Table of Contents

1. Core Engine Lab

Goals

After completing this lab, you should understand the following Apache Camel key concepts:

- Route
- Processor
- CamelContext
- DSL
- Message
- Exchange

You should be familiar with JBoss Developer Studio and the Fuse Camel Editor.

Lab Assets

The lab exercises and solutions are available in the following zip archives:

- https://github.com/gpe-mw-training/camel-labs/archive/v0.3-exercise.zip.
- https://github.com/gpe-mw-training/camel-labs/archive/v0.3-solution.zip.

1.1. Explore a Project

The goal of this exercise is to familiarize you with a typical integration project.

In this lab exercise, you will complete the following activities:

- Use JBoss Developer Studio to create a new Fuse project.
- View Apache Camel Routes.
- Run a project locally.
- Use the Palette within the Camel Editor to add a log processor.

1.1.1. Create a New Fuse Project

1. Launch JBoss Developer Studio on your computer.
2. Select another workspace `~/Temp/workspace-gpe-demo`.

3. Select **File** → **New** → **Fuse Project**.

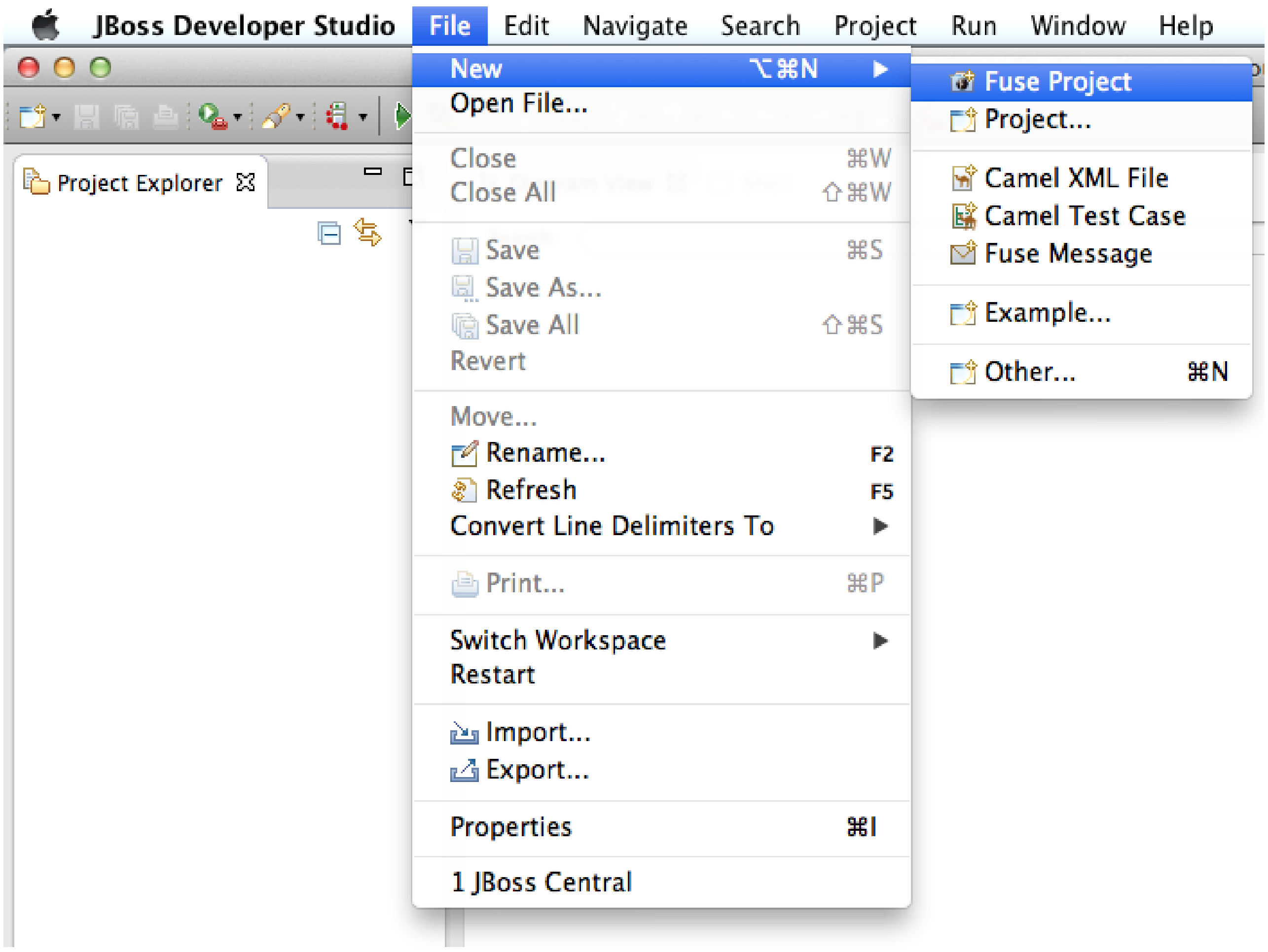
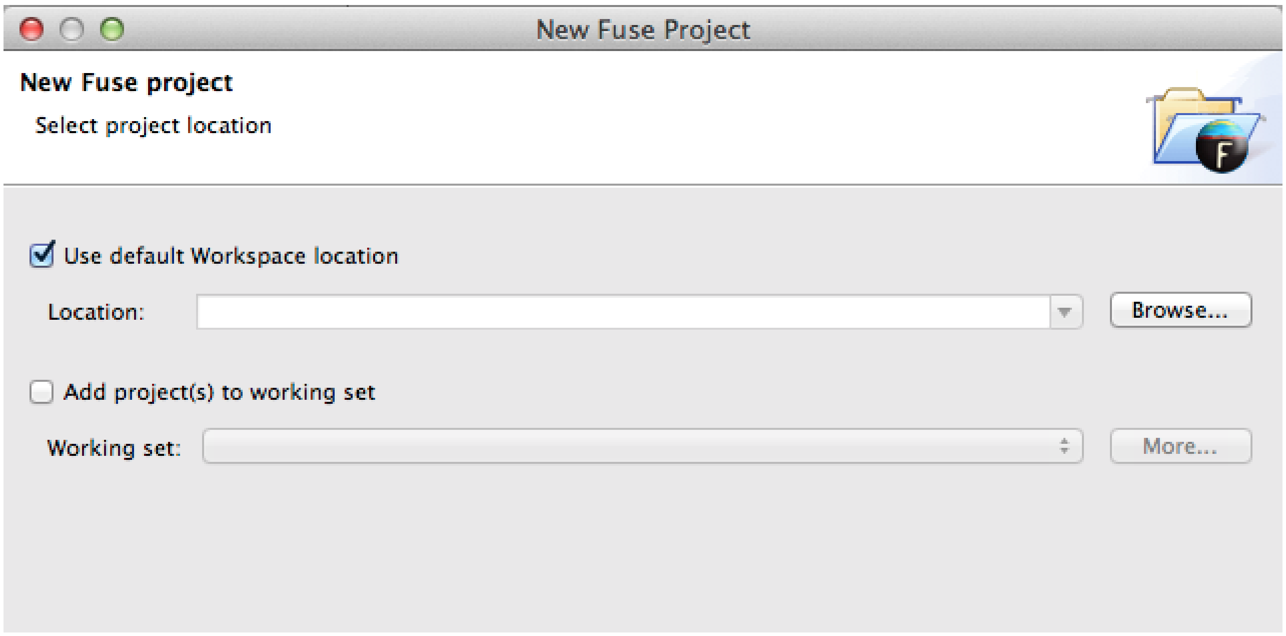


Figure 1. JBoss Developer Studio - New Fuse Project

4. Do not change the default project **Location**.
5. Click **Next**.


New Fuse Project

New Fuse project

Select project location

☑ Use default Workspace location

Location:

☐ Add project(s) to working set

Working set:

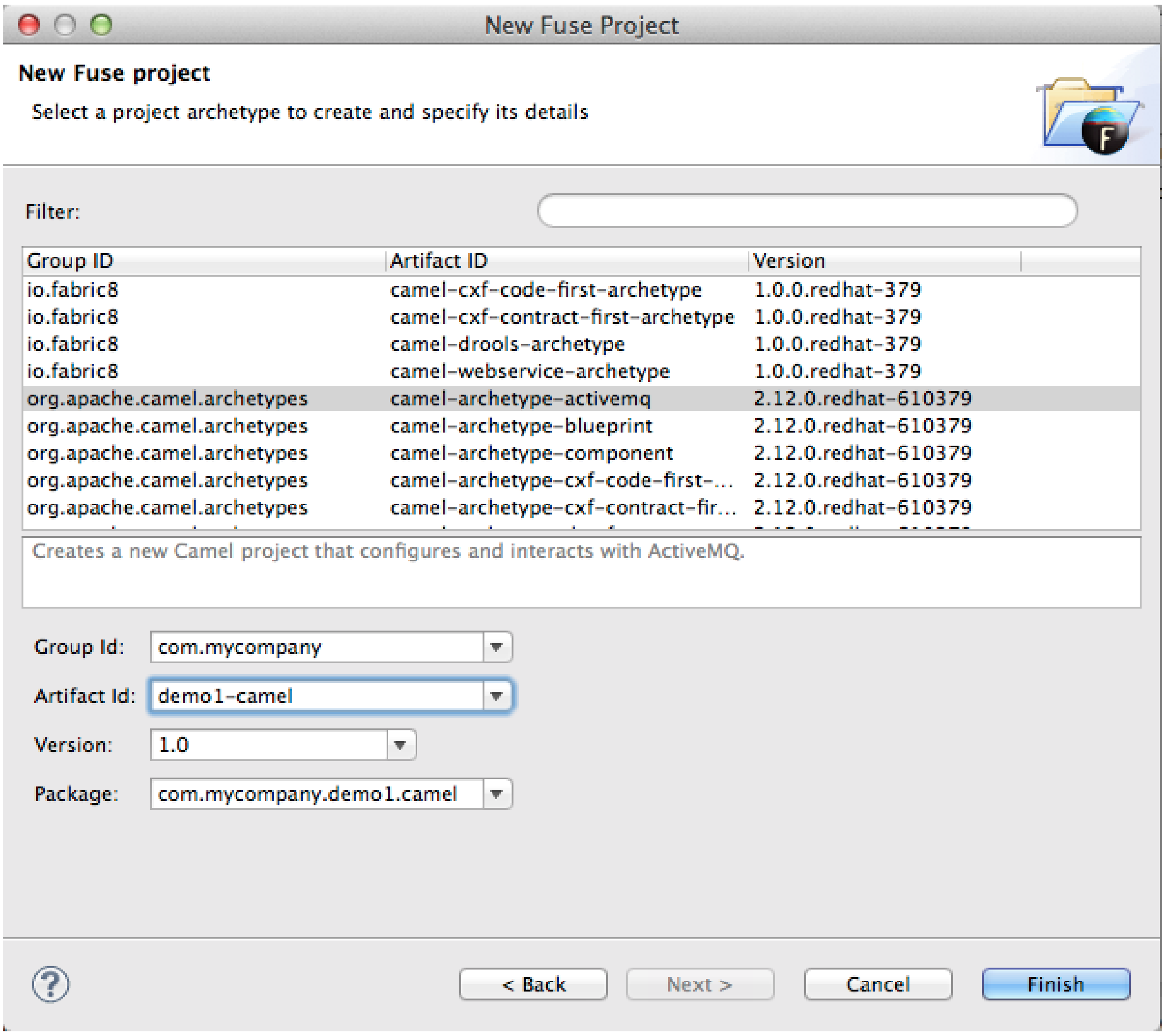
Figure 2. New Fuse Project - Select project location

6. In the **New Fuse Project — Select a project archetype to create and specify details** window, select `camel-archetype-activemq` as the **Apache Maven archetype**.

a. For the **Artifact Id** select `demo1-camel`.

b. For the **Version** number, select `1.0`.

c. For the **Package** name, leave the default selection, `com.mycompany.demo1.camel`.



New Fuse Project

New Fuse project

Select a project archetype to create and specify its details

Filter:

Group ID	Artifact ID	Version
io.fabric8	camel-cxf-code-first-archetype	1.0.0.redhat-379
io.fabric8	camel-cxf-contract-first-archetype	1.0.0.redhat-379
io.fabric8	camel-drools-archetype	1.0.0.redhat-379
io.fabric8	camel-webservice-archetype	1.0.0.redhat-379
org.apache.camel.archetypes	camel-archetype-activemq	2.12.0.redhat-610379
org.apache.camel.archetypes	camel-archetype-blueprint	2.12.0.redhat-610379
org.apache.camel.archetypes	camel-archetype-component	2.12.0.redhat-610379
org.apache.camel.archetypes	camel-archetype-cxf-code-first-...	2.12.0.redhat-610379
org.apache.camel.archetypes	camel-archetype-cxf-contract-fir...	2.12.0.redhat-610379

Creates a new Camel project that configures and interacts with ActiveMQ.

Group Id: com.mycompany

Artifact Id: demo1-camel

Version: 1.0

Package: com.mycompany.demo1.camel

Figure 3. New Fuse Project - Select project archetype

7. Click **Finish**.

1.1.2. View the Apache Camel Routes

This project contains two Apache Camel routes. The first route consumes three XML files from the `src/data` directory and creates an Exchange for every XML file. The Exchange contains the file metadata as headers and properties, and it publishes a JMS Message to the `personal.records` queue.

The second route consumes the JMS Message from the queue, and it creates an Exchange that is evaluated against a condition using a `Content Base Router` (CBR) Enterprise Integration Pattern (EIP). The CBR checks the JMS message and corresponding XML file to see if it contains the `/person/city/` tag with a value equal to `London`. If the condition is matched, the Exchange creates a file with the contents of the JMS Message in the `target/messages/uk` directory. If the condition is not matched, the Exchange creates a file in the `target/messages/others` directory instead.

Follow these steps to view the Apache Camel Routes:

1. On the menu, select **Window** → **Open Perspective** → **Fuse Integration**.

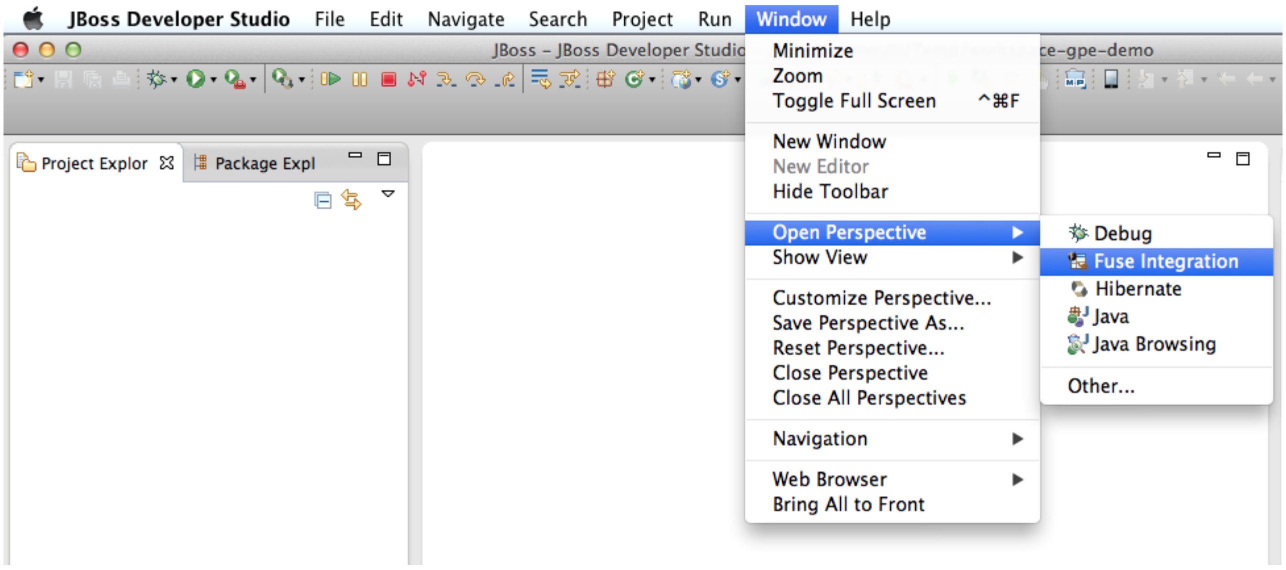


Figure 4. Open Perspective - Fuse Integration

2. Use **Project Explorer** to open the `demo1-camel` project and verify that the contents of the project appear:
   - `src/main/java`
   - `src/main/resources`
3. Expand the collapsed `src/main/resources` directory to reveal the `spring` subfolder.
4. In the `spring` subfolder, double-click the `camel-context.xml` file. The Fuse Camel Editor appears displaying an Apache Camel route.

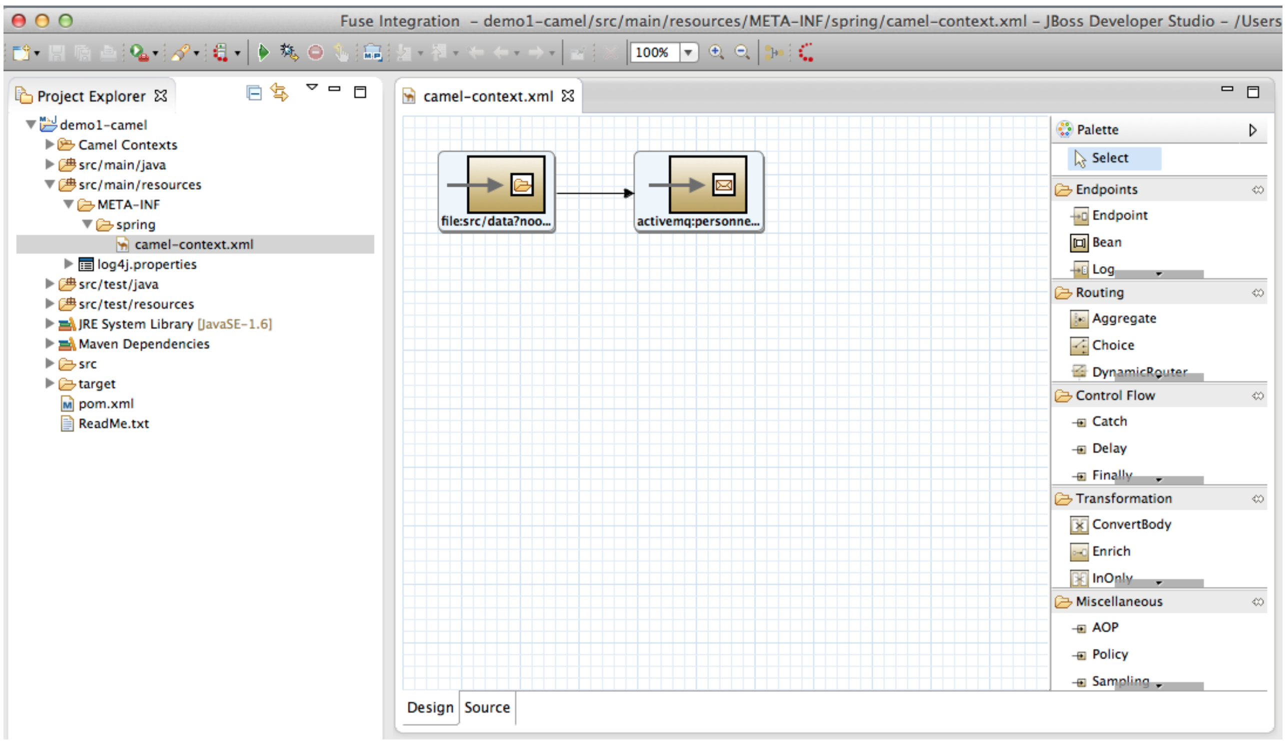


Figure 5. Camel IDE Window

5. On the menu, select **Routes**, and then select a route to view, either **Route: 1** or **Route: 2**.

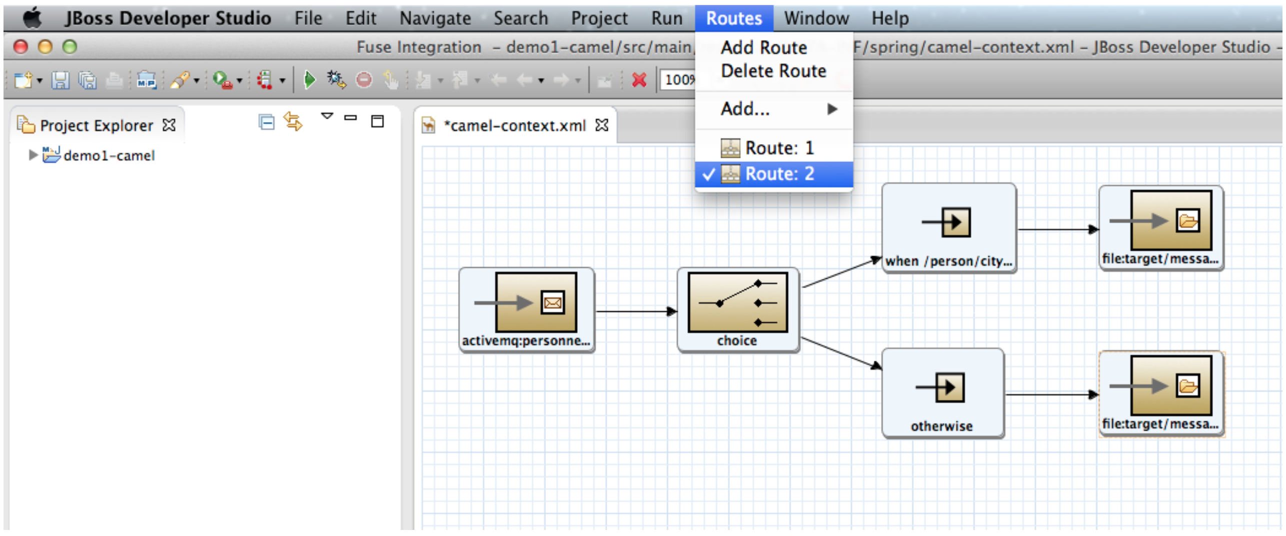


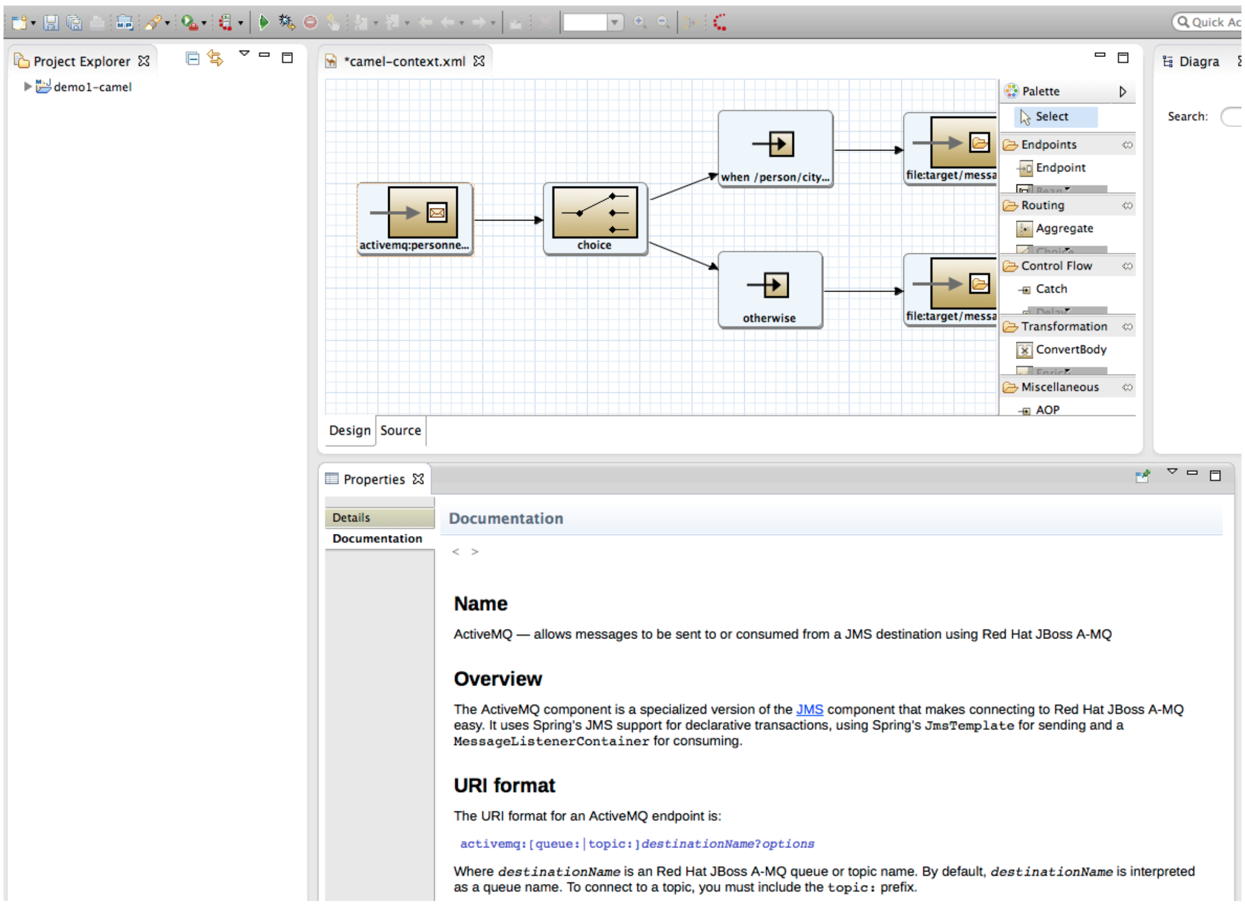
Figure 6. Select a route to view

6. Explore the properties defined for an endpoint by selecting one in the **Camel Editor** view and reviewing the information in the **Properties** view:

   a. Select the endpoint `activemq:personnel.records`.

   b. Open the **Properties** view (if necessary) and inspect the endpoint URI, ID, and description.

   c. In the **Properties** view, click the **Documentation** tab to review documentation for the endpoint/component.



Documentation

Name

ActiveMQ — allows messages to be sent to or consumed from a JMS destination using Red Hat JBoss A-MQ

Overview

The ActiveMQ component is a specialized version of the JMS component that makes connecting to Red Hat JBoss A-MQ easy. It uses Spring's JMS support for declarative transactions, using Spring's `JmsTemplate` for sending and a `MessageListenerContainer` for consuming.

URI format

The URI format for an ActiveMQ endpoint is:

```
activemq:[queue:|topic:]destinationName?options
```

Where *destinationName* is an Red Hat JBoss A-MQ queue or topic name. By default, *destinationName* is interpreted as a queue name. To connect to a topic, you must include the `topic:` prefix.

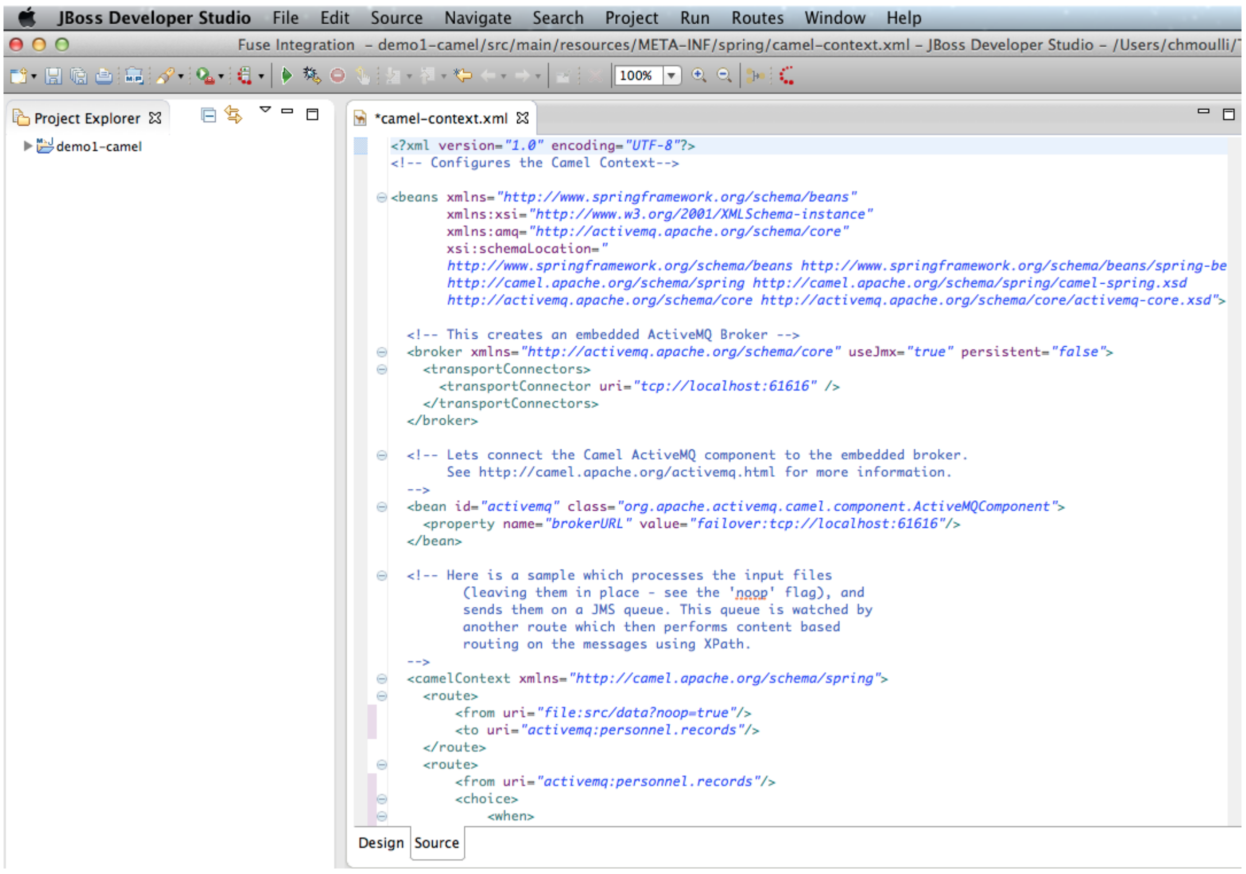
Figure 7. Documentation tab

d. Review the information for the `when` processor: The expression is `/person/city = 'London'` and the language is `xpath`.

e. Verify that the URI syntax of the `activemq` endpoint is `activemq:personnel.records`

f. Click the **Source** tab of the Camel Editor to inspect the Camel routes in XML format.

Both Java DSL and Camel Spring XML are supported languages for the source view.


JBoss Developer Studio File Edit Source Navigate Search Project Run Routes Window Help
Fuse Integration – demo1-camel/src/main/resources/META-INF/spring/camel-context.xml – JBoss Developer Studio – /Users/chmoulli/
100%
Project Explorer
demo1-camel
*camel-context.xml
Design
Source

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Configures the Camel Context-->

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:amq="http://activemq.apache.org/schema/core"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-be
       http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
       http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-core.xsd">

  <!-- This creates an embedded ActiveMQ Broker -->
  <broker xmlns="http://activemq.apache.org/schema/core" useJmx="true" persistent="false">
    <transportConnectors>
      <transportConnector uri="tcp://localhost:61616" />
    </transportConnectors>
  </broker>

  <!-- Lets connect the Camel ActiveMQ component to the embedded broker.
       See http://camel.apache.org/activemq.html for more information.
  -->
  <bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">
    <property name="brokerURL" value="failover:tcp://localhost:61616"/>
  </bean>

  <!-- Here is a sample which processes the input files
         (leaving them in place - see the 'noop' flag), and
         sends them on a JMS queue. This queue is watched by
         another route which then performs content based
         routing on the messages using XPath.
  -->
  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="file:src/data?noop=true"/>
        <to uri="activemq:personnel.records"/>
    </route>
    <route>
        <from uri="activemq:personnel.records"/>
        <choice>
            <when>
```

Figure 8. Sample XML file

Now you have seen what a typical Camel project looks like, and you have learned how to use the Camel Editor to view the Apache Camel Routes.

1.1.3. Run the Project Locally

A Fuse project is a collection of Camel Routes associated with a CamelContext. You start the Fuse project within JBoss Developer Studio. As explained in the course, whenever a Spring application context or XML module blueprint is created, the different beans declared within the `camel-context.xml` file are instantiated by Spring or OSGi Blueprint. This is how both the `DefaultCamelContext` and the `RouteBuilder` classes (containing the DSL based Route definitions) are created.

Follow these steps to run the project:

1. Expand the folder `src/main/resources` to reveal the `spring` folder containing the file `camel-context.xml`.
2. Right-click the `camel-context.xml` file and select **Run as** → **Local Camel Context**.

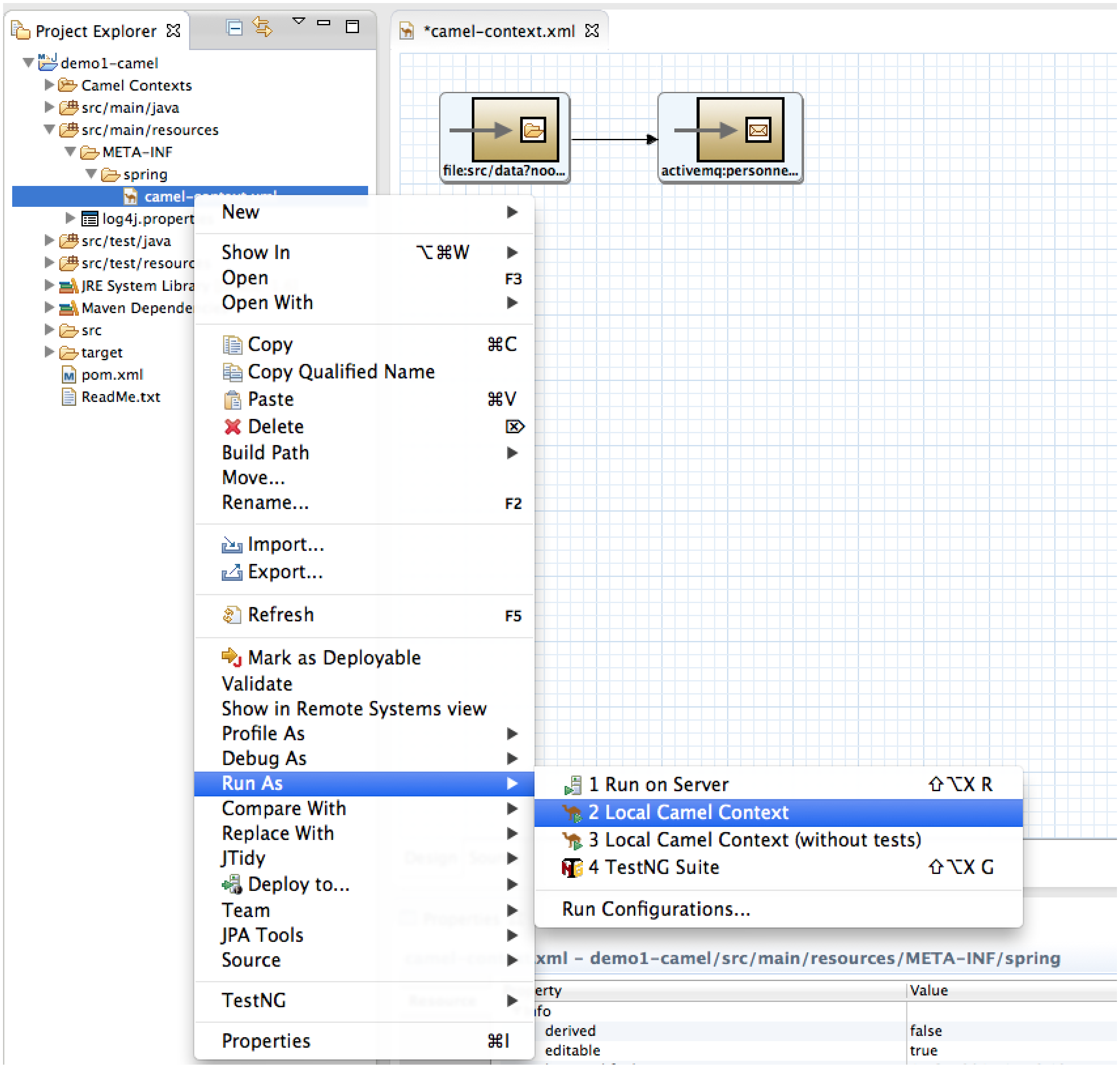
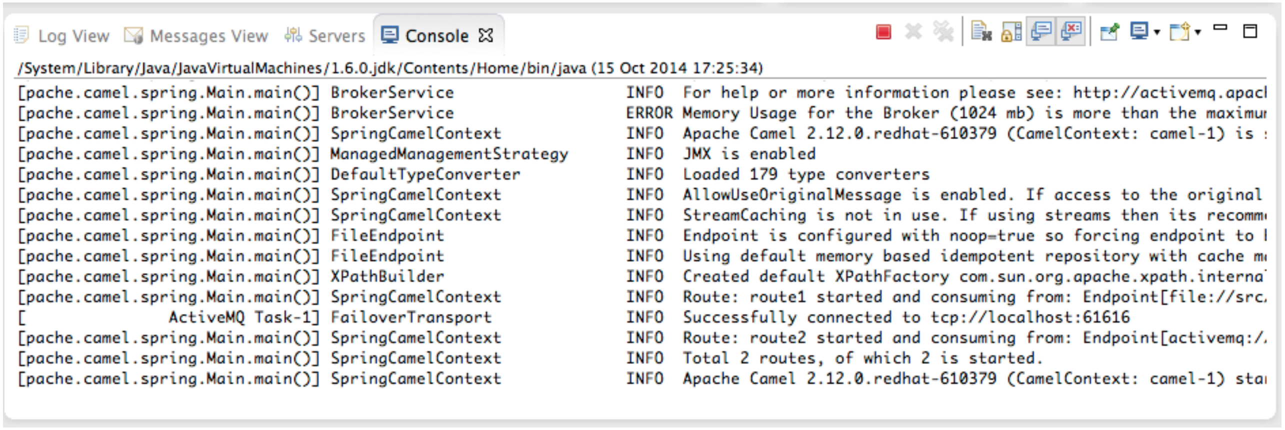


Figure 9. Run Local Camel Context

The Apache Camel Maven plug-in starts and the Maven console log shows that the CamelContext is created, routes are started, and endpoints are consumed.

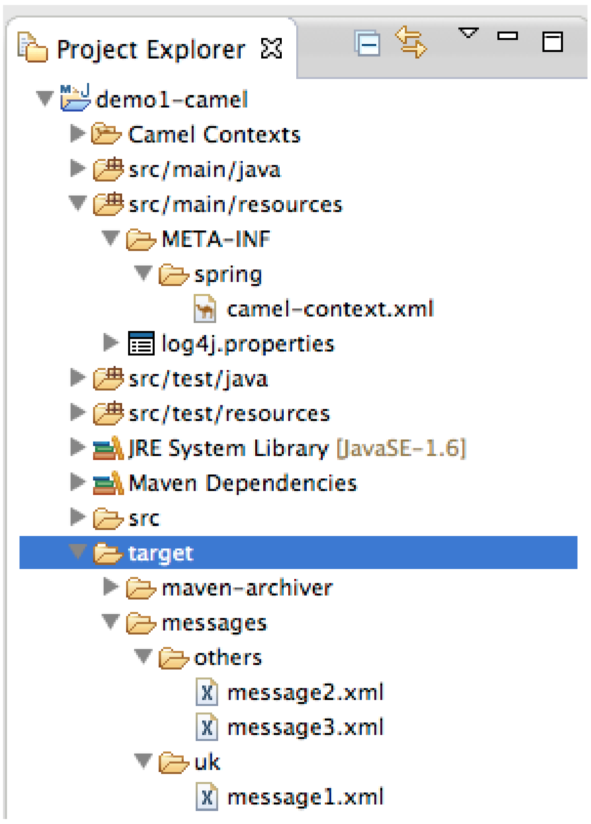


```
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (15 Oct 2014 17:25:34)
[pache.camel.spring.Main.main()] BrokerService                   INFO  For help or more information please see: http://activemq.apacl
[pache.camel.spring.Main.main()] BrokerService                   ERROR Memory Usage for the Broker (1024 mb) is more than the maximur
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) is :
[pache.camel.spring.Main.main()] ManagedManagementStrategy       INFO  JMX is enabled
[pache.camel.spring.Main.main()] DefaultTypeConverter            INFO  Loaded 179 type converters
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  AllowUseOriginalMessage is enabled. If access to the original
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  StreamCaching is not in use. If using streams then its recomm
[pache.camel.spring.Main.main()] FileEndpoint                    INFO  Endpoint is configured with noop=true so forcing endpoint to l
[pache.camel.spring.Main.main()] FileEndpoint                    INFO  Using default memory based idempotent repository with cache m
[pache.camel.spring.Main.main()] XPathBuilder                    INFO  Created default XPathFactory com.sun.org.apache.xpath.interna
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  Route: route1 started and consuming from: Endpoint[file://src.
[               ActiveMQ Task-1] FailoverTransport               INFO  Successfully connected to tcp://localhost:61616
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  Route: route2 started and consuming from: Endpoint[activemq:/.
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  Total 2 routes, of which 2 is started.
[pache.camel.spring.Main.main()] SpringCamelContext              INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) sta
```

Figure 10. Maven Console

3. Right-click the `target` directory of **Project Explorer** and select **Refresh**.

   a. Inspect the contents of the subdirectories `messages/uk` or `messages/others` to see the XML files that were created.


Project Explorer

```
demo1-camel
  Camel Contexts
  src/main/java
  src/main/resources
    META-INF
      spring
        camel-context.xml
    log4j.properties
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.6]
  Maven Dependencies
  src
  target
    maven-archiver
    messages
      others
        message2.xml
        message3.xml
      uk
        message1.xml
```

Figure 11. Project Explorer - Target Directories

4. To differentiate between what is running within the JVM and what was created, use the JMX layer and the **JMX Navigator** to discover the different MBeans objects which form both the CamelContext as well as the ActiveMQ broker:

   a. Select the **JMX Navigator** view.

   b. Expand the **Local Processes** tree.

   c. Click the green plus icon to add a new JMX Server connection.

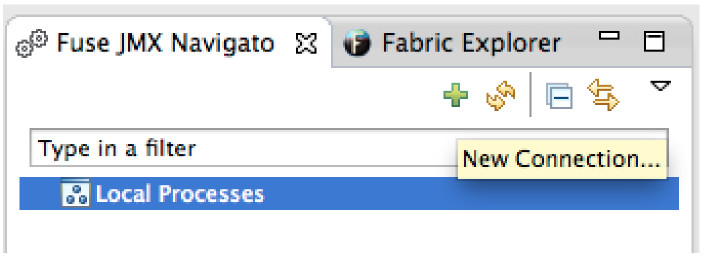


Figure 12. JMX Server Connection

   d. In the **Create a new JMX Connection dialog box**, click **Next**.

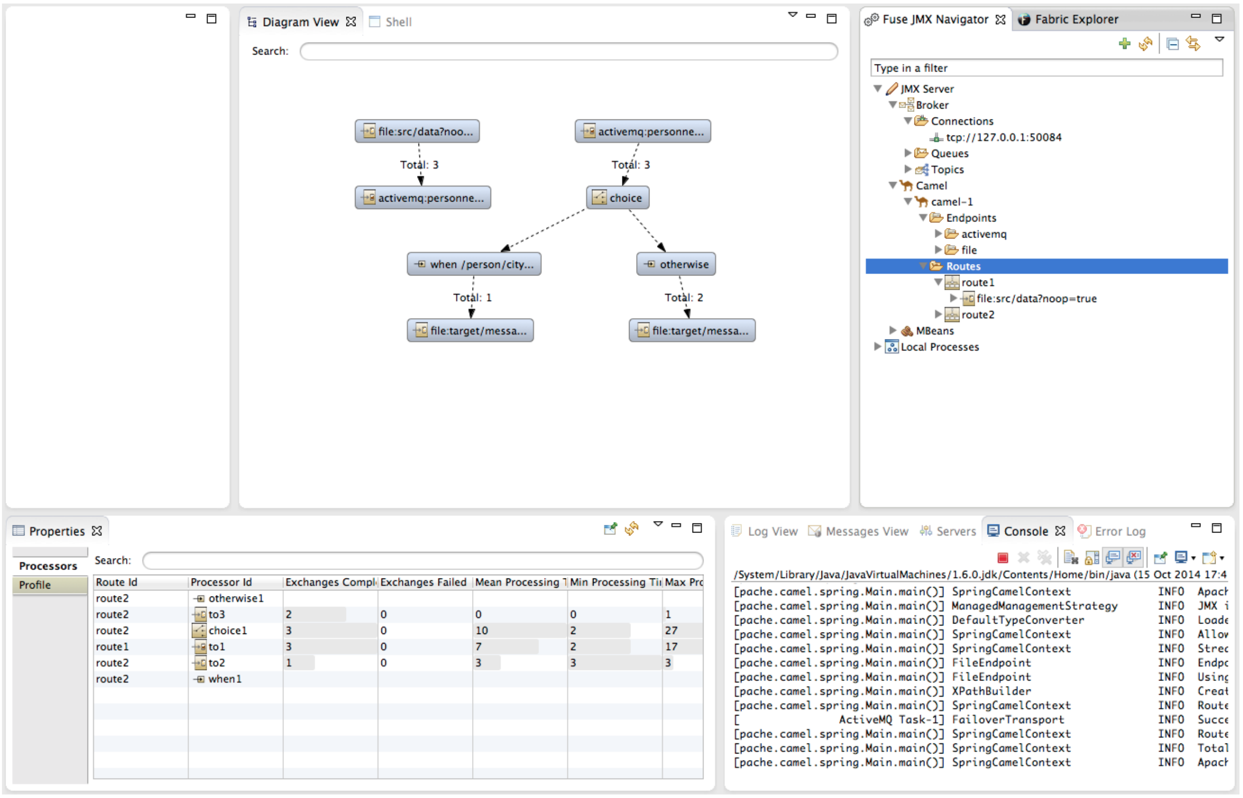
   e. Click the **Advanced** tab and add the JMX Url: `service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi`

   f. Click **Finish**. The `Camel` and `Broker` JMX domains are now displayed with icons.

   g. Expand the JMX Server icon, and select the Camel domain.

   h. Click **Window**→**Show View**→**Other**, and then select **Diagram** view to view a graphical representation of the Camel routes.

i. Switch to the **Properties** view to view the number and processing duration of the processed messages.



Route Id	Processor Id	Exchanges Compl	Exchanges Failed	Mean Processing T	Min Processing Ti	Max Pro
route2	otherwise1					
route2	to3	2	0	0	0	1
route2	choice1	3	0	10	2	27
route1	to1	3	0	7	2	17
route2	to2	1	0	3	3	3
route2	when1					

