

F4: External Integrations

Introducing Camel

What is Camel?

Good question. See: <http://camel.apache.org/what-is-camel.html>

Too many buzzwords - what exactly is Camel?

Okay, so the description above is technology focused. There's a great discussion about Camel at **Stack Overflow**.

So really you want see this: <http://stackoverflow.com/questions/8845186/what-exactly-is-apache-camel>

In short...

- Camel is a framework for creating small message based applications... and then some.
- Camel formalizes working with messages so well it can be described in multiple formats: Java, Spring/Blueprint XML, and Scala.
- Camel is all the code you should not have to write in order to work with queues, files, databases, RESTful APIs, common data formats, command line utilities, etc... in a consistent and reliable manner.

Camel can run...

- As a stand-alone Java application
- In a servlet container like Tomcat or Jetty
- In an OSGi runtime such as Karaf

What is OSGi?

- Open Service Gateway Initiative
- Framework for modularizing and deploying Java applications
 - Hot deployment
 - Automatic reloading of configuration
 - Sophisticated dependency resolution
 - XML scripting for complex deployments (features)

Hot Deployment

Bundles can be started, stopped, updated, etc... at runtime!

In other words:

**YOU DO NOT HAVE TO RESTART
YOUR SERVER TO UPDATE CODE OR
CONFIGURATION**

Terminology

- Apache Camel
 - Endpoints
 - Components
 - Messages
 - Routes
- Apache Karaf -- OSGi
 - Bundles
 - Features

Available Camel Components

<http://camel.apache.org/components.html>

- ActiveMQ
- AWS SQS
- DropBox
- System calls
- Local files
- FTP
- HTTP resources
- LDAP
- SMTP
- SQL
- Twitter
- etc, etc, etc

Hands-On: Gitting the Examples

> vagrant ssh

or:

> ssh -p 2222 vagrant@localhost

password = vagrant

Hands-On: Inspect features

```
> /opt/karaf/bin/client
```

```
>> feature:list | grep fcrepo
```

```
fcrepo-camel
```

```
fcrepo-indexing-triplestore
```

```
fcrepo-audit-triplestore
```

```
fcrepo-indexing-solr
```

```
fcrepo-reindexing
```

```
fcrepo-fixity
```

Hands-On: Helpful Commands

```
>> feature:install fcrepo-audit-triplestore
```

```
>> feature:stop <whichever>
```

```
>> camel:route-list
```

```
>> bundle:list | grep fcrepo
```

```
>> ctrl-d
```

