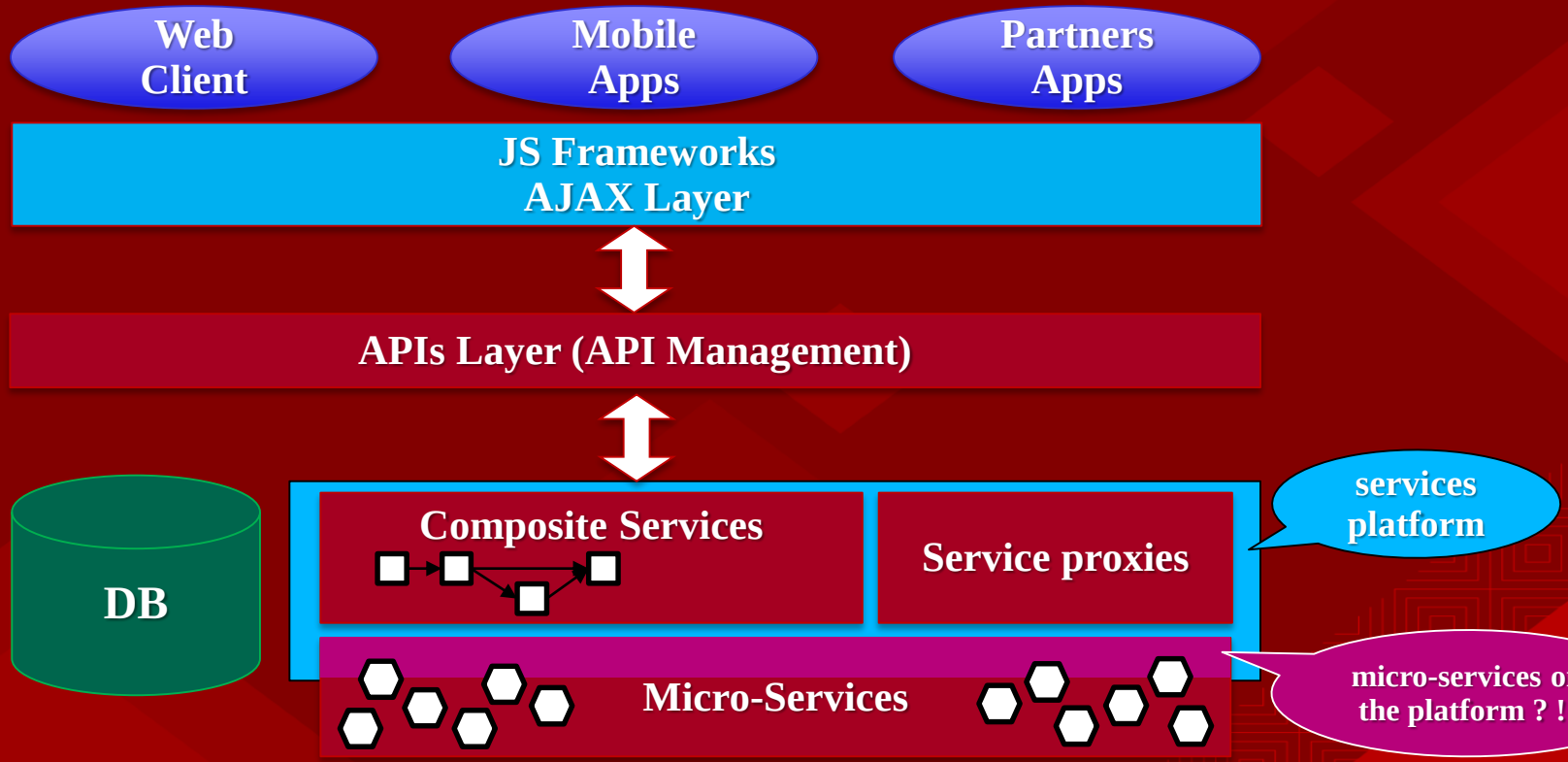
redhat.

