What is Camel?
System Integration

Figure 1.1 Camel is the glue between disparate systems.
Integration Framework
PATTERN BASED INTEGRATION

Apache Camel, a powerful pattern-based integration engine with a comprehensive set of connectors and data formats to tackle any integration problem.

ENTERPRISE INTEGRATION PATTERNS

Build integrations using enterprise best practices.

200+ COMPONENTS

Batch, messaging, web services, cloud, APIs, and more ...

BUILT-IN DATA TRANSFORMATION

JSON, XML, HL7, YAML, SOAP, Java, CSV, and more ...

INTUITIVE ROUTING

Develop integrations quickly in Java or XML.

NATIVE REST SUPPORT

Create, connect, and compose APIs with ease.
Enterprise Integration Patterns
Enterprise Integration Patterns

- Content Based Router
- Message Filter
- Dynamic Router
- Recipient List
- Splitter
- Aggregator
- Resequencer
- Content Enricher
- Content Filter
- Pipes and Filters
Camel Routes with Splitter
Camel Routes with Splitter
Camel Routes with Splitter
Camel Routes with Splitter
Camel Routes with Splitter

from("file:inbox")
from("file:inbox")
    .split(body()._tokenize("\n"))
Camel Routes with Splitter

from("file:inbox")
  .split(body().tokenize("\n"))
  .marshal(customToXml)

Custom data transformation
from("file:inbox")
  .split(body().tokenize("\n"))
  .marshal(customToXml)
  .to("activemq:line");
from("file:inbox")
    .split(body().tokenize("\n"))
    .marshal(customToXml)
    .to("activemq:line");
public void configure() throws Exception {

    from("file:inbox")
        .split(body().tokenize("\n"))
        .marshal(customToXml)
        .to("activemq:line");
}
public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("file:inbox")
            .split(body().tokenize("\n"))
            .marshal(customToXml)
            .to("activemq:line");
    }
}
import org.apache.camel.builder.RouteBuilder;

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("file:inbox")
            .split(body().tokenize("\n"))
            .marshal(customToXml)
            .to("activemq:line");
    }
}

Camel Routes

```java
from("file:data/inbox")
    .to("jms:queue:order");
```

**Java DSL**

```xml
<route>
    <from ri="file:data/inbox"/>
    <to uri="jms:queue:order"/>
</route>
```

**XML DSL**
Camel Architecture

Routing engine
A DSL wires endpoints and processors together to form routes.

CamelContext

Route 1

Route 2

Route N

from("file:c:\aDir")

.filter()

.xpath(expression)

.to("jms:aQueue");

Message filter processor

Processors
Handle things in between endpoints like:
- EIPs
- Routing
- Transformation
- Mediation
- Enrichment
- Validation
- Interception

Content-based router processor

Components
- Provide a uniform endpoint interface
- Connect to other systems
Camel runs everywhere

Application Servers

Linux Containers
Camel connects everything

Enterprise Systems
- File
- FTP
- JMS
- AMQP
- JDBC
- SQL
- TCP/UDP
- Mail
- HDFS
- JPA
- MongoDB
- Kafka
- ...

Public Cloud
- AWS
  - S3
  - SQS
  - Kinesis
  - ...
- Google
  - BigQuery
  - PubSub
- Azure
  - Blob
  - Queue

SaaS
- Box
- Dropbox
- Facebook
- Linkedin
- Salesforce
- SAP
- ServiceNow

IoT
- CoAP
- MQTT
- PubNub
What is Camel Rest DSL?
NATIVE REST SUPPORT

Create, connect, and compose APIs with ease.
Rest DSL

Server
- Hosting RESTful APIs
- Declared using Rest DSL
- API Swagger Docs

Client
- Invoke remote APIs

Tooling
- Swagger Doc → Rest DSL generator
Rest DSL Server

- Uses REST verbs
  - GET
  - POST
  - PUT

- URI templating
  - GET /order/{id}
  - PUT /order

- Configuration
  - Camel Component
  - Context-Path
  - Port
  - CORS
  - ...
Rest DSL Hello World Example

```java
@Component
public class HelloWorldRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        rest()
            .get("/hello")
            .route()
                .setBody().constant( value: "Hello World" );
    }
}
```
Rest DSL Hello World Example

```java
@Component
public class HelloWorldRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        rest()
            .get("/hello")
            .to("direct:hello");

        from(uri: "direct:hello")
            .log("Hello World")
            .setBody().constant( value: "Hello World");
    }
```
Hello World JSon Example