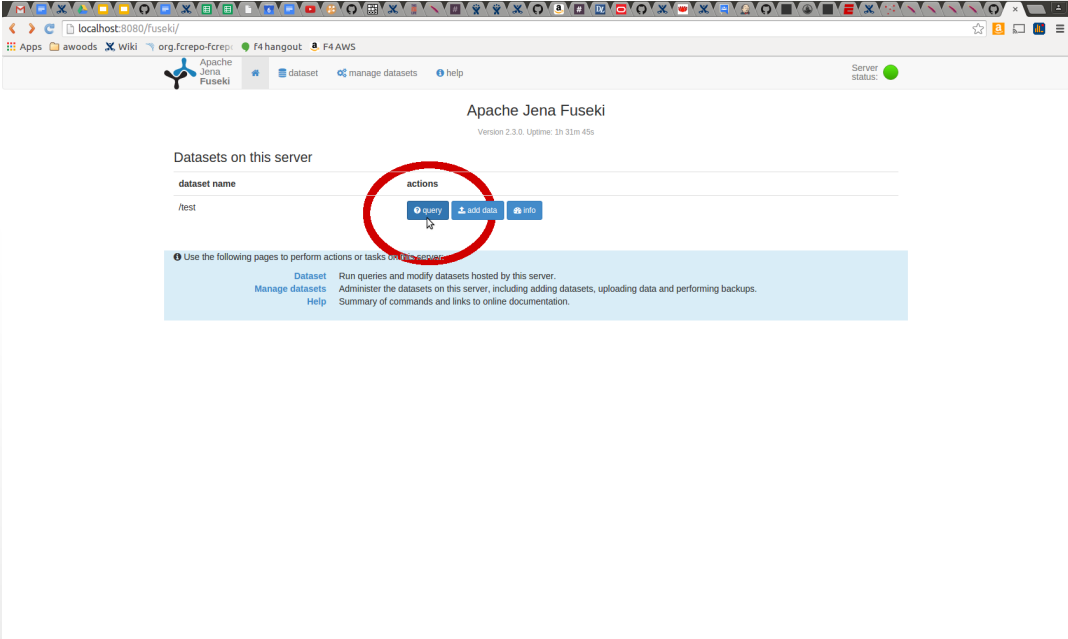
Hands-On: Watch the log

In a new vagrant ssh terminal:

```
> sudo tail -f /opt/karaf/data/log/karaf.log
```

Hands-On: Indexing in triplestore

http://localhost:8080/fuseki



The screenshot shows the Apache Jena Fuseki web interface. The browser address bar displays 'localhost:8080/fuseki/'. The page title is 'Apache Jena Fuseki' with the version '2.3.0' and 'Uptime: 1h 31m 45s'. The main content area is titled 'Datasets on this server' and contains a table with one dataset named '/test'. The 'actions' column for this dataset has three buttons: 'query', 'upload data', and 'info'. The 'query' button is circled in red. Below the table, there is a blue box with instructions: 'Use the following pages to perform actions or tasks on the server: Dataset Run queries and modify datasets hosted by this server. Manage datasets Administer the datasets on this server, including adding datasets, uploading data and performing backups. Help Summary of commands and links to online documentation.'

dataset name	actions
/test	query upload data info

Use the following pages to perform actions or tasks on the server:

- [Dataset](#) Run queries and modify datasets hosted by this server.
- [Manage datasets](#) Administer the datasets on this server, including adding datasets, uploading data and performing backups.
- [Help](#) Summary of commands and links to online documentation.

Hands-On: Indexing in triplestore

```
select * where {  
  <http://localhost:8080/fcrepo/rest/cover> ?p ?o  
}
```

Hands-On: Indexing in triplestore

PREFIX ldp: <http://www.w3.org/ns/ldp#>

PREFIX ebucore: <http://www.ebu.
ch/metadata/ontologies/ebucore/ebucore#>

```
select * where {  
  ?s ldp:contains ?o .  
  ?o ebucore:hasMimeType ?m  
}
```

Hands-On: Indexing in triplestore

Audit

prefix premis: <http://www.loc.gov/premis/rdf/v1#>

prefix xsd: <http://www.w3.org/2001/XMLSchema#>

```
select ?s ?d where {
```

```
  ?s ?p <http://fedora.info/definitions/v4/audit#InternalEvent> .
```

```
  ?s premis:hasEventRelatedObject <http://localhost:8080/fcrepo/rest/cover> .
```

```
  ?s premis:hasEventDateTime ?d .
```

```
  FILTER (?d > "2015-10-06T04:21:14Z"^^xsd:dateTime)
```

```
}
```


Hands-On: Indexing in Solr

http://localhost:8080/solr

The screenshot displays the Apache Solr Admin UI. The left sidebar contains navigation links: Dashboard, Logging, Core Admin, Java Processes, and Thread Dump. The main content area is divided into several sections:

- Instance:** Shows the instance started 'about an hour ago'.
- Versions:** A table listing installed versions:

Component	Version
solr-spec	4.10.3
solr-impl	4.10.3 1644336 - mark - 2014-12-10 00:35:44
lucene-spec	4.10.3
lucene-impl	4.10.3 1644336 - mark - 2014-12-10 00:28:00
- System:** A dashboard of system metrics with progress bars:
 - Physical Memory: 62.4% (1.22 GB / 1.96 GB)
 - Swap Space: 0.0%
 - File Descriptor Count: 3.3% (143 / 4096)
 - JVM-Memory: 65.4% (80.99 MB / 123.75 MB)
- JVM:** Details for the Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8.0_60 25.60-b23):
 - Runtime Processors: 1
 - Args: `-Djava.io.tmpdir=/tmp/tomcat7-tomcat7-tmp`, `-Dcatalina.home=/var/lib/tomcat7`, `-Dcatalina.base=/var/lib/tomcat7`, `-Djava.endorsed.dirs=/usr/share/tomcat7/endorsed`, `-Dcore.repo.audit.container=audit`, `-XX:+UseConcMarkSweepGC`, `-Xmx128m`, `-Djava.net.headless=true`, `-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager`, `-Djava.util.logging.config.file=/var/lib/tomcat7/conf/logging.properties`

A red circle highlights the 'Core Selector' dropdown menu in the left sidebar, which currently shows 'collection1'.

Hands-On: Indexing in Solr

The screenshot shows the Apache Solr Admin UI for a collection named 'collection1'. The left sidebar contains a navigation menu with the following items: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, collection1 (dropdown), Overview (selected), Analysis, Dataimport, Documents, HLC, Ping, Plugins / Stats, Query (circled in red), Replication, and Schema Browser. The main content area is divided into several sections:

- Statistics:** Last Modified: about 12 hours ago, Num Docs: 7, Max Docs: 11, Heap Memory Usage: 35136, Deleted Docs: 4, Version: 21, Segment Count: 6. Includes an 'optimize now' button.
- Instance:** CWD: /var/lib/tomcat7, Instance: /var/lib/tomcat7/solr/collection1, Data: /var/lib/tomcat7/solr/collection1/data, Index: /var/lib/tomcat7/solr/collection1/data/index, Impl: org.apache.solr.core.NRTCachingDirectoryFactory.
- Replication (Master):** A table showing replication status for the Master (Searching) and Master (Replicable).
- Healthcheck:** Ping request handler is not configured with a healthcheck file.
- Admin Extra:** A section for additional administrative actions.

At the bottom of the page, there are links for Documentation, Issue Tracker, IRC Channel, Community forum, and Solr Query Syntax. The browser address bar shows 'localhost:8080/solr/#/collection1' and the footer shows 'localhost:8080/solr/#/collection1/query'.

Hands-On: Reindexing - prep

> sudo service tomcat7 stop

> sudo rm -rf /etc/fuseki/databases/test_data/*

> sudo service tomcat7 start

Hands-On: Reindexing

```
> curl -XPOST localhost:9080/reindexing/cover  
-H"Content-Type: application/json" -d  
'["activemq:queue:triplestore.reindex"]'
```

Hands-On: Fixity

```
> curl -XPOST localhost:9080/reindexing/cover  
-H"Content-Type: application/json" -d  
'["activemq:queue:fixity"]'
```

```
> less /tmp/fixityErrors.log
```

Bonus Round

Git the examples

-- from within vagrant --

> cd

> git clone <https://github.com/awoods/fcrepo-camel-workshop.git>

Hands-On: Install a Feature

```
>> feature:install camel-exec
```

This installs the camel-exec feature...

which we will use later on for executing command-line utilities.

1 - Make our own “feature”

Use case:

- Whenever a JPEG image is ingested in Fedora, generate a thumbnail

01: Hello World

```
<camelContext id="helloWorld">
  <route id="timerToLog">
    <from uri="timer:foo?period=5000"/>
      <setBody>
        <simple>Hello Whirled!</simple>
      </setBody>
    <to uri="log:demo"/>
  </route>
</camelContext>
```

Deploy the first route

```
> sudo cp /home/vagrant/fcrepo-camel-workshop/01-HelloWorld.xml /opt/karaf/deploy/
```