		Manageable			
Independent					Small
	Secure		Reusable		
				Atomic	
		Decoupled			
	Integrated				Standard
Testable				Scalable	
			Simple		

So…

WHY ESBS ARE STILL AROUND?

Generic application architecture



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 …

RED HAT FORUM
Europe, Middle East & Africa

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

RED HAT
FORUM
Europe, Middle East & Africa

JBOSS FUSE: BENEFITS FOR MICROSERVICES

JBoss Fuse (v6.3) Components

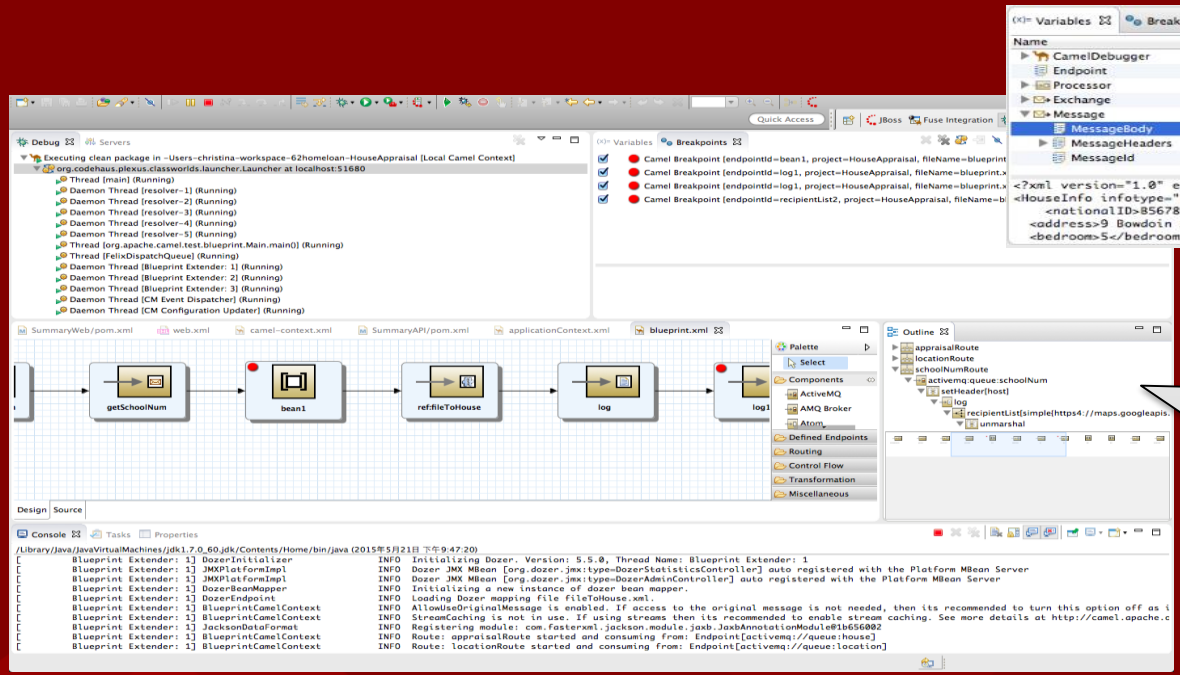
- **JBoss Dev Studio** **8.x-10.x** **- Makes things visual**
- **Apache Camel** **2.17.0** **- Bindings, routing, DSLs**
- **Apache CXF** **3.1.5** **- SOAP/REST full-featured**
- **Fabric8** **1.2** **- Platform management**
- Apache ActiveMQ 5.11.0 - JMS, supported in Camel
- SwithYard 2.1.0 - SCA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