1. Add JSon libraries (camel-jackson)
2. Configure Rest DSL in JSon mode
3. POJO with JSon model (response)
4. Build response POJO in Rest DSL
Hello World JSon Example

1. Add JSon libraries (camel-jackson)

```xml
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jackson-starter</artifactId>
</dependency>
```

Add dependency in pom.xml
Hello World JSon Example

2. Configure Rest DSL in JSon mode

```java
@Component
public class HelloWorldRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {

        // configures rest-dsl to use servlet component and in JSon mode
        restConfiguration()
            .component("servlet")
            .bindingMode(RestBindingMode.json);

        // Turn on JSon binding mode (POJO <-> JSon)
    }
}
```
Hello World JSon Example

3. POJO with JSon model (response)

```java
public class ResponseObject {
    private String response;
    private String name;

    public String getResponse() { return response; }
    public void setResponse(String response) { this.response = response; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

POJO class with getter/setters
4. Build response POJO in Rest DSL

```java
// route called from REST service that builds a response message
from( uri: "direct:hello")
    .log("Hello World")
    .bean( bean: this, method: "createResponse" );

/**
 * Method that creates a POJO with the response
 */
public ResponseObject createResponse() {
    ResponseObject response = new ResponseObject();
    response.setResponse("Hello World");
    response.setName("your name");
    return response;
}
```
What is 3Scale?
What is 3Scale?

- API Management
- Access Control, Rate Limiting, Analytics
- Developer Workflows
- API Traffic control
PreReqs

- Maven
- Git
- JDK
- Red Hat Developer Account
- Free Trial Account with 3Scale
  - https://www.3scale.net/signup/
- Text Editor or IDE of your choice
  - Screenshots and any directions will be based on JBoss Developer Studio
High Level Workshop Steps

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab

- Create Your OpenShift Online Account and Download the OC tools
- Download the project base from Github
- Write your camel route
- Run it standalone using spring-boot to ensure you route is working
- Deploy to OpenShift
- Manage with 3Scale
- Play Your API and 3Scale
- BONUS: Setup Swagger
OpenShift Online Registration and OC Tools

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/00-create-openshift-online-account

- Select sign up for free
- Select ‘Add a New Plan’, Free Plan
- Wait for email
- Install oc utils from here or by going to about section in web console:
  - https://access.redhat.com/downloads/content/293/ver=3.3/rhel---7/3.3/x86_64/product-software
  - https://console.starter-ca-central-1.openshift.com/console/command-line
  - Download the appropriate ‘oc’ tool for your operating system
Download the Project Base from GitHub

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/summit-example
Writing Your Camel Route

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/01-create-camel-route

1. Create new class called ‘MyRoute’ in package my.project.route
2. Annotate the class with @Component
3. Make the class extend the RouteBuilder class
4. Implement the configure method with your Rest DSL route
Run Standalone Spring Boot App

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/01-create-camel-route

1. Run your Camel route using standalone spring-boot to ensure you route is working
   a. mvn spring-boot:run

2. Hit your Camel Rest Endpoint to Ensure it works
Deploy Your Route to OpenShift

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/02-deploy-to-openshift

- Deploy to openshift online
  - Click link to go to OpenShift Management Console
  - Upper right hand corner <copy login command>
- Go to terminal window
  - Paste the copied oc login command
  - Browse to Camel project
  - Run ‘mvn fabric8:deploy’
Manage Your API with 3Scale

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/03-manage-with-3scale

1. Create your free 3Scale Account if you have not already
   a. https://www.3scale.net/signup/
2. Create a Developer and a Service
3. Create an Application Plan
4. Assign Application Plan to Developer to Create an Application
5. Define and test you API
   a. curl
      "https://mary-test-summit-2445582096281.staging.gw.apicast.io:443/camel/hello?user_key=88a33411066fbccaa723b738fccc5f65a183951b7045a8b73c5331bf6e0e0038"
Play Your API and 3Scale

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/03-manage-with-3scale

Options:

- Add Rate Limits to your hello_World method
- Set up Alerts
- View Analytics
- Set up a Developer Portal
BONUS: Setup Swagger

https://github.com/mmistretta/RHSummit2018Camel3ScaleLab/tree/master/04-swagger-docs

- Add Swagger docs to Camel Route
- Display Swagger docs in Developer Portal
HELPFUL LINKS

https://docs.openshift.com/online/getting_started/beyond_the_basics.html#getting-started-beyond-the-basics

Not so free book

- Discount code (39%): camel39

(ordering from Manning)

THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat