

# 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

# MATRIX INTEGRATION AND SOA COE

# Matrix CoE Concept

Non-  
Functional

Software

Consulting

Experience

Architects

Business  
Partner

Expertise,  
Knowledge

Delivery

Center Of Excellence

# Integration & SOA CoE

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 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



**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”



# Microservices – pragmatic SOA

		<b>Manageable</b>			
<b>Independent</b>					<b>Small</b>
	<b>Secure</b>		<b>Reusable</b>		
				<b>Atomic</b>	
		<b>Decoupled</b>			
	<b>Integrated</b>				<b>Standard</b>
<b>Testable</b>				<b>Scalable</b>	
			<b>Simple</b>		



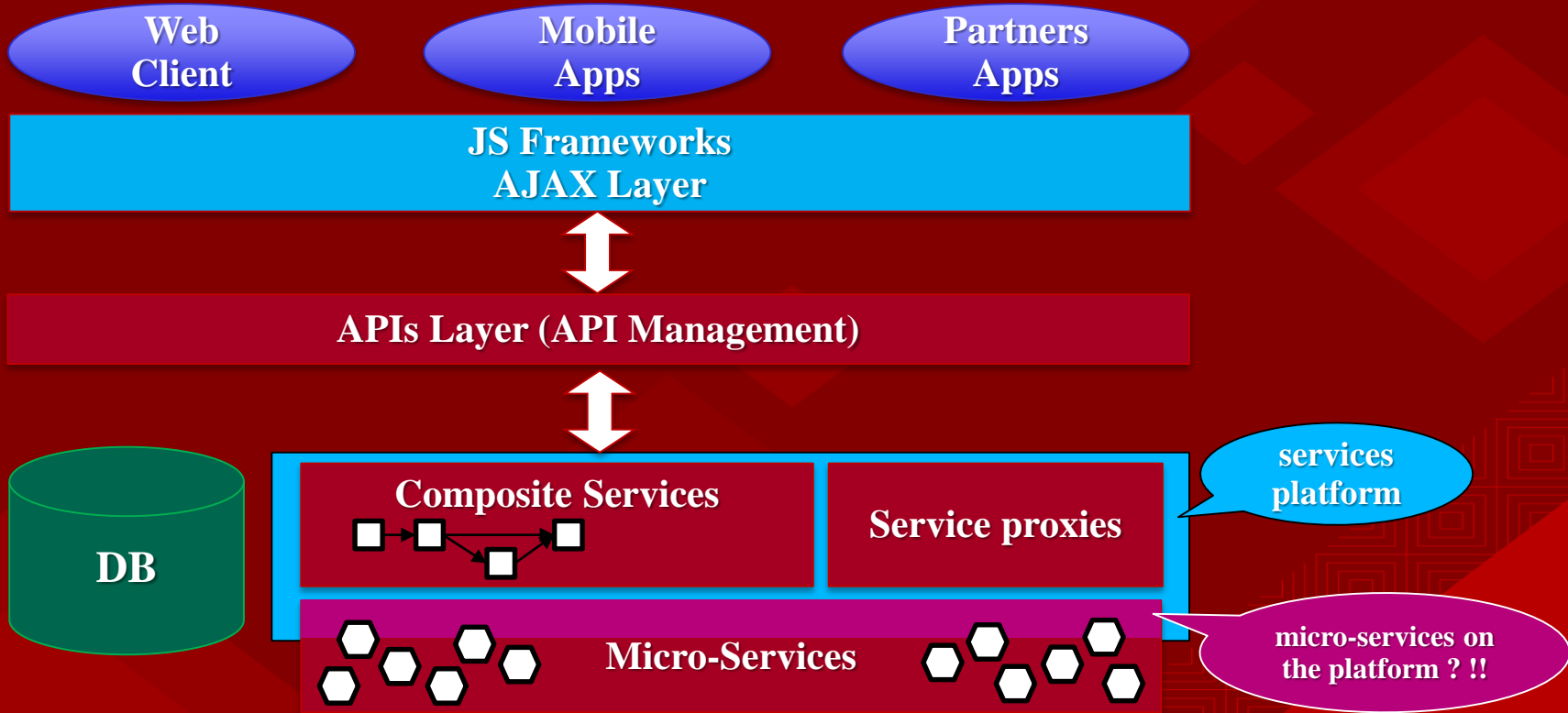
RED HAT  
**FORUM**  
Europe, Middle East & Africa



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 ...



# 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



RED HAT  
FORUM  
Europe, Middle East & Africa

# JBoss Fuse visual development



redhat.

The screenshot displays the JBoss Fuse IDE interface. At the top, there are tabs for 'Variables' and 'Breakpoints'. Below this, a 'Debug' window shows the execution of a clean package in a local Camel context. The main workspace features a visual Camel route diagram with components like 'getSchoolNum', 'bean1', 'refFileToHouse', 'log', and 'log1'. A 'Variables' window shows several Camel breakpoints. A 'MessageBody' window displays an XML message body. The 'Console' window at the bottom shows the application's log output.

**Variables**

Name	Value
CamelDebugger	CamelDebuggerSettings (id=-1491535378)
Endpoint	to1
Processor	CamelProcessor (id=114966)
Exchange	CamelExchange (id=1739850177)
Message	CamelMessage (id=1739850177)
MessageBody	<?xml version="1.0" encoding="UTF-8"?>\n<HouseInfo infotype=\nCamelFileAbsolute = false\nCamelFileAbsolutePath = /Users/C\nID=ChristinatekiMacBook-Air-local-50104-1431907009838-0-
MessageHeaders	
MessageId	

**MessageBody**

```
<?xml version="1.0" encoding="UTF-8"?>\n<HouseInfo infotype="LoanHouse">\n  <nationalID=856789</nationalID>\n  <address>9 Bowdoin St, Boston, MA 02114, United States</address>\n  <bedroom>5</bedroom>
```

**Console**

```
/Library/Java/JavaVirtualMachines/jdk1.7.0_60.jdk/Contents/Home/bin/java (2015年5月21日 下午9:47:30)\n[INFO] Initializing Dozer: Version: 5.5.0, Thread Name: Blueprint Extender: 1\n[INFO] Dozer JMX MBean [org.dozer.jmx:type=DozerStatisticsController] auto registered with the Platform MBean Server\n[INFO] Dozer JMX MBean [org.dozer.jmx:type=DozerAdminController] auto registered with the Platform MBean Server\n[INFO] Initializing a new instance of dozer bean mapper.\n[INFO] Loading Dozer mapping file fileToHouse.xml.\n[INFO] AllowOriginalMessage is enabled. If access to the original message is not needed, then its recommended to turn this option off as it\n[INFO] StreamCaching is not in use. If using streams then its recommended to enable stream caching. See more details at http://camel.apache.c\n[INFO] Registering module: com.fastermw.jackson.module:jaxb:JaxbAnnotationModule@1b656802\n[INFO] Route: appraisalRoute started and consuming from: Endpoint[activemq://queue:house]\n[INFO] Route: locationRoute started and consuming from: Endpoint[activemq://queue:location]
```

Visual debugger for  
Camel routes:

- Break points
- Tracing



# JBoss Fuse visual development



The screenshot displays the JBoss Fuse visual development environment. The main window shows a data mapping interface with a 'Source: CustInfo' tree on the left and a 'Target: CustInfo' tree on the right. A central 'Transformations' table lists the mappings:

Source	Target
age	age
firstName	firstName
infotype	infotype
lastName	lastName
nationalID	nationalID
occupation	occupation

Below the mapping interface, a 'New Fuse Transformation' dialog box is open. It contains the following fields:

- Project: webui
- Transformation ID: dataconvert
- Dozer File Path: dataconvert.xml
- Camel File Path: src/main/webapp/WEB-INF/applicationContext.xml

The 'Types Transformed' section shows 'Source Type' set to 'XML' and 'Target Type' set to 'Other'.

**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 uses 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



```
@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