+ **JBoss EAP** as your JEE Application Server

RED HAT FORUM
Europe, Middle East & Africa

JBoss Fuse visual development

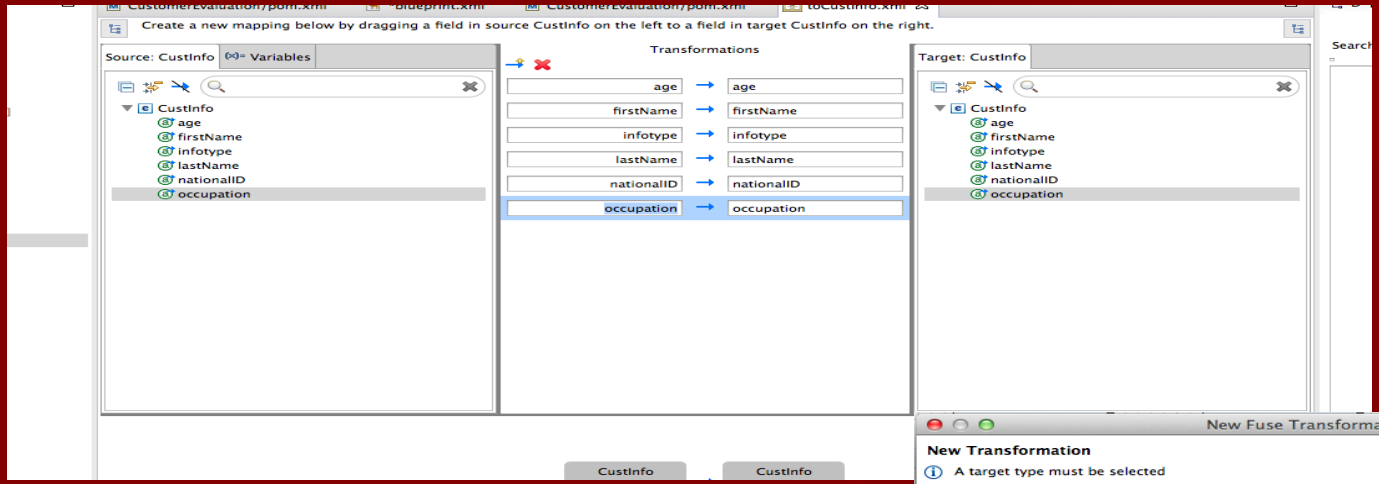
redhat.

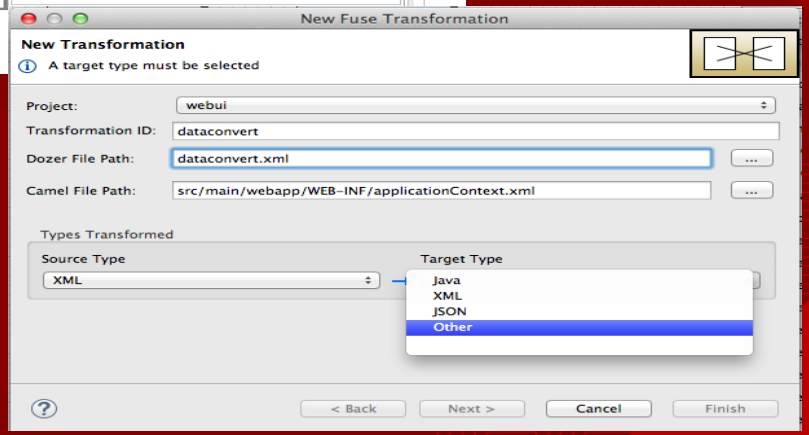


Visual debugger for Camel routes:

- **Break points**
- **Tracing**

JBoss Fuse visual development





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

```
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

```
<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

Service Bindings - REST


```java
@Override
public void configure() throws Exception {

    restConfiguration().component("netty4-http").bindingMode(RestBindingMode.json)
        .dataFormatProperty("prettyPrint", "true")
        .contextPath("/").port(8080);

    rest("/user")
        .consumes("application/json").produces("application/json")

        .get("/{id}")
            .to("bean:userService?method=getUser(${header.id})")

        .put()
            .to("bean:userService?method=updateUser")

        .get("/findAll")
            .to("bean:userService?method=listUsers");
}
```