Routes

- Processing pipelines, with beginning and ending points
- Begin with a 'from' uri
 - A file
 - A queue
 - A timer
 - Another route
- Other routes are called by using 'to' and a uri
 - `<to uri="direct:myRoute"/>`

Camel Components / Endpoints

- From previous example
 - “timer:foo?period=5000”
- Prefixes in camel URIs are Components
- Components provide endpoints for communicating with other software
- Two types of endpoints
 - Producer
 - Consumer
- Options can be provided

Component: fcrepo-camel

- fcrepo-camel project provides an F4 component
- Used as a producer to interact with F4's REST API
- URIs look like
 - `fcrepo:hostname[:port][/resourceUrl][?options]`
- Used as a consumer to handle messages published by F4

02: Responding to Fedora events

```
<route id="thumbnailRouter">  
  <from uri="activemq:topic:fedora"/>  
  <log message="GOT A MESSAGE FROM FEDORA"/>  
  <log message="HEADERS: ${headers}"/>  
</route>
```

Anatomy of a Message

- **Body**

- The main content of message. Can be any text or binary data type:
 - html, json, xml, etc...
 - image, audio, video, etc...

- **Headers**

- Key/Value properties for the message
 - HTTP: Accept, Content-Type, etc...
 - JMS: Timestamp, Expiry, Destination, etc...

Fedora Headers

- org.fcrepo.jms.baseURL
- org.fcrepo.jms.identifier
- org.fcrepo.jms.eventType
- org.fcrepo.jms.properties
- org.fcrepo.jms.timestamp
- and more...

03: Getting RDF from Fedora

```
<route id="thumbnailRouter">  
  <from uri="activemq:topic:fedora"/>  
  <to uri="fcrepo:localhost:8080/fcrepo/rest"/>  
  <log message="{body}"/>  
</route>
```

Watch out!

If you are scripting in XML, remember that you are still in XML! ...and have to escape your xml entities.

So things like:

- $x < y$
- this && that

Become:

- $x \< y$
- this && that

04: Filtering using headers

```
<from uri="activemq:topic:fedora"/>
```

```
<filter>
```

```
<simple>
```

```
  ${header[org.fcrepo.jms.eventType]} != 'http://fedora.  
info/definitions/v4/repository#NODE_REMOVED' & amp; & amp;
```

```
  ${header[org.fcrepo.jms.properties]} contains 'http://fedora.  
info/definitions/v4/repository#hasContent'
```

```
</simple>
```

```
<log message="Got an upsert event with content!"
```

```
</filter>
```

05: Filtering on body content

```
<route id="thumbnailRouter">
  <from uri="activemq:topic:fedora"/>
  <to uri="fcrepo:localhost:8080/fcrepo/rest"/>
  <setProperty propertyName="mimetype">
    <xpath>/rdf:RDF/rdf:Description/ebucore:hasMimeType/text()</xpath>
  </setProperty>
  <filter>
    <simple>${property.mimetype} == 'image/jpeg'</simple>
    <log message="JPEG!"/>
  </filter>
</route>
```

06a: Getting Non-Rdf Content

```
<route id="thumbnailRouter">  
  <from uri="activemq:topic:fedora"/>  
  <to uri="fcrepo:localhost:8080/fcrepo/rest?metadata=false"/>  
  <log message="\${body}"/>  
</route>
```

06b: Generating a Thumbnail

```
<route id="thumbnailGenerate">  
  <from uri="activemq:topic:fedora"/>  
  <to uri="fcrepo:localhost:8080/fcrepo/rest?metadata=false"/>  
  <to uri="exec:convert?args=-thumbnail 100x100 - png:-"/>  
  <log message="THUMBNAIL: ${body}"/>  
</route>
```

Put it all together

- Ingest a JPEG file
- Derivative thumbnail at `/tmp/cover/<>.png`
- View thumbnail:
 - > `cp /tmp/cover/<>.png /vagrant`
- View `<>.png` outside of vagrant

Success!