```
.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 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>
```

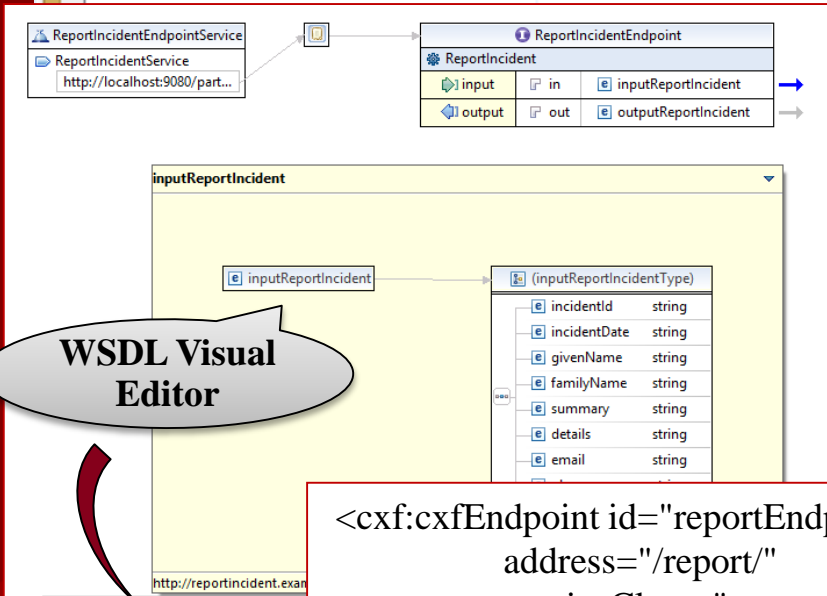
And similar with XML DSL



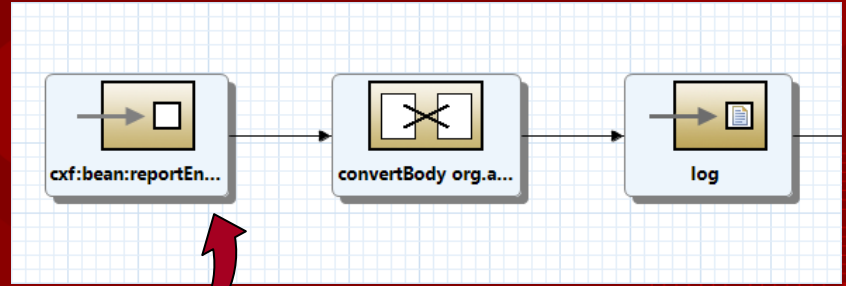
# Service Contracts

org.apache.camel.archetypes	camel-archetype-blueprint	2.15.1.redhat-621084
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-621084
org.apache.camel.archetypes	camel-archetype-java	2.15.1.redhat-621084
org.apache.camel.archetypes	camel-archetype-spring	2.15.1.redhat-621084

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