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

```java
.get("/{id}").description("Find user by id").outType(User.class)
    .param().name("id").type(path).description("The id of the user to get").data
    .to("bean:userService?method=getUser(${header.id})")

.put().description("Updates or create a user").type(User.class)
    .param().name("body").type(body).description("The user to update or create")
    .to("bean:userService?method=updateUser")

.get("/findAll").description("Find all users").outTypeList(User.class)
    .to("bean:userService?method=listUsers");
```

Service Bindings - REST

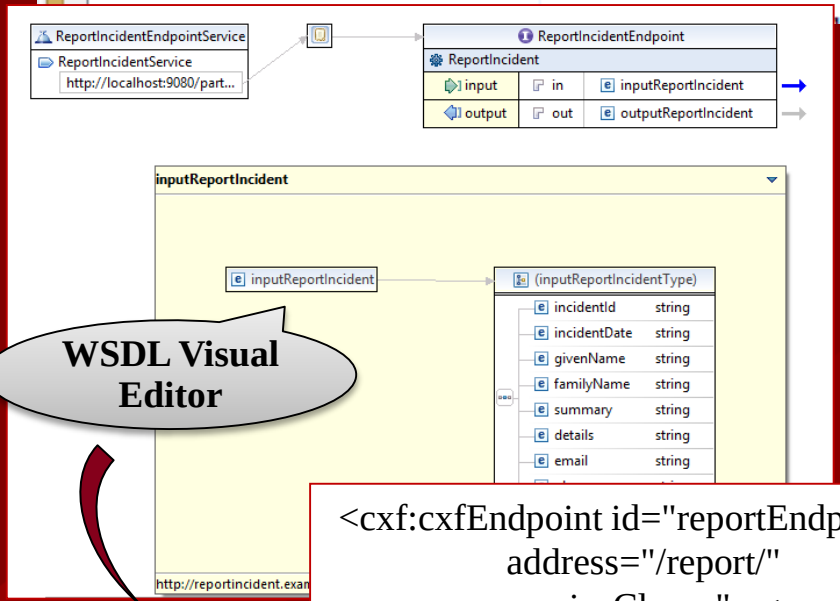

```xml
<?xml version="1.0" encoding="UTF-8"?>

<camelContext xmlns="http://camel.apache.org/schema/spring">
  <rest path="/say">
    <get uri="/hello">
      <to uri="direct:hello"/>
    </get>
    <get uri="/bye" consumes="application/json">
      <to uri="direct:bye"/>
    </get>
    <post uri="/bye">
      <to uri="mock:update"/>
    </post>
  </rest>
  <route>
    <from uri="direct:hello"/>
    <transform>
      <constant>Hello World</constant>
    </transform>
  </route>
  <route>
    <from uri="direct:bye"/>
    <transform>
      <constant>Bye World</constant>
    </transform>
  </route>
</camelContext>
```

And similar with XML DSL

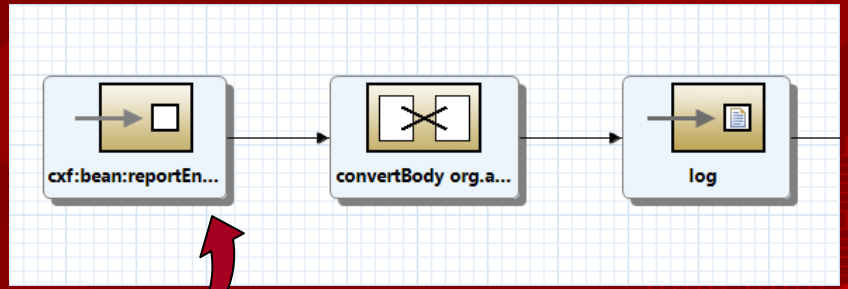
Service Contracts

org.apache.camel.archetypes	camel-archetype-cxf-code-first-blueprint	2.15.1.redhat-621084
org.apache.camel.archetypes	camel-archetype-cxf-contract-first-blueprint	2.15.1.redhat-
org.apache.camel.archetypes	camel-archetype-java	2.15.1
org.apache.camel.archetypes	camel-archetype-spring	2.15

