Building Cloud Integration Application Through Knative and Apache Camel-K

Willem Jiang & Longchun Zhang
Speaker Introduction

Jiang Ning is the technical expert of Huawei, a member of the Apache Software Foundation, he worked on many Apache projects like Camel, CXF, ServiceMix and ServiceComb.

Longchun Zhang is the Technical Expert of PaaS CTO office of Huawei, is also senior architect in middle-ware domain, in charging of research, design, and implementation of gPaaS/iPaaS and middle-ware related services now.
Agenda

- The common user case of Smart park
- Why do we use Apache Camel
- How to use Camel-K and Knative to run Camel application
- Demo
Smart Park User Case

- Turnstile
- Visitor System
- Camera
- Notification System
- EI Services
- Visitor System
1. A new face event arrived face photo()
2. Recognize the face id by EI Services
3. Access control by Visitor System
4. Send message to the on duty mobile
5. Get order from the on duty mobile
6. Send the “Open” command to turnstile
Smart Park User Case

1. Cozmo
2. Visitor System
3. Check if the visitor information need to be notified.
4. Notification System
5. Integration Application
6. Cozmo Blocks

AI Edge
Why Apache Camel

- The Swiss knife of Integration
- Java-based integration framework
- Easy to run and integrate
- EIP implementation
- Powerful DSL
- 300+ Components
- Amazing Community, 12 + years development, still very active
Integration work

from("ApplicationA").doSomething().to("ApplicationB")
ROMA Integration Platform

Private Cloud

- Domestic Branch
  - ERP
  - PLM
  - MES
  - FIN
  - Enterprise Gateway

- Overseas Branch
  - ERP
  - PLM
  - MES
  - FIN
  - Enterprise Gateway

Internal Network

DMZ

Enterprise Apps

- Devices
- ERP
- PLM
- MES
- SRM
- CRM
- CS
- MKT
- FIN
- HR
- OA
- ...

Public Cloud

DMZ

ROMA Site

Enterprise Gateway

Partner Apps

- Supplier
- Bank
- Logistics
- Warehouse

Powered By Apache Camel
Enterprise Integration Pattern

https://www.enterpriseintegrationpatterns.com/patterns/messaging/
from("jms://inQueue")
  .choice()
    .when(header("foo").isEqualTo("bar"))
      .to("jms://widgetQueue")
    .otherwise()
      .to("jms://gadgetQueue")
  .otherwise()
    .to("jms://gadgetQueue")
Camel Routes

Cozmo

Send request with name

Cozmo Blocks

SendCallback

Visitor System

http://68.183.250.205:8080

Notification
public void configure() throws Exception {
    restConfiguration().component("restlet").host("0.0.0.0").port("9080");

    rest().get("/visitor/{name}").to("direct:visitorCheckIn");

    from("direct:visitorCheckIn")
        .transform().simple("{"visitorName":"${header.name}"}")
        .removeHeaders("*")
        .setHeader(Exchange.CONTENT_TYPE, constant("application/json"))
        .to("http4://" + visitorServiceAddress + "/visitors/checkIn")
        .transform().jsonpath("$.visitorName")
        .to("direct:notifyService");

    from("direct:notifyService")
        .choice()
        .when().simple("${body} == "Alex"")
            .transform().simple("VIP ${body}")
        .end()
        .setHeader("message").simple("Welcome ${body} to robot world")
        .transform().simple("{"message":"${body} is here.\"}""
        .setHeader(Exchange.CONTENT_TYPE, constant("application/json"))
        .to("http4://" + notificationServiceAddress + "/notification/notify")
        .setBody().simple("${headers.message}"");
    // We need to clean up the header to send out the message
    .removeHeaders("*");
}
How to run Camel Application

• Spring Boot Application with Camel Route
• Need to add the dependencies of Camel component
• Build the java artifacts and images
• Push the images into docker registry
• Run the Application from the docker registry

What if we update the Camel Route?
SOA to Cloud Native

Service Oriented Architecture
(Smart pipes, dumb endpoints)

Microservices Architecture
(Smart endpoints, dumb pipes)

Cloud Native Architecture
(Infrastructure focused smart platform, business logic focused smart services)

https://www.infoq.com/articles/microservices-post-kubernetes
Camel-K

- Lightweight Integration platform
- Running Camel Application natively on Kubernetes.
- Designed for serverless and microservice architectures.
- Leveraging the Operator SDK
- Building Application on top of Knative Eventing

https://github.com/apache/camel-k
K8s Operator

from("gate:xxx").to("kafka:topic");
...
from("kafka:topic").transform(...)

kind: Integration
apiVersion: camel.apache.org/v1alpha1
metadata:
  name: my-integration
spec:
  sources:
    - name: source.java
      content:
        from("gate:xxx").to("kafka:topic");
        ...

1. Choose a Runtime
2. Create a project
3. Add dependencies
4. Build docker image
5. Deploy
Operator SDK

**Build, test, iterate**

- **Foundation**
- **Test locally**
- **Test with live cluster**
- **Publish**
- **Operator manifest**
- **Operator image**

https://coreos.com/operators/
Build provides a pluggable model for building containers from source code.

Serving is a scale-to-zero, request-driven compute runtime which leverages Istio to route traffic amongst revisions.

Eventing provides building blocks for consuming and producing events that adhere to the CloudEvents specification.

https://knative.dev/
Knative Service With Camel-K

- Service (my-function)
- AutoScale Min ... Max
- AutoScale Target
- Camel-K
  - PodAutoScaler
  - Route (name)
  - Configuration records history of
  - Revision
Knative Event

```java
from("direct:reset").wireTap("knative:channel/visitorCheckIn");
from("knative:channel/visitorCheckIn").transform(...).to(xxx);
from("knative:channel/noftifyService").transform(...).to(xxx);
```
Kubecon Camel 技术讨论群

https://github.com/WillemJiang/smarter-park-demo