```
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (15 Oct 2014 17:4
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Apach
[pache.camel.spring.Main.main()] ManagedManagementStrategy   INFO  JMX i
[pache.camel.spring.Main.main()] DefaultTypeConverter        INFO  Loade
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Allow
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Strea
[pache.camel.spring.Main.main()] FileEndpoint                INFO  Endpo
[pache.camel.spring.Main.main()] FileEndpoint                INFO  Using
[pache.camel.spring.Main.main()] XPathBuilder                INFO  Creat
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Route
[                 ActiveMQ Task-1] FailoverTransport          INFO  Succe
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Route
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Total
[pache.camel.spring.Main.main()] SpringCamelContext          INFO  Apach
```

Figure 13. Sample JMX Navigator

1.1.4. Enable Tracing

Tracing enables you to track the contents of the Exchange and the activity of the processors.

Follow these steps to enable tracing:

1. If it is collapsed, expand the `CamelContext` Mbean within the **JMX Navigator**.
2. Right-click `camel-1` Mbean, and then select **Start Tracing**.

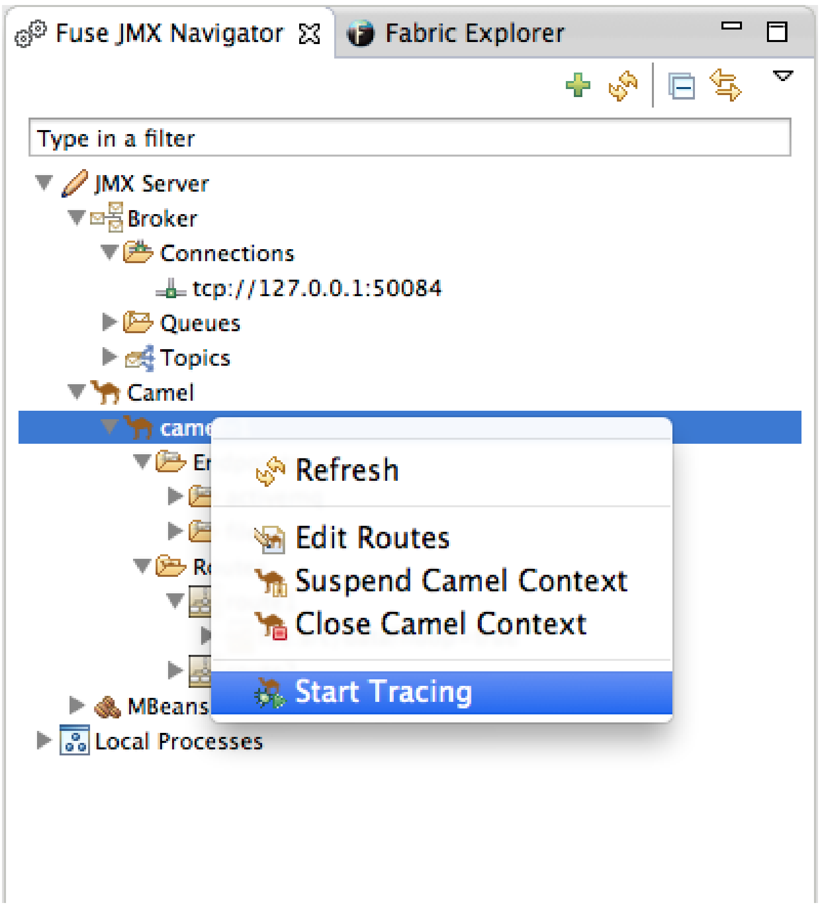


Figure 14. JMX Navigator - Start Tracing

3. Refresh the `camel-1`.
4. Inspect the new icon (same icon as before but with a green bug) that represents the CamelContext.

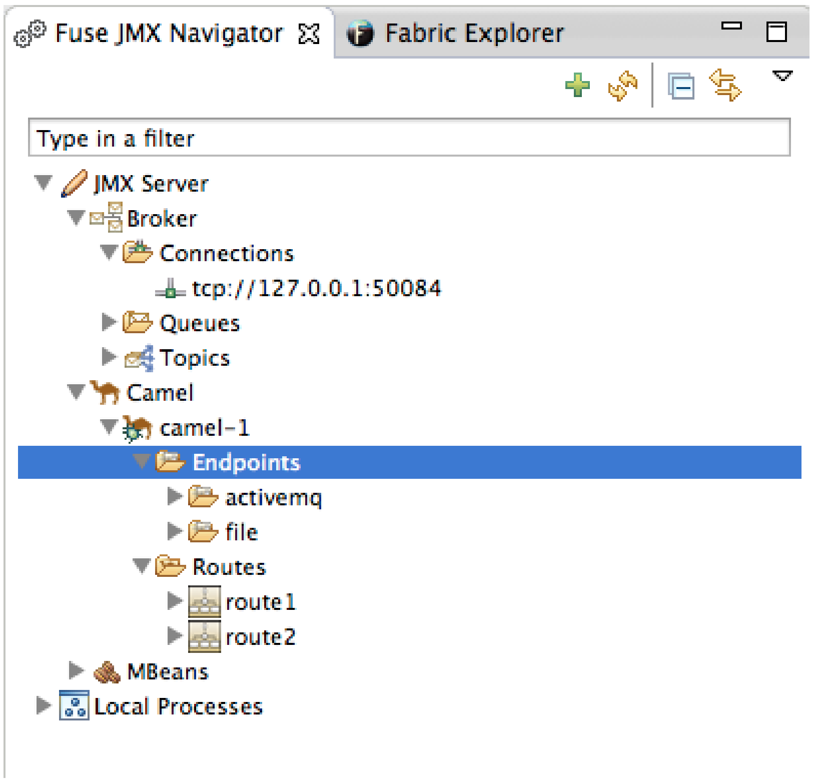


Figure 15. JMX Navigator - Tracer Context

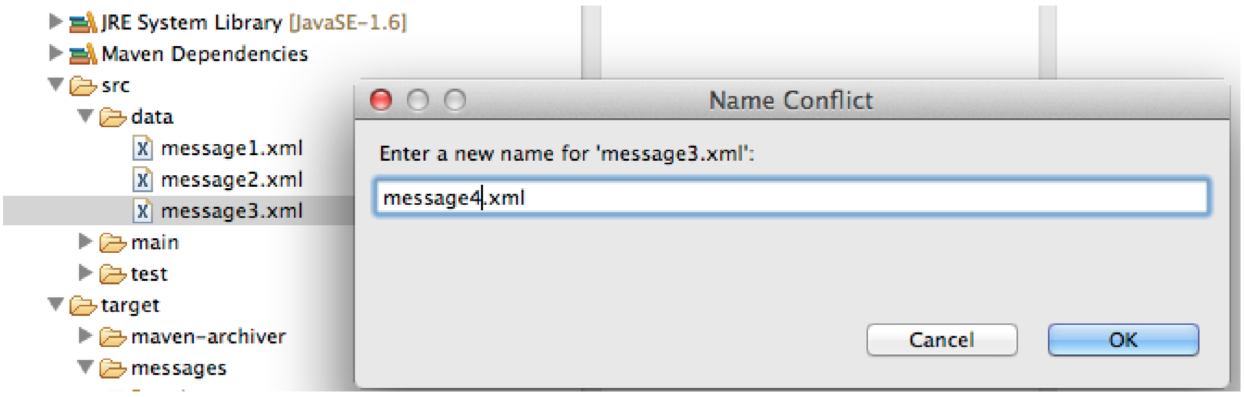
5. To further test the tracing feature, create a new message and and use the **Messages**

view and **Properties** view to review processing details:

a. Use **Project Explorer** to expand `src/data`.

b. Copy the contents of `message3.xml` and paste into a new file, `message4.xml`.

c. Save the new file in the same directory.



Enter a new name for 'message3.xml':

message4.xml

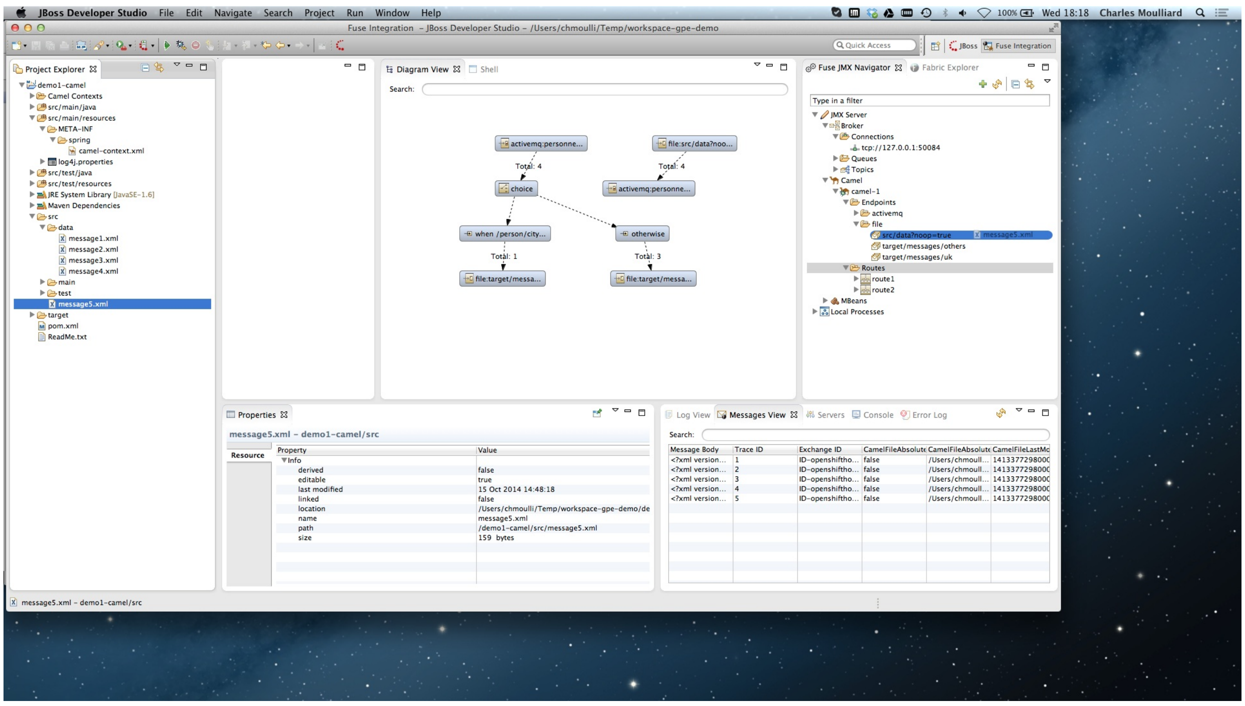
Figure 16. Sample - New Message

d. Use **Project Explorer** to expand `src/data`.

e. Copy the contents of `message3.xml` and paste into a new file `message5.xml`.

f. Save the new file in another directory, such as `src/test` for example, to exclude `message5.xml` from polling.

g. Drag the message and drop it onto the `file` endpoint in the **JMX Navigator**.



message5.xml – demo1-camel/src

Property	Value
Info	
derived	false
editable	true
last modified	15 Oct 2014 14:48:18
linked	false
location	/Users/chmoulli/Temp/workspace-gpe-demo/de
name	message5.xml
path	/demo1-camel/src/message5.xml
size	159 bytes

Message Body	Trace ID	Exchange ID	CamelFileAbsolute	CamelFileAbsolute	CamelFileLastMo
<?xml version...	1	ID-openshiftho...	false	/Users/chmoull...	141337729800
<?xml version...	2	ID-openshiftho...	false	/Users/chmoull...	141337729800
<?xml version...	3	ID-openshiftho...	false	/Users/chmoull...	141337729800
<?xml version...	4	ID-openshiftho...	false	/Users/chmoull...	141337729800
<?xml version...	5	ID-openshiftho...	false	/Users/chmoull...	141337729800

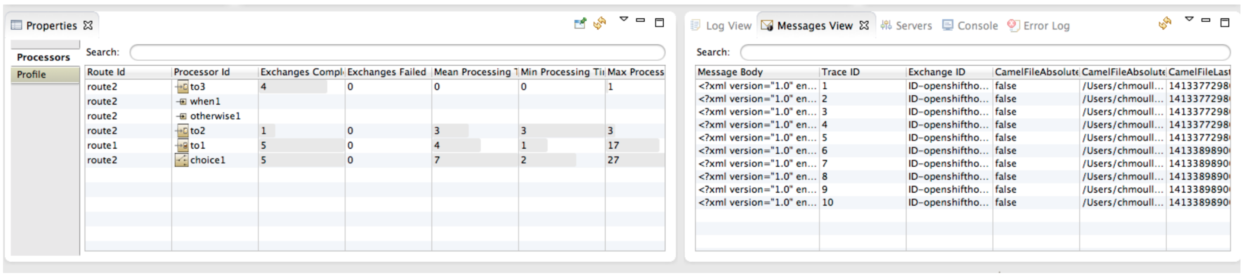
Figure 17. Sample - Add a File to an Endpoint

h. Open the **Properties** view to inspect tabular information about the processed Exchanges, including:

- Route ID
- Processor ID

- Exchanges Completed
- Exchanges Failed
- Mean Processing Time
- Max Processing Time
- Min Processing Time
- Last Processing Time

i. Open the **Messages** view to inspect information reported by the trace feature about the content of the Exchange, including body and headers.



Properties

Processors | Profile

Search:

Route Id	Processor Id	Exchanges Compl	Exchanges Failed	Mean Processing T	Min Processing Ti	Max Process
route2	to3	4	0	0	0	1
route2	when1					
route2	otherwise1					
route2	to2	1	0	3	3	3
route1	to1	5	0	4	1	17
route2	choice1	5	0	7	2	27

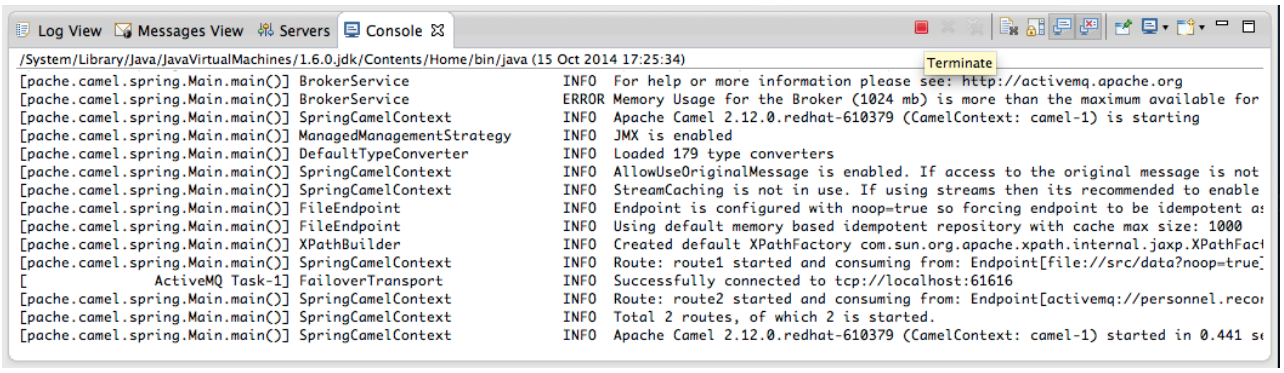
Log View | Messages View | Servers | Console | Error Log

Search:

Message Body	Trace ID	Exchange ID	CamelFileAbsolute	CamelFileAbsolute	CamelFileLast
<?xml version="1.0" en...	1	ID-openshiftho...	false	/Users/chmoull...	1413377298
<?xml version="1.0" en...	2	ID-openshiftho...	false	/Users/chmoull...	1413377298
<?xml version="1.0" en...	3	ID-openshiftho...	false	/Users/chmoull...	1413377298
<?xml version="1.0" en...	4	ID-openshiftho...	false	/Users/chmoull...	1413377298
<?xml version="1.0" en...	5	ID-openshiftho...	false	/Users/chmoull...	1413377298
<?xml version="1.0" en...	6	ID-openshiftho...	false	/Users/chmoull...	1413389890
<?xml version="1.0" en...	7	ID-openshiftho...	false	/Users/chmoull...	1413389890
<?xml version="1.0" en...	8	ID-openshiftho...	false	/Users/chmoull...	1413389890
<?xml version="1.0" en...	9	ID-openshiftho...	false	/Users/chmoull...	1413389890
<?xml version="1.0" en...	10	ID-openshiftho...	false	/Users/chmoull...	1413389890

Figure 18. Sample - Tracing results

6. Click the red square icon in the **Console** view to terminate the Maven process.



