External Integrations

Introducing Camel
Repository Scope

What Fedora can do

What your repository system can do
- indexing
- text-based search
- analytics
- etc
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 is middleware for integration with external systems

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.
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
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
Camel Examples
from("activemq:topic:fedora")
  .to("http4:localhost/api");
Transformation

```java
from("activemq:topic:fedora")
  .process(new SparqlUpdater())
  .to("http4:localhost/api");
```
from("activemq:topic:fedora")
  .filter(header("org.fcrepo.jms.identifier").startsWith("/audit"))
  .to("http4:localhost/api");
Triplestore
Fedora 3: internal triplestore

Resource index (mulgara)

Very useful

Very slow

Very resource intensive
Fedora 4: no internal triplestore

This keeps fedora running faster

There is less code to maintain in the fedora core

The fedora API is simpler

Not everyone uses a triplestore

Decoupled components are easier to scale out
Mirror Fedora content in a triplestore

Two approaches:

1) Synchronous: client(s) operate on content in fedora and then do the same in a triplestore

2) Asynchronous: client(s) operate on content only in fedora, a separate process synchronizes changes
Asynchronous Patterns

Camel recipe for this (DSL for integration patterns)

fcrepo-indexing-triplestore in github. com/fcrepo4-exts/fcrepo-camel-toolbox
Diagram View

Client

Fedora

JMS / Camel

Triplestore

Client
Indexing to a Triplestore

```java
from("activemq:topic:fedora")
  .to("fcrepo:{{fcrepo.baseUrl}}?accept=application/n-triples")
  .process(new SparqlUpdateProcessor())
  .to("http4://{{triplestore.baseUrl}}");
```
Choose your own triplestore

Fuseki comes with the fcrepo-vagrant VM

In production, any will work. These have been tested:
● Sesame/OpenRDF
● Jena/Fuseki
● Blazegraph

...just update the location of the triplestore in the camel configuration
SPARQL & RDF Triples: S P O

Syntax like English (sort of):

- Statements end with a period
- Clauses end with a semicolon
- Lists are separated with a comma
Hands-On: Indexing in triplestore

http://localhost:8080/fuseki
Hands-On: Indexing in triplestore

```sparql
select * where {
}
```
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 {
  ?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 premis:hasEventDateTime ?d .
  FILTER (?d > "2015-10-06T04:21:14Z"^^xsd:dateTime)
}
Solr
Indexing to Solr

```java
from("activemq:topic:fedora")
 .match()
 .when(header("org.fcrepo.jms.eventType"))
 .isEqualTo("http://fedora.info/definitions/v4/repository#NODE_REMOTE_DELETE")
 .thenReturn("direct:delete")
 .otherwise()
 .thenReturn("direct:index");

from("direct:delete")
 .process(new SolrDeleteProcessor())
 .thenReturn("http4://localhost:8983/solr/collection1/update");
rom("direct:index")
 .thenReturn("fcrepo:localhost:8080/rest?transform=default")
 .thenReturn("http4://localhost:8983/solr/collection1/update");
```
Hands-On: Indexing in Solr

http://localhost:8080/solr
Hands-On: Indexing in Solr
Success!