10 Novel Ways To Use Alfresco

Gethin James

1. Riding a Camel

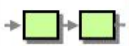
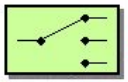
Enterprise Integration Patterns

Camel supports most of the Enterprise Integration Patterns from the excellent book by Gregor Hohpe and Bobby Woolf.

If you are new to Camel you might want to try the Getting Started in the User Guide before attempting to implement these patterns.

The EIP icons library is available as a Visio stencil file adapted to render the icons with the Camel color : sand. Download it here for your presentation, functional and technical analysis documents. The original EIP stencil is also available in OpenOffice 3.x Draw (thanks to Marco Garbelini) , Microsoft Visio, or Omnigraffle.

Messaging Systems

	Message Channel	How does one application communicate with another using messaging?
	Message	How can two applications connected by a message channel exchange a piece of information?
	Pipes and Filters	How can we perform complex processing on a message while maintaining independence and flexibility?
	Message Router	How can you decouple individual processing steps so that messages can be passed to different filters depending on a set of conditions?
	Message	How can systems using different data formats communicate with each

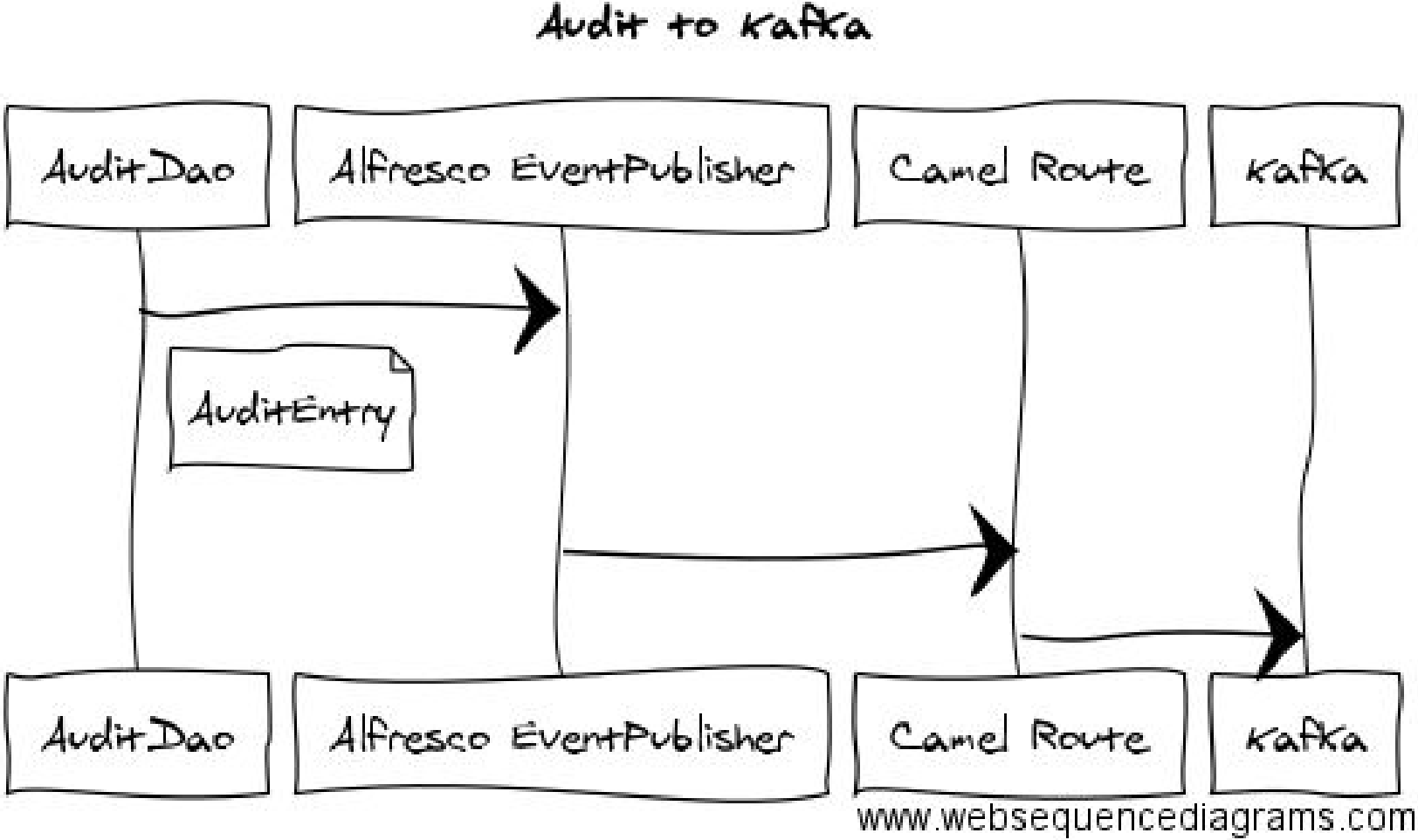
Overview
- Home
- Download
- Getting Started
- FAQ