Using CXF & Maven with "contract-first" archetype allows to automatically generate endpoint definition from a WSDL



WSDL Visual Editor




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

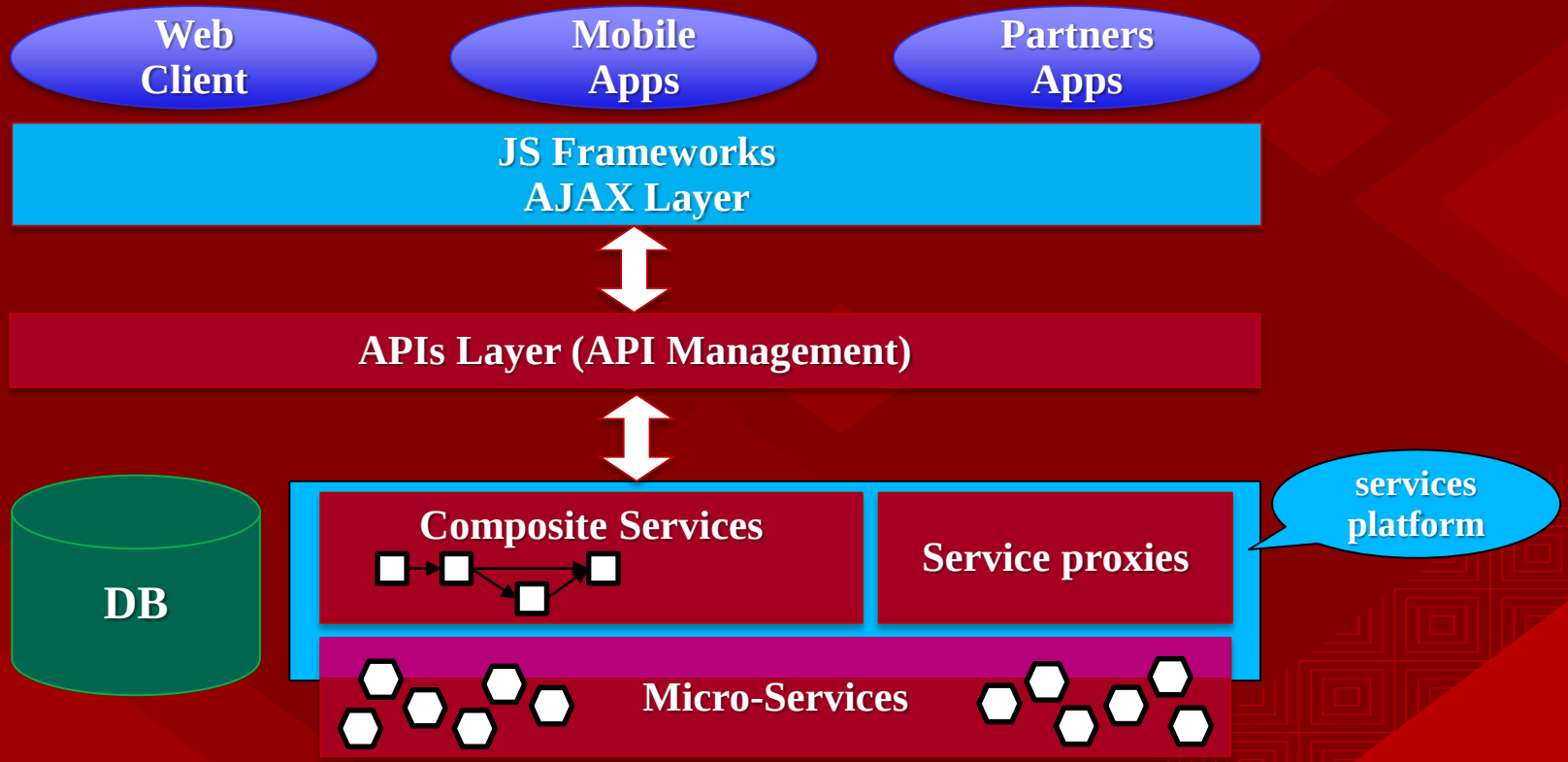
Service deployment and scaling

- 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
FORUM
Europe, Middle East & Africa