```
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (15 Oct 2014 17:25:34)
[pache.camel.spring.Main.main()] BrokerService                    INFO  For help or more information please see: http://activemq.apache.org
[pache.camel.spring.Main.main()] BrokerService                    ERROR Memory Usage for the Broker (1024 mb) is more than the maximum available for
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) is starting
[pache.camel.spring.Main.main()] ManagedManagementStrategy        INFO  JMX is enabled
[pache.camel.spring.Main.main()] DefaultTypeConverter             INFO  Loaded 179 type converters
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  AllowUseOriginalMessage is enabled. If access to the original message is not
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  StreamCaching is not in use. If using streams then its recommended to enable
[pache.camel.spring.Main.main()] FileEndpoint                     INFO  Endpoint is configured with noop=true so forcing endpoint to be idempotent a
[pache.camel.spring.Main.main()] FileEndpoint                     INFO  Using default memory based idempotent repository with cache max size: 1000
[pache.camel.spring.Main.main()] XPathBuilder                     INFO  Created default XPathFactory com.sun.org.apache.xpath.internal.jaxp.XPathFac
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  Route: route1 started and consuming from: Endpoint[file://src/data?noop=true
[               ActiveMQ Task-1] FailoverTransport                INFO  Successfully connected to tcp://localhost:61616
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  Route: route2 started and consuming from: Endpoint[activemq://personnel.reco
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  Total 2 routes, of which 2 is started.
[pache.camel.spring.Main.main()] SpringCamelContext               INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) started in 0.441 s
```

Figure 19. Console - Terminate a Maven process

You have enabled and tested the tracing feature.

1.1.5. Add a Log Processor

In this activity, you explore the Palette in the Camel Editor and add a Log Processor to the project. The Palette contains various endpoints and processors that you use to design an Apache Camel project. They are organized by topic:

- Endpoints
- Routing
- Control Flow
- Transformation
- Miscellaneous

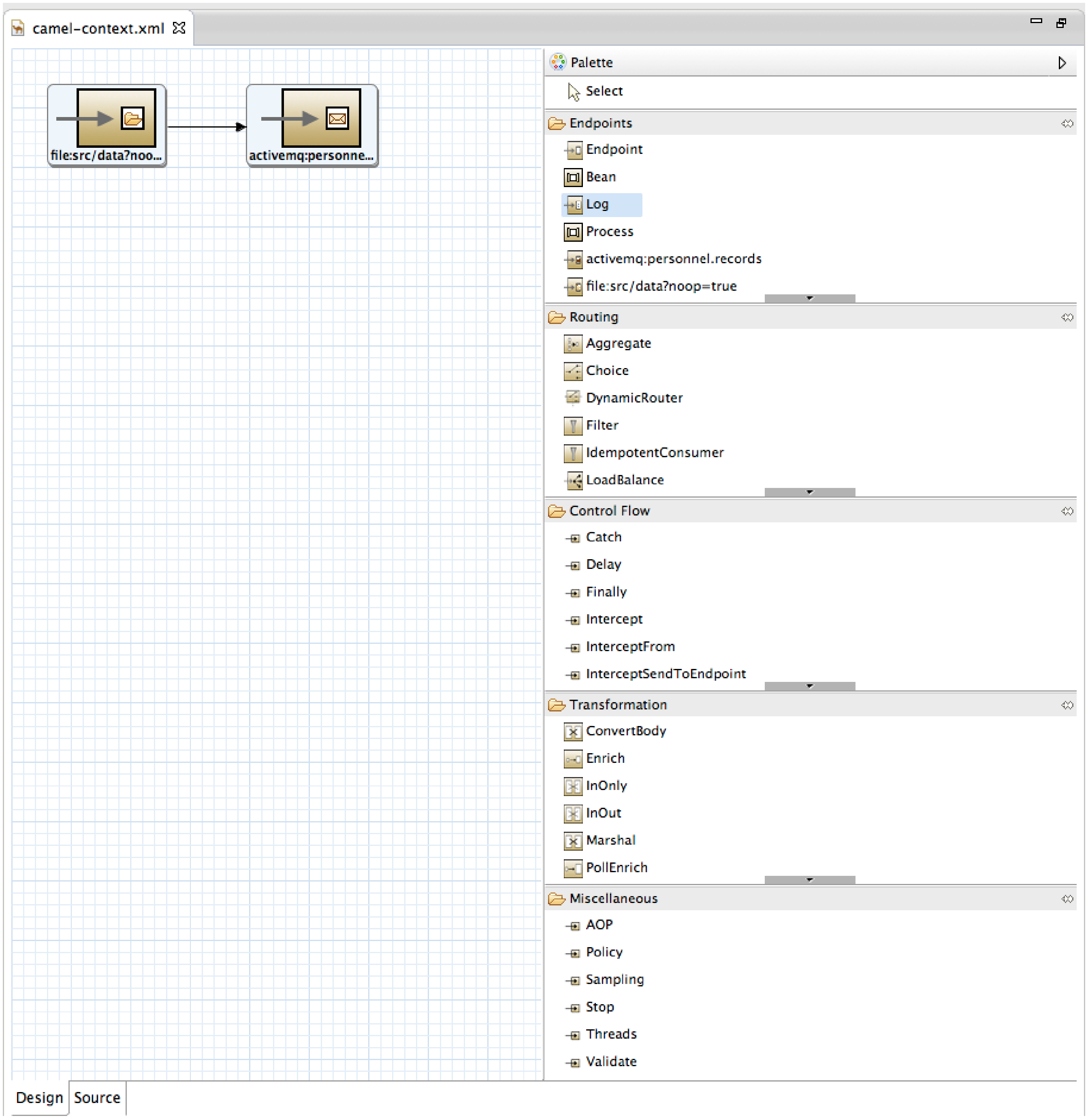


Figure 20. Camel Editor - Palette

Follow these steps to add a log processor:

1. From the **Palette** (right side of the Camel Editor view) select **Log** under the **Components** folder.
2. Drag the **Log** endpoint and drop it onto `Route 1`.

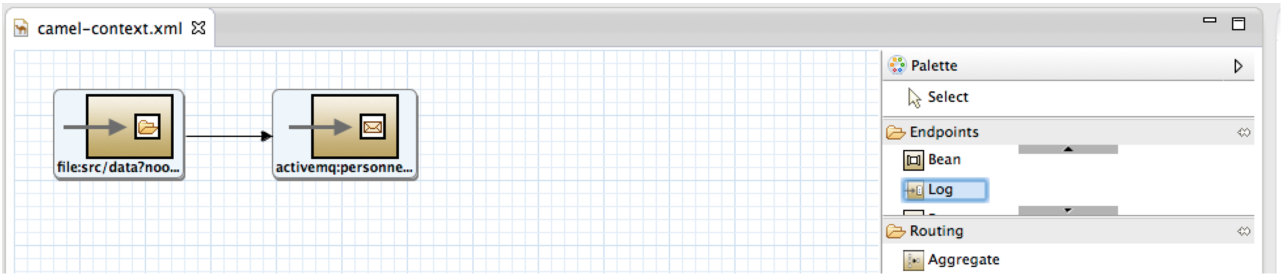


Figure 21. Sample - Add a Log Endpoint

3. Inspect the **Log** endpoint icon that appears on the **Design** view.

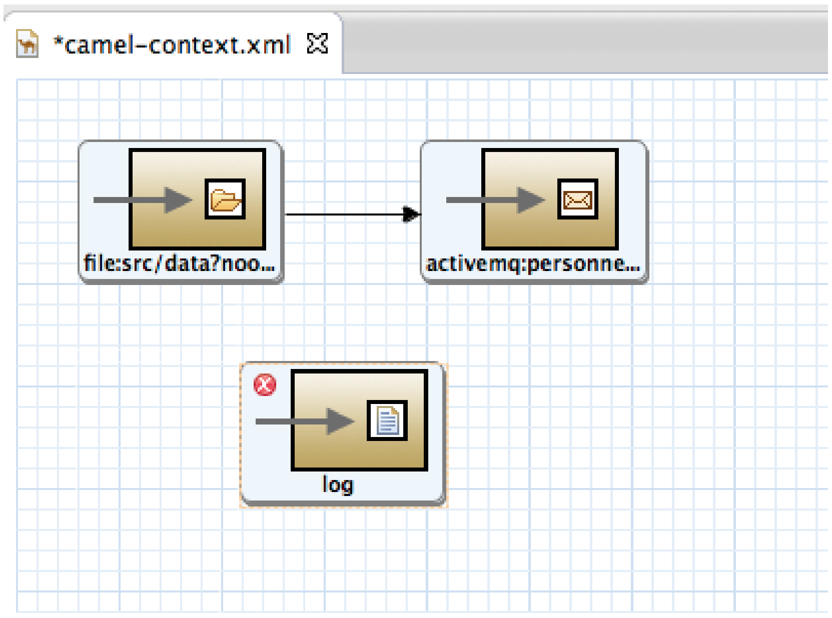


Figure 22. Sample - Endpoint Added

4. In the **Design** view, remove the connection between the `file:src/data` endpoint and the `activemq` endpoint:

   a. Right-click the connection, and then select **Remove**.

   b. Connect `file:src/data` to `log`.

   To remove any endpoint, you must first delete all connections to or from it.

5. In the **Design** view, add an arrow connecting the `log endpoint` to the `activemq` endpoint:

   a. Move your cursor over the endpoint and click the **Create Connection** icon that appears.

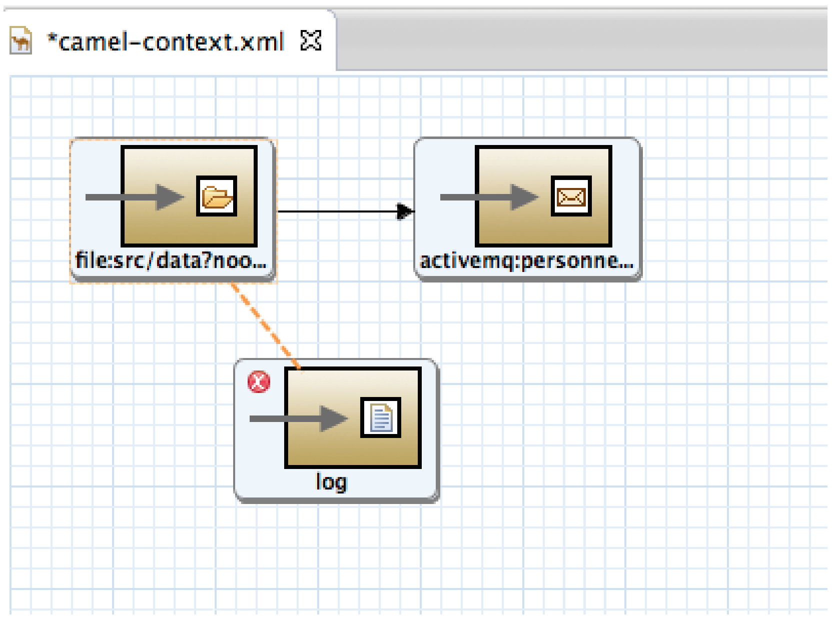


Figure 23. Sample - Connecting Log Endpoint to Activemq

6. Edit the log endpoint to add a message `File has been processed`.

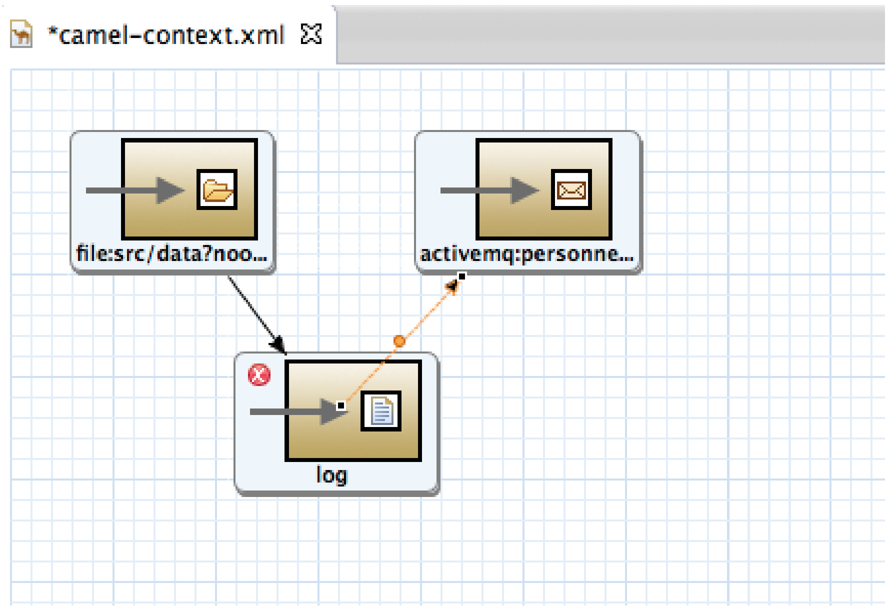
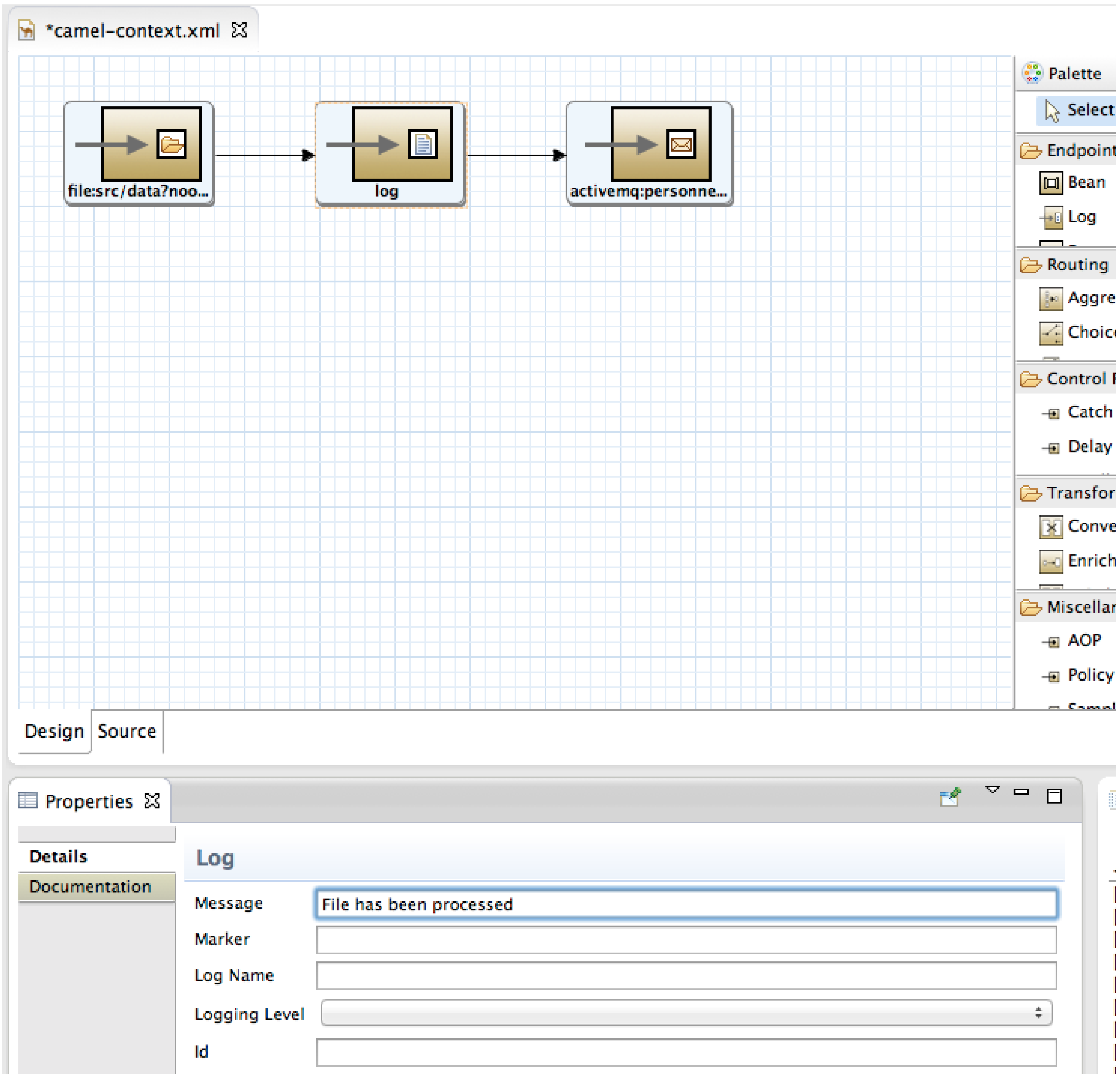


Figure 24. Sample - Activemq Log

7. Review the message log.



Log

Message	File has been processed
Marker	
Log Name	
Logging Level	
Id	

Figure 25. Sample - Message Log

8. Rearrange the icons neatly and save the modified `camel-context.xml` file.

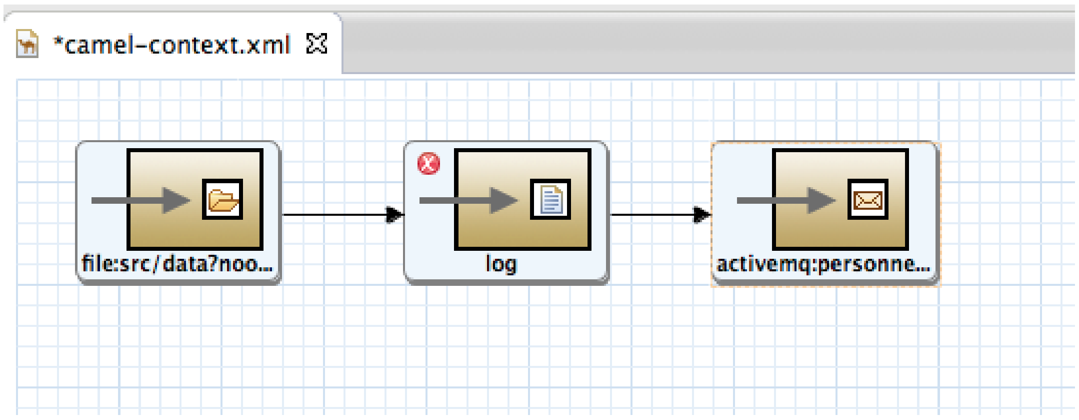
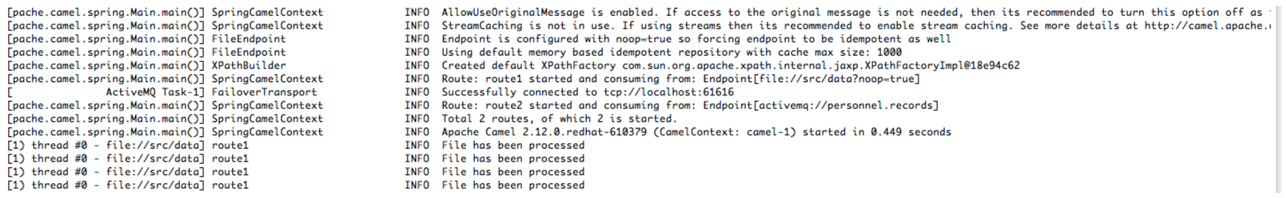


Figure 26. Sample - Exercise Result with Icons

9. Click **Run as** → **Local Camel Context** to run the project again.
10. Verify that a message appears at least four times in the Maven console.