Documentation
- User Guide
- Manual
- Books
- Tutorials
- Examples
- Cookbook
- Architecture
- Enterprise Integration Patterns
- DSL
- Components
- Data Format
- Languages
- Security
- Security Advisories

Search
Google™ Custom Search
Search

Community
- Support
- Contributing
- Discussion Forums
- Mailing Lists
- User Stories
- News
- Articles
- Site
- Team
- Camel Extra

2. Audit to Kafka


Audit to kafka
AuditDao
Alfresco EventPublisher
Camel Route
kafka
AuditEntry
AuditDao
Alfresco EventPublisher
Camel Route
kafka
www.websequencediagrams.com

Audit to Kafka

Audit Data as Events

https://github.com/covolution/alfresco-audit-events

Kafka jars

https://github.com/covolution/alfresco-kafka-integration

Docker

alfresco-kafka-integration/src/docker/readme.txt

Alfresco Audit Data as Events

Takes Alfresco Audit Data and sends it as Events

Overview

- alfresco-events-audit - Jar that contains the AuditEvent bean (needed by the listener for deserialization)
- alfresco-messaging-audit - Configures the Alfresco messaging system with a new audit topic
- audit-events-amp - Wraps the projects in an amp, for installing in Alfresco.

Quick start

- `mvn install` and the repo amp is in audit-events-amp/target

The Sample Listener

The sample listener is a standard logging listener just with a dependency on the alfresco-events-audit jar (for deserialization of messages)

- `cd sample-listener`
- `mvn install`
- `java -jar target/sample-listener-1.0.0-SNAPSHOT.jar --topic=alfresco.audit.events`

Alfresco global properties

I added this to alfresco-global.properties when running the amp.

```
echo '\n ##ActiveMQ Setup' >> $GLOBAL_PROPS \
 && echo 'messaging.broker.url=failover:(tcp://activemq:61616)?timeout=3000' >> $GLOBAL_PROPS \
 && echo 'messaging.events.repo.node.targetTopic.endpoint=log:MessagesAreDisgarded?level=OFF' >> $GLOBAL_PROPS \
 && echo 'audit.alfresco-access.enabled=true' >> $GLOBAL_PROPS \
 && echo 'messaging.subsystem.autoStart=true' >> $GLOBAL_PROPS \
 && echo 'events.subsystem.autoStart=true' >> $GLOBAL_PROPS
```

License

3. Machine Learning

Machine Learning Approaches

Collaborative / Social Approach

Usage - social graph

Content-based Approach

Metadata

Content

Collaborative Filtering

Collaborative filtering

Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. `spark.ml` currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. `spark.ml` uses the alternating least squares (ALS) algorithm to learn these latent factors. The implementation in `spark.ml` has the following parameters:

- *numBlocks* is the number of blocks the users and items will be partitioned into in order to parallelize computation (defaults to 10).
- *rank* is the number of latent factors in the model (defaults to 10).
- *maxIter* is the maximum number of iterations to run (defaults to 10).
- *regParam* specifies the regularization parameter in ALS (defaults to 1.0).
- *implicitPrefs* specifies whether to use the *explicit feedback* ALS variant or one adapted for *implicit feedback* data (defaults to `false` which means using *explicit feedback*).
- *alpha* is a parameter applicable to the implicit feedback variant of ALS that governs the *baseline* confidence in preference observations (defaults to 1.0).
- *nonnegative* specifies whether or not to use nonnegative constraints for least squares (defaults to `false`).

Note: The DataFrame-based API for ALS currently only supports integers for user and item ids. Other numeric types are supported for the user and item id columns, but the ids must be within the integer value range.