WSDL Visual Editor



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

- 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

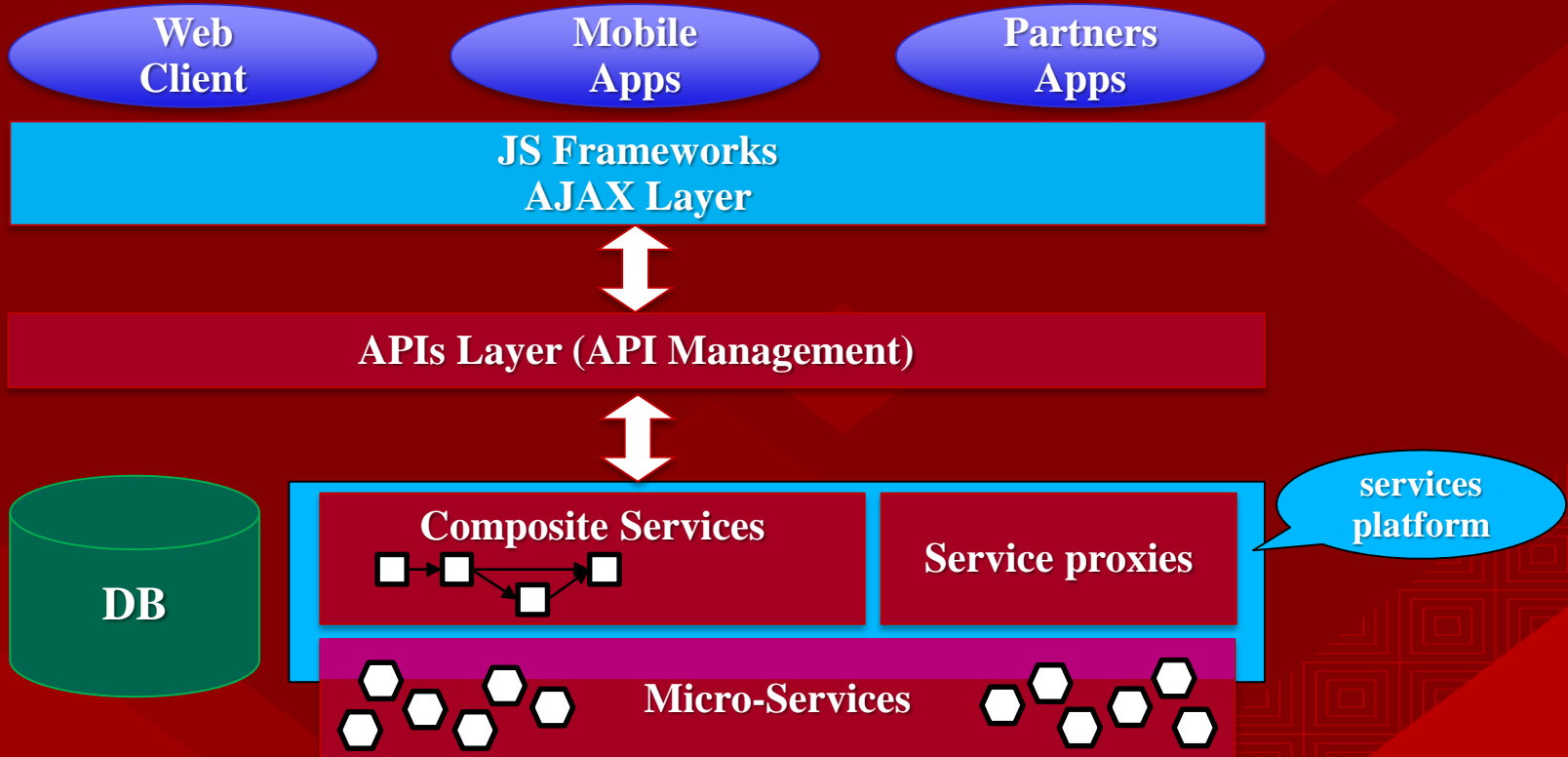


RED HAT  
**FORUM**  
Europe, Middle East & Africa

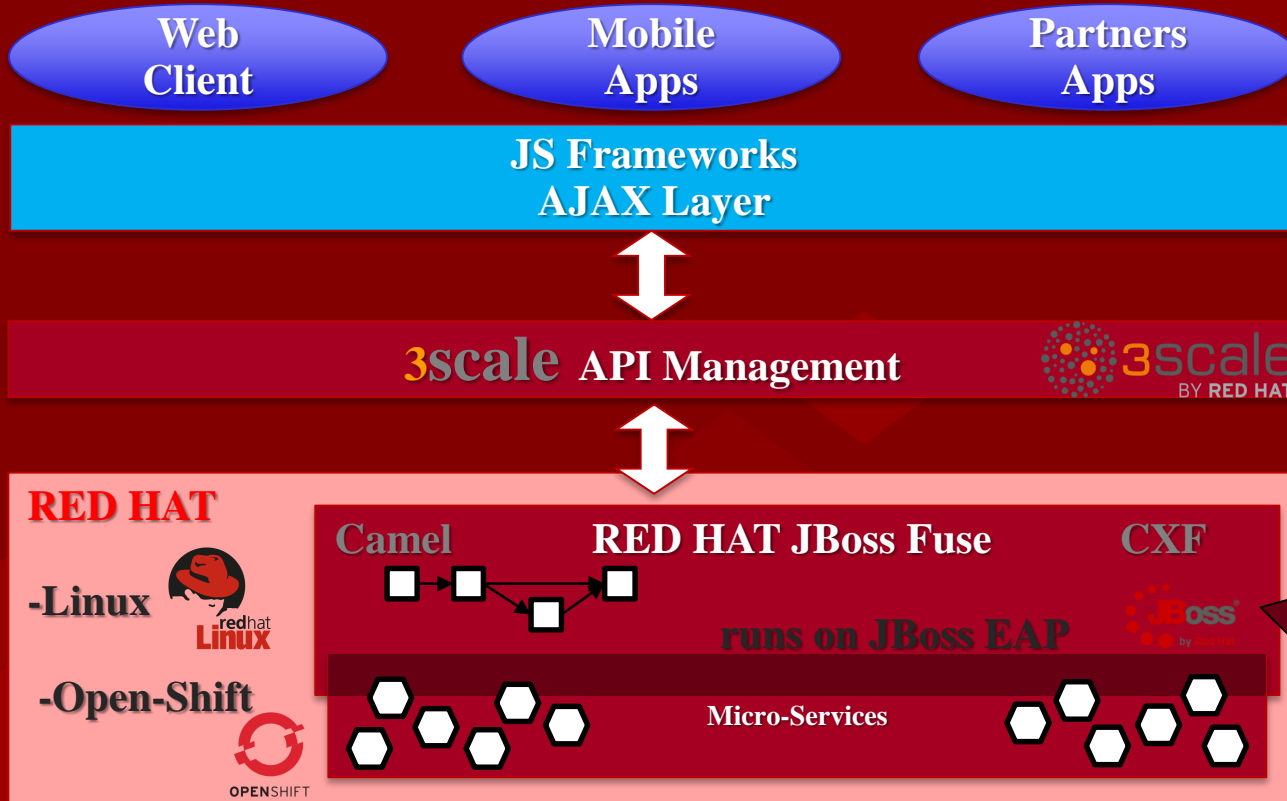


# RED HAT MICROSERVICES PLATFORM – END TO END

# Generic application architecture



# RED HAT based architecture



JBoss Fuse gives us an end to end service platform

# 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](mailto:alexeybr@matrix.co.il)



**redhat.**®

**Thank You**