RED HAT MICROSERVICES PLATFORM – END TO END

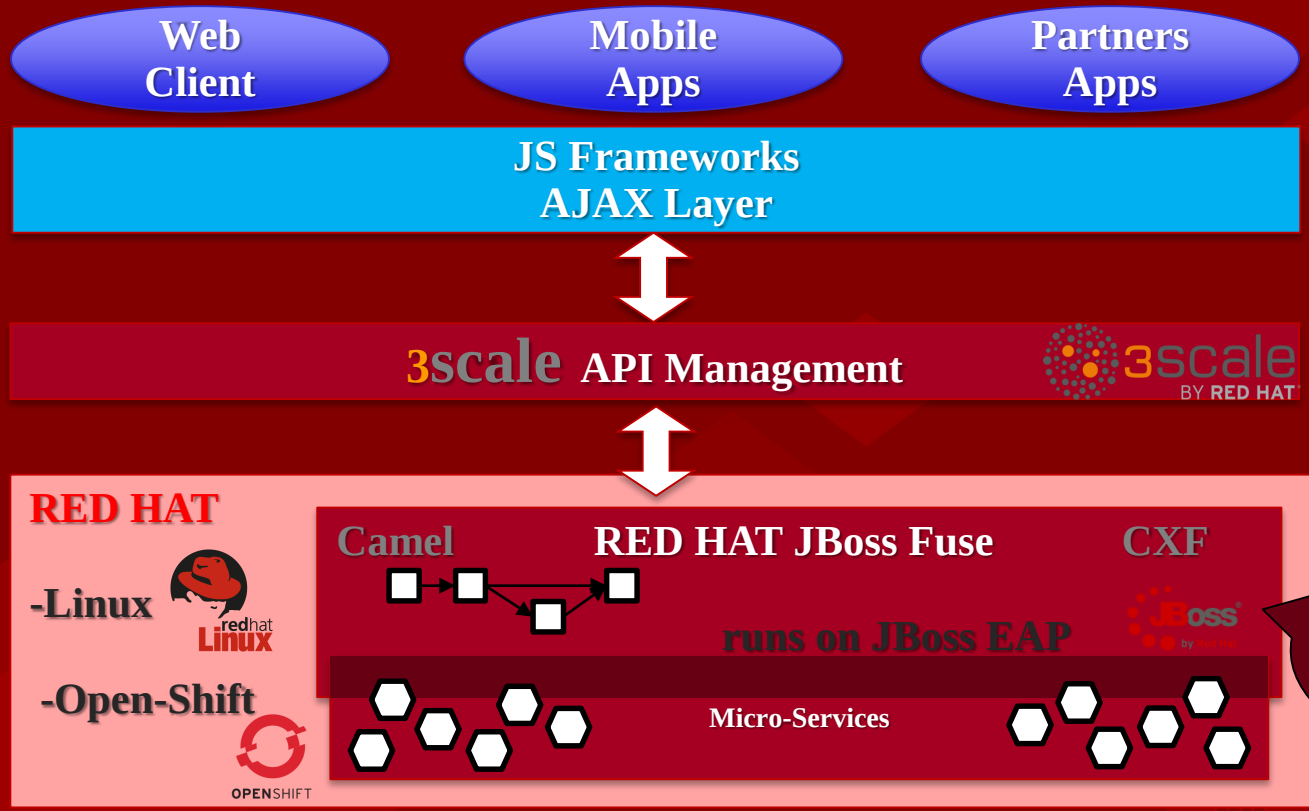
Generic application architecture



RED HAT FORUM
Europe, Middle East & Africa

RED HAT based architecture

redhat.



JBoss Fuse gives us an end to end service platform

RED HAT FORUM
Europe, Middle East & Africa

redhat.

Summary

*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

redhat®

Thank You