Explicit vs. implicit feedback

org.apache.spark.mllib.recommendation

Rating

Related Doc: package recommendation

```
case class Rating(user: Int, product: Int, rating: Double) extends Product with Serializable
```

A more compact class to represent a rating than Tuple3[Int, Int, Double].

Annotations @Since("0.8.0")

Source ALS.scala

▶ Linear Supertypes

Ordering Alphabetic By Inheritance

Inherited Rating Serializable Serializable Product Equals AnyRef Any

Hide All Show All

Visibility Public All

Instance Constructors

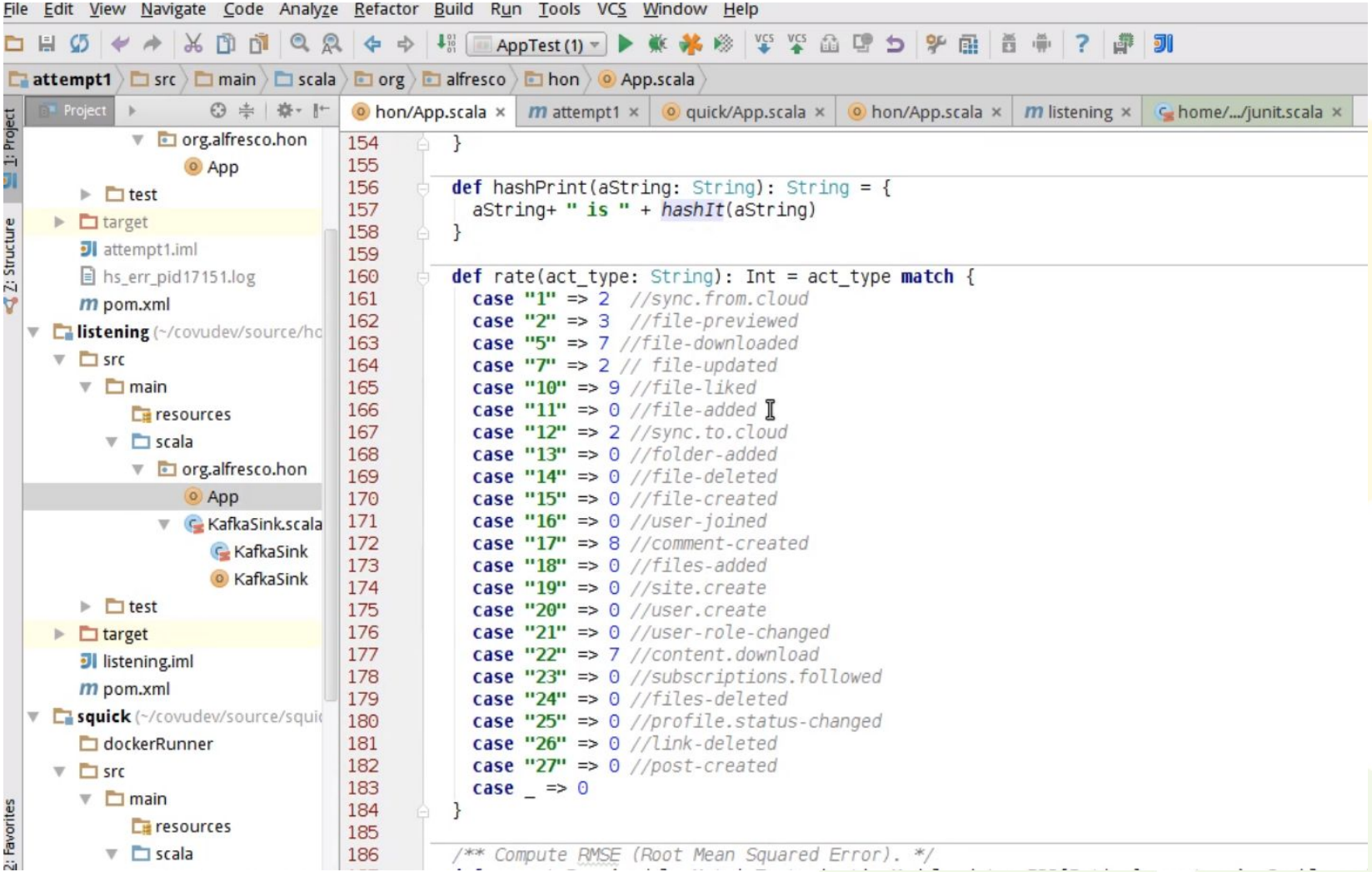
▶ `new Rating(user: Int, product: Int, rating: Double)`