```
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  AllowUseOriginalMessage is enabled. If access to the original message is not needed, then its recommended to turn this option off as
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  StreamCaching is not in use. If using streams then its recommended to enable stream caching. See more details at http://camel.apache.
[pache.camel.spring.Main.main()] FileEndpoint                  INFO  Endpoint is configured with noop=true so forcing endpoint to be idempotent as well
[pache.camel.spring.Main.main()] FileEndpoint                  INFO  Using default memory based idempotent repository with cache max size: 1000
[pache.camel.spring.Main.main()] XPathBuilder                  INFO  Created default XPathFactory com.sun.org.apache.xpath.internal.jaxp.XPathFactoryImpl@18e94c62
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  Route: route1 started and consuming from: Endpoint[file://src/data?noop=true]
[               ActiveMQ Task-1] FailoverTransport             INFO  Successfully connected to tcp://localhost:61616
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  Route: route2 started and consuming from: Endpoint[activemq://personnel.records]
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  Total 2 routes, of which 2 is started.
[pache.camel.spring.Main.main()] SpringCamelContext            INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) started in 0.449 seconds
[1) thread #0 - file://src/data] route1                        INFO  File has been processed
[1) thread #0 - file://src/data] route1                        INFO  File has been processed
[1) thread #0 - file://src/data] route1                        INFO  File has been processed
[1) thread #0 - file://src/data] route1                        INFO  File has been processed
```

Figure 27. Sample - Results in Maven Console

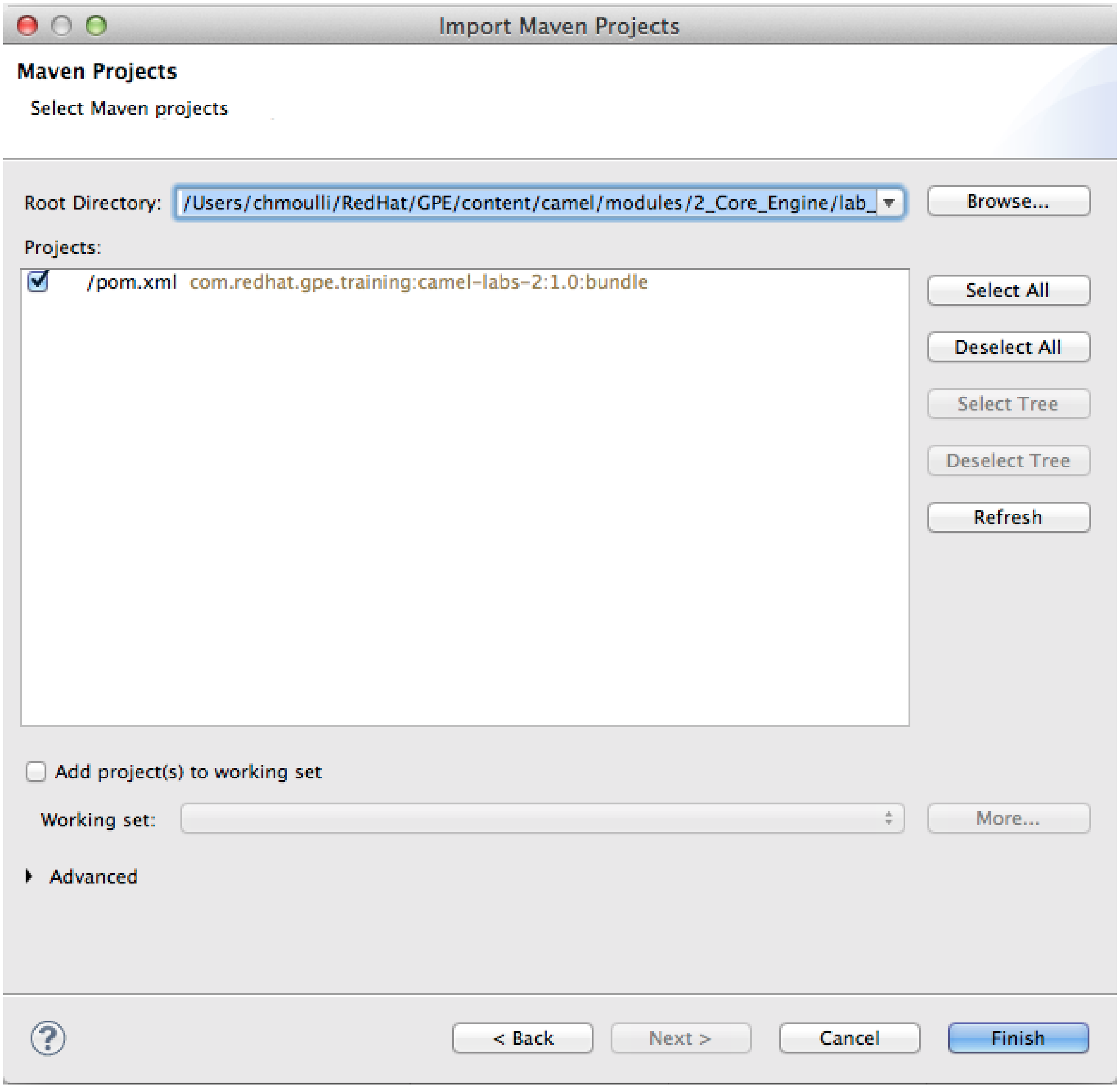
11. In **Project Explorer**, right-click the `demo1-camel` project and then select **Close project**.

1.2. Design a New Project

The goal of this exercise is to design a new integration project and create an Apache Camel Route using the Camel Editor. To do this, you use an existing Maven project and add to it a Camel route, a HelloBean, and business logic that sends a message to the console.

1.2.1. Import the Maven Project if not yet done

1. Open **JBoss Developer Studio**.
2. Import the `camel-labs-VERSION-NUMBER-excercise` Apache Maven project:
   a. From the menu, Select **File** → **Import**.
   b. Click **Maven** → **Existing Maven Projects**.


Import Maven Projects

Maven Projects

Select Maven projects

Root Directory: /Users/chmoulli/RedHat/GPE/content/camel/modules/2_Core_Engine/lab_

Projects:

☑ /pom.xml com.redhat.gpe.training:camel-labs-2:1.0:bundle

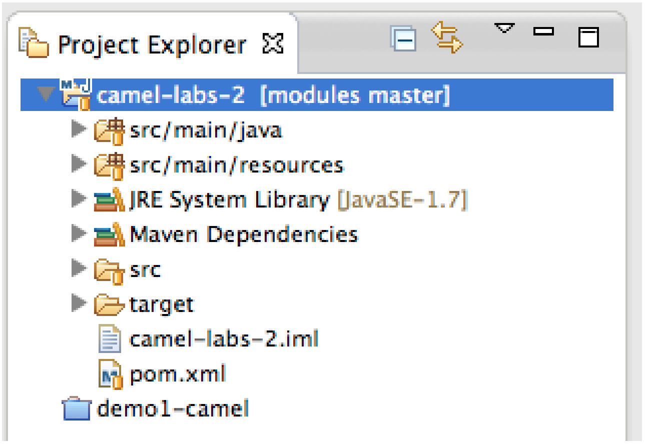
☐ Add project(s) to working set

Working set:

Figure 28. Import Maven Projects window

c. Click **Finish**.

3. In **Project Explorer**, confirm that the `camel-lab-2` project was imported.


Project Explorer

camel-labs-2 [modules master]
- src/main/java
- src/main/resources
- JRE System Library [JavaSE-1.7]
- Maven Dependencies
- src
- target
- camel-labs-2.iml
- pom.xml

demo1-camel

Figure 29. Sample - Imported Project

4. Review the project contents:

   - `pom.xml`

- `HelloBean` class

1.2.2. Add the `sayHello` Method to the Bean

Before you design the Apache Camel Route, you must add a `sayHello` method to the body of the class `HelloBean`. This method will be used by the route. The method accepts an exchange as an input parameter and returns a String.

1. Open the `HelloBean` class and add a `sayHello` method.

   ```
   public String sayHello(Exchange exchange) {
    String body = (String)exchange.getIn().getBody();
    return "Hello world ! " + body;
   }
   ```

The return type and the object returned will be used by the Apache Camel engine to add a Body to the Exchange with the String object `Hello world ! xxxx"`

1.2.3. Design the Apache Camel Route

You use the Camel Editor to create a new route that triggers a timer event at a specified interval (every X seconds). To design the route, you will complete these tasks:

- Create a new CamelContext.
- Edit the Endpoint and SetBody properties.
- Add a bean tag and Bean processor to the project.
- Add a log processor to the project.

  1. Create a new CamelContext:

     a. Open the `Fuse Integration perspective`.

     b. In **Project Explorer** (left panel), expand the `src/main/resources/OSGI-INF/blueprint` folder.

     c. Right-click the `blueprint folder` to select it.

     d. Select `New -> Camel XML File`.

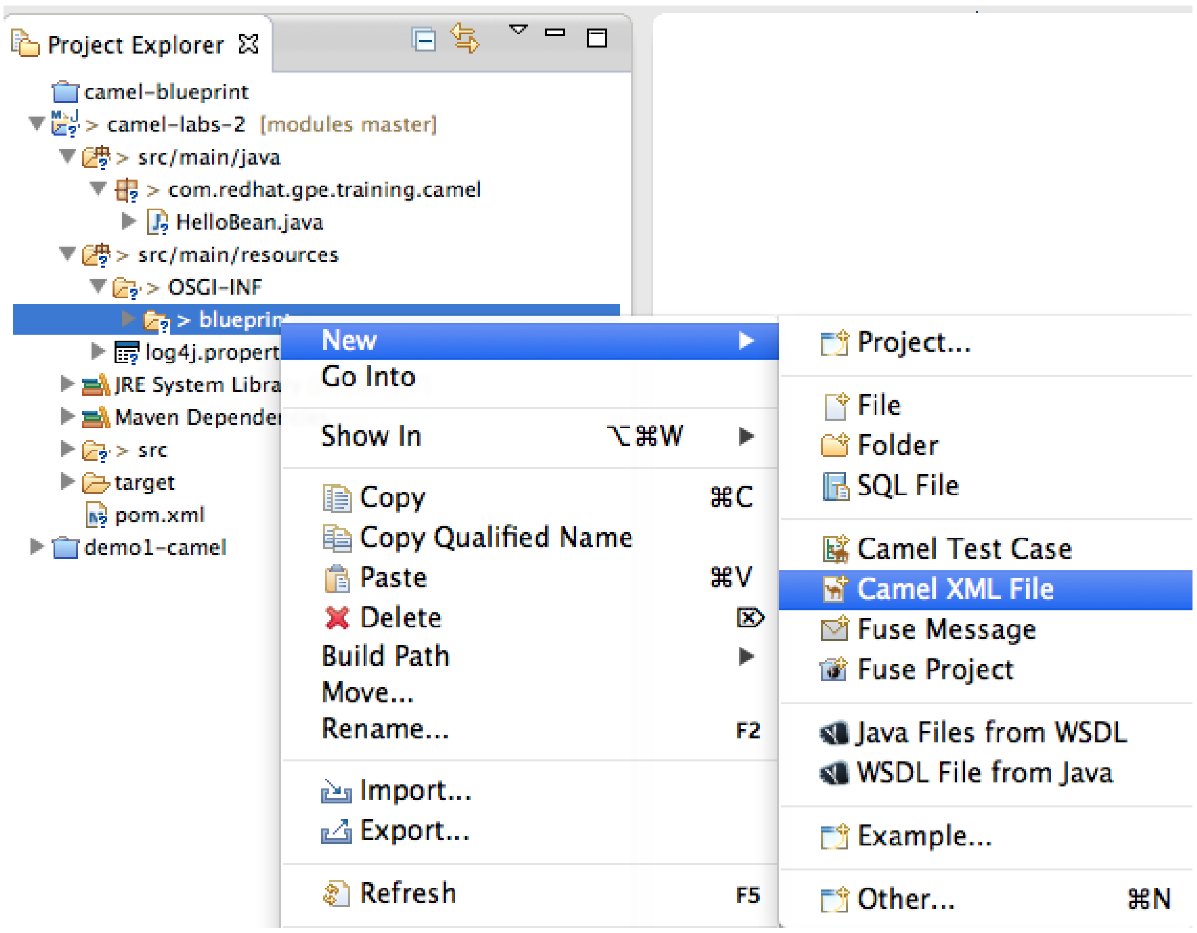


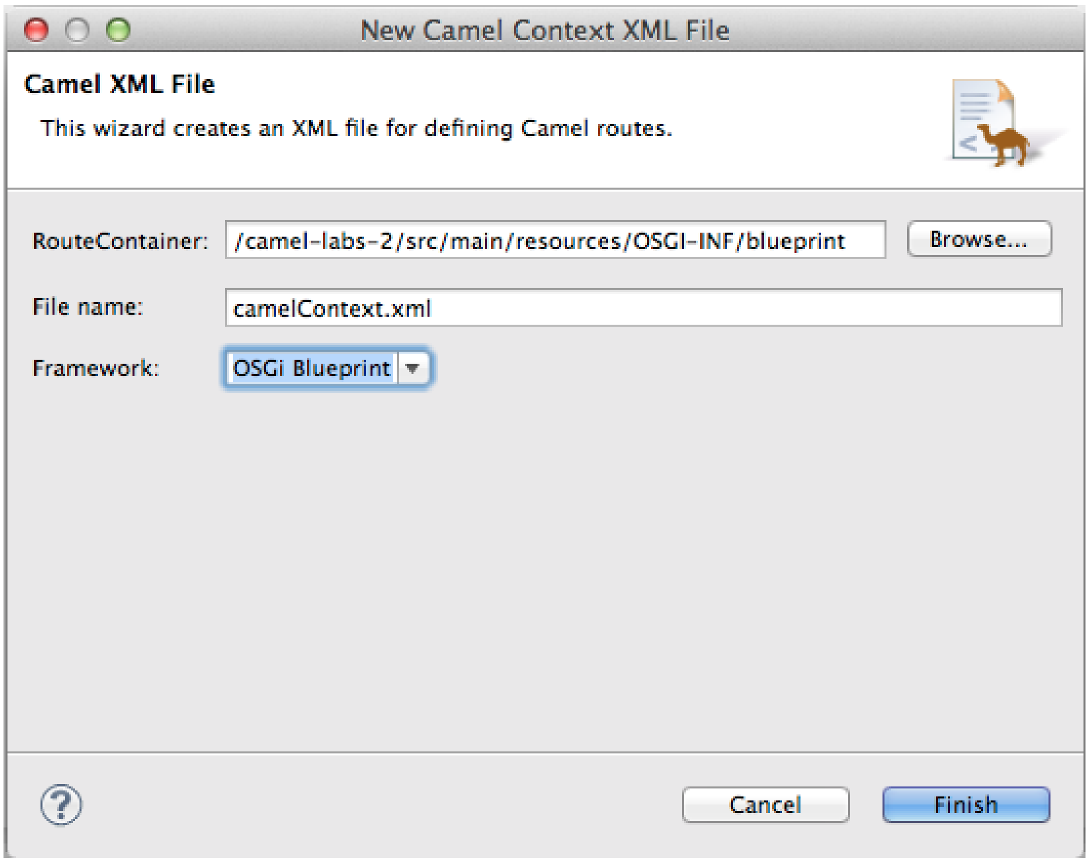
Figure 30. Project Explorer - New Camel XML file

e. For the **Framework**, select `OSGi Blueprint`, and then click **Finish**.

f. Inspect the folder to confirm that a new `camelContext.xml` was added.

2. Edit the Endpoint and SetBody properties:

   a. Double-click the `camelContext.xml` file you just created to open it in the Camel Editor.


New Camel Context XML File

Camel XML File

This wizard creates an XML file for defining Camel routes.

RouteContainer: /camel-labs-2/src/main/resources/OSGI-INF/blueprint

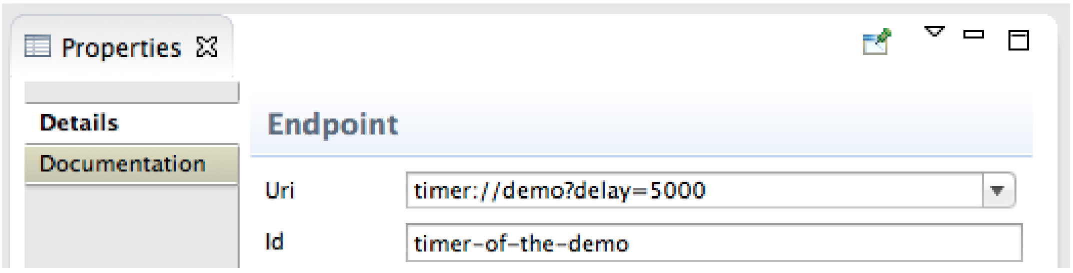
File name: camelContext.xml

Framework: OSGi Blueprint

Figure 31. Camel Route Blueprint

b. From the **Components** section of the **Palette**, select the **timer** endpoint and drag it onto the project.

c. For the **Uri** field, select this text to define the URI: `timer://demo?delay=5000`


Properties

Endpoint

Uri timer://demo?delay=5000

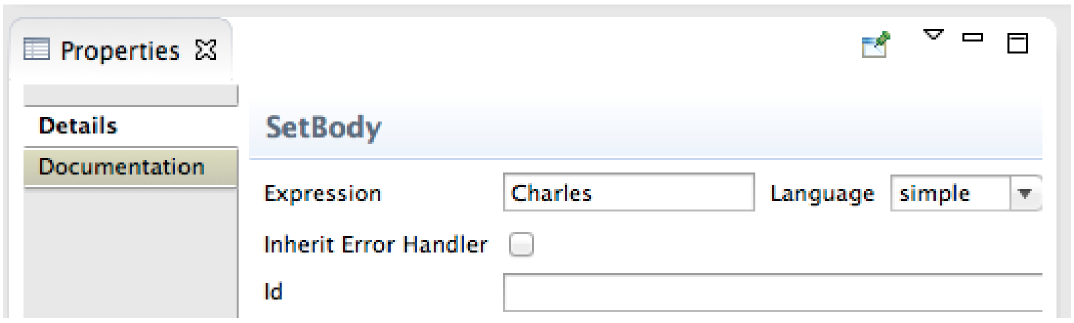
Id timer-of-the-demo

Figure 32. Timer Endpoint

d. From the **Transformation** section of the **Palette**, select the **SetBody** processor and drag it onto the project.

e. Edit the properties of this processor to add an expression and set the language:

- Enter your name for the **Expression**.
- For the **Language**, select `simple`.



SetBody

Expression: Charles
Language: simple
Inherit Error Handler: ☐
Id:

Figure 33. SetBody Processor

3. Add the bean tag and bean processor:

   a. Switch to the source of the Camel route to add the following `<bean>` tag. This is required to instantiate the singleton bean, when the Blueprint context is created.

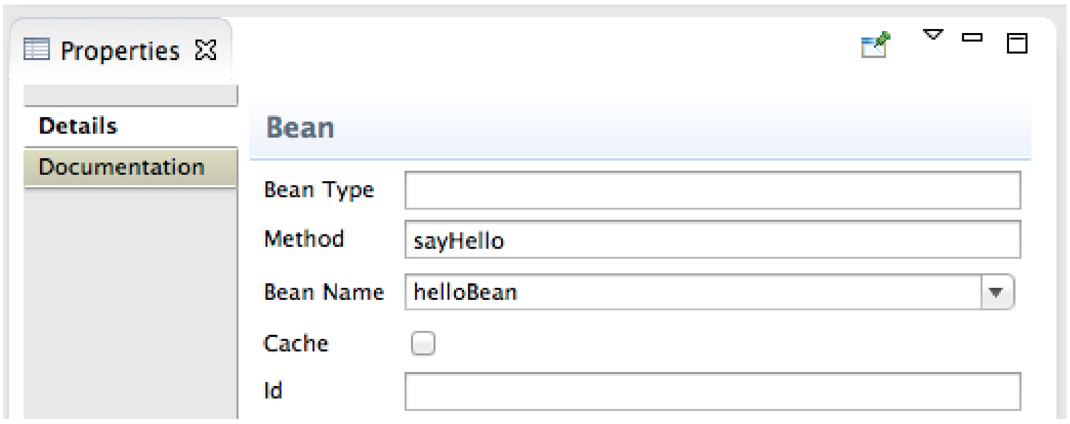
```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
       xmlns:camel="http://camel.apache.org/schema/blueprint"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
       http://camel.apache.org/schema/blueprint
http://camel.apache.org/schema/blueprint/camel-blueprint.xsd">

  <bean id="helloBean" class="com.redhat.gpe.training.camel.HelloBean"/>
```

   b. Return to the **Design** view.

   c. From the **Components** section of the **Palette**, select a **Bean** processor and drag it onto the project.

   d. Edit the properties of the Bean processor:

      - Enter `sayHello` for the **Method**.
      - Select `helloBean` for the the **Bean Name**.



Bean

Bean Type:
Method: sayHello
Bean Name: helloBean
Cache: ☐
Id:

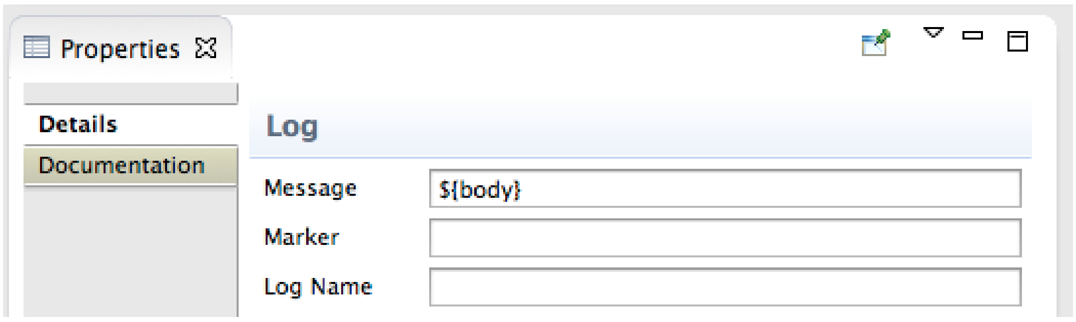
Figure 34. Bean Endpoint

4. Add a log processor:

   a. From the **Components** section of the **Palette**, select a **Log** processor and drag it onto the project.

b. Edit the properties of this Log processor to add a simple expression that extracts the contents of the Exchange body:

   - For the **Message**, enter `${body}`.



Properties

Details

Documentation

Log

Message: ${body}

Marker:

Log Name:

Figure 35. Log Endpoint

c. Connect the endpoints and processors.

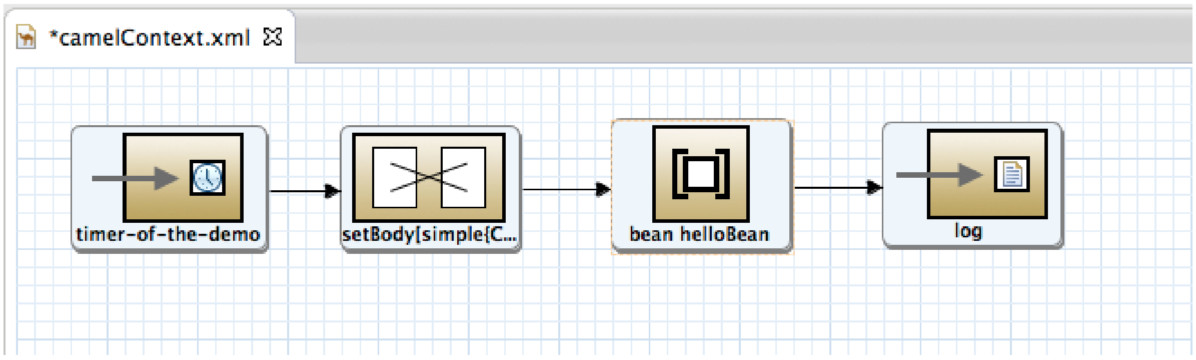


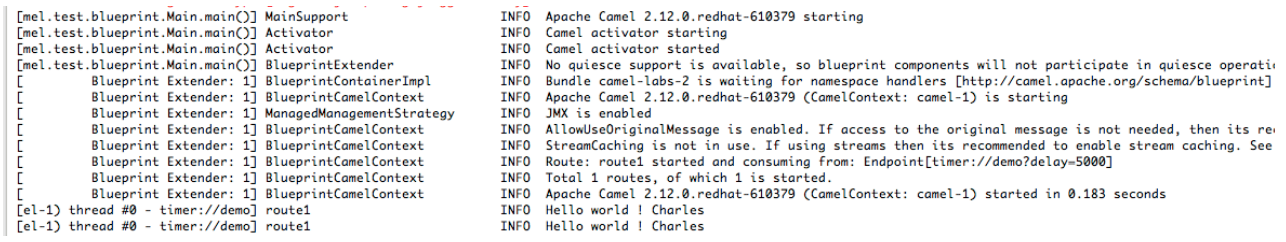
Figure 36. Route Blueprint - Connect Objects

d. Save the modified Camel route and verify the result of the XML DSL route generated:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
       xmlns:camel="http://camel.apache.org/schema/blueprint"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
       http://camel.apache.org/schema/blueprint
http://camel.apache.org/schema/blueprint/camel-blueprint.xsd">

  <bean id="helloBean" class="com.redhat.gpe.training.camel.HelloBean"/>

  <camelContext trace="false"
xmlns="http://camel.apache.org/schema/blueprint">
    <route>
        <from uri="timer://demo?delay=5000" id="timer-of-the-demo">
            <description/>
        </from>
        <setBody>
            <simple>Charles</simple>
        </setBody>
        <bean ref="helloBean" method="sayHello"/>
        <log message="${body}"/>
    </route>
</camelContext>

</blueprint>
```

1.2.4. Run the Project Locally

1. Right-click the `camel-context.xml` file and select **Run as → Local Camel Context**.
2. Verify that messages appear in the log.



```
[mel.test.blueprint.Main.main()] MainSupport                     INFO  Apache Camel 2.12.0.redhat-610379 starting
[mel.test.blueprint.Main.main()] Activator                       INFO  Camel activator starting
[mel.test.blueprint.Main.main()] Activator                       INFO  Camel activator started
[mel.test.blueprint.Main.main()] BlueprintExtender               INFO  No quiesce support is available, so blueprint components will not participate in quiesce operati
[         Blueprint Extender: 1] BlueprintContainerImpl          INFO  Bundle camel-labs-2 is waiting for namespace handlers [http://camel.apache.org/schema/blueprint]
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) is starting
[         Blueprint Extender: 1] ManagedManagementStrategy       INFO  JMX is enabled
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  AllowUseOriginalMessage is enabled. If access to the original message is not needed, then its re
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  StreamCaching is not in use. If using streams then its recommended to enable stream caching. See
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  Route: route1 started and consuming from: Endpoint[timer://demo?delay=5000]
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  Total 1 routes, of which 1 is started.
[         Blueprint Extender: 1] BlueprintCamelContext           INFO  Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) started in 0.183 seconds
[el-1) thread #0 - timer://demo] route1                          INFO  Hello world ! Charles
[el-1) thread #0 - timer://demo] route1                          INFO  Hello world ! Charles
```

Figure 37. Sample - Log Messages

You have completed this lab.

Last updated 2015-11-12 12:04:12 EST