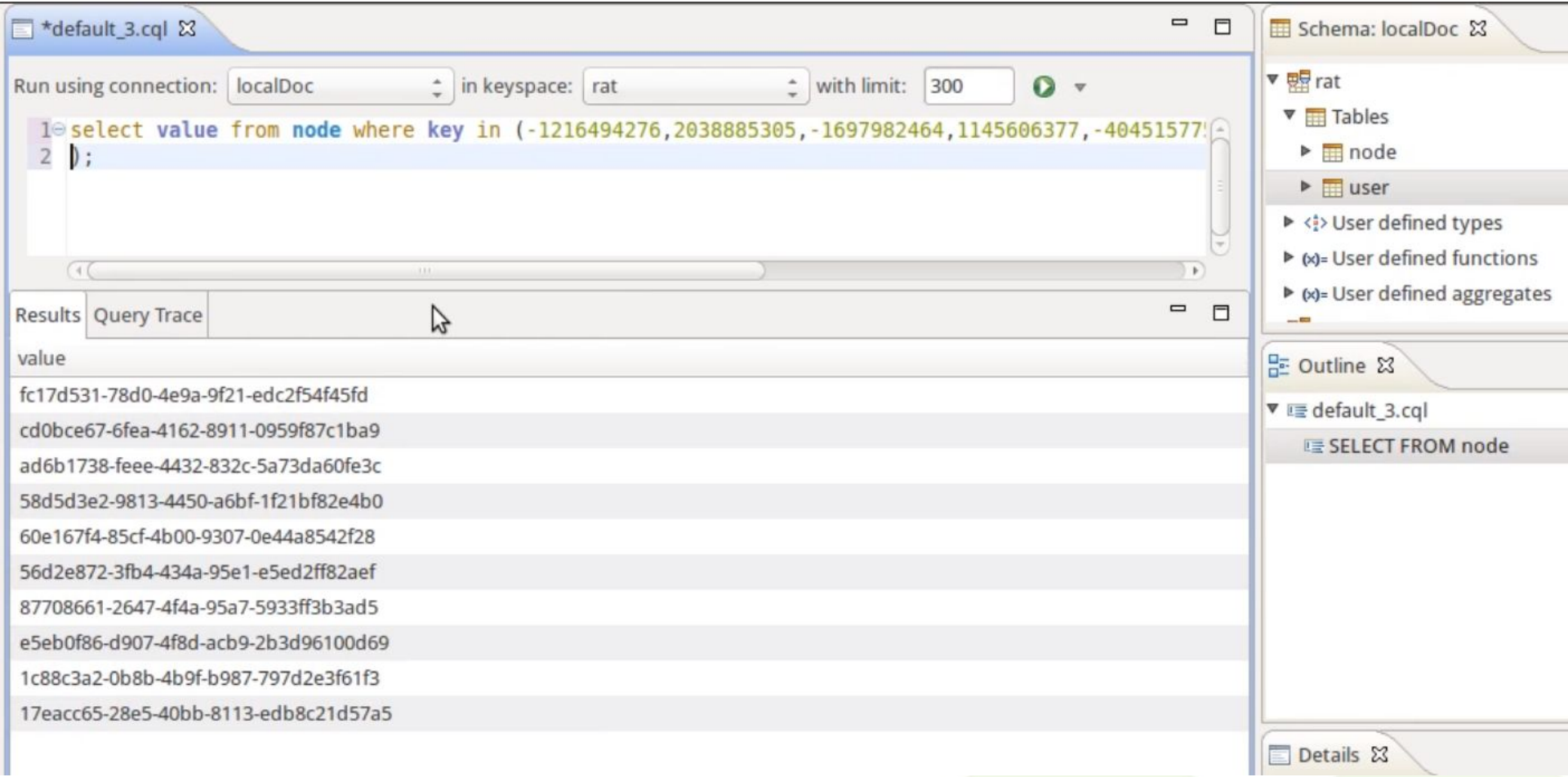
Value Members

▶ `val product: Int`

▶ `val rating: Double`

▶ `val user: Int`

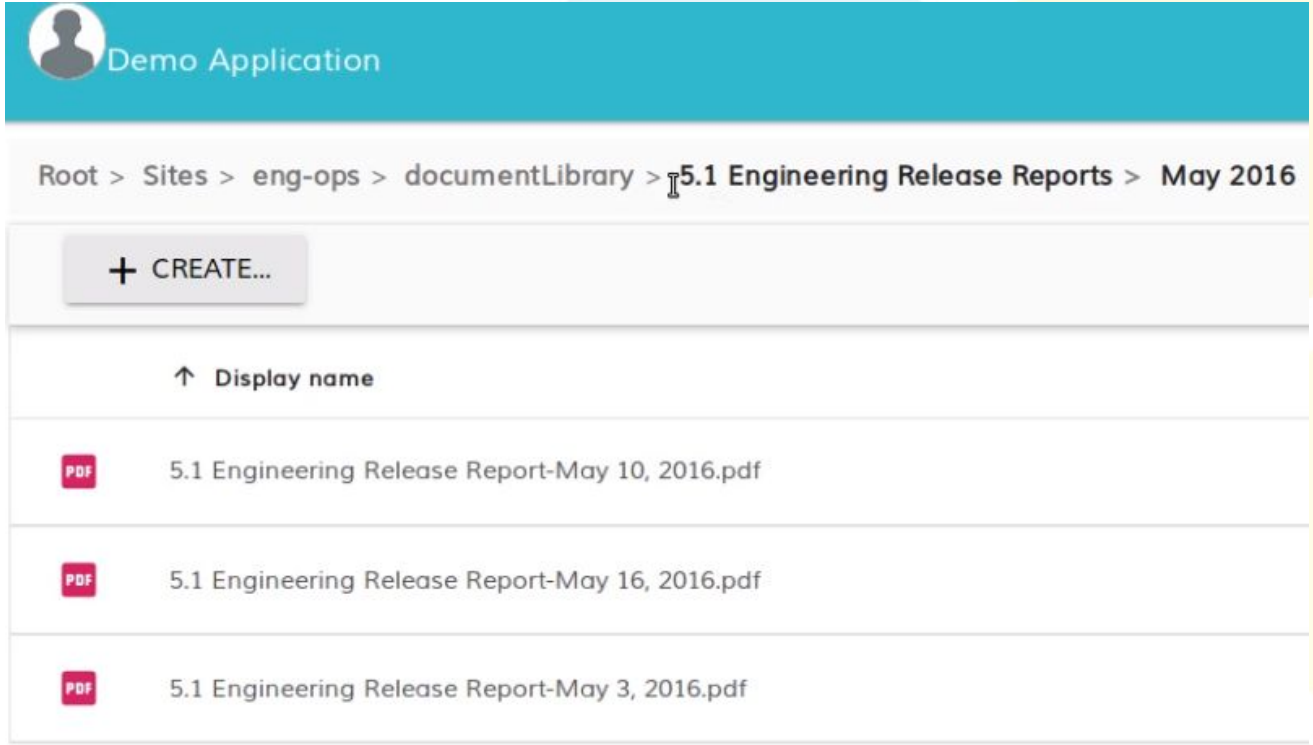

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AppTest (1)
attempt1 src main scala org alfresco hon App.scala
Project
org.alfresco.hon
App
test
target
attempt1.iml
hs_err_pid17151.log
pom.xml
listening (~/covudev/source/hc
src
main
resources
scala
org.alfresco.hon
App
KafkaSink.scala
KafkaSink
KafkaSink
test
target
listening.iml
pom.xml
squick (~/covudev/source/squi
dockerRunner
src
main
resources
scala
hon/App.scala
attempt1
quick/App.scala
hon/App.scala
listening
home/.../junit.scala
}
def hashPrint(aString: String): String = {
aString+ " is " + hashIt(aString)
}
def rate(act_type: String): Int = act_type match {
case "1" => 2 //sync.from.cloud
case "2" => 3 //file-previewed
case "5" => 7 //file-downloaded
case "7" => 2 // file-updated
case "10" => 9 //file-liked
case "11" => 0 //file-added
case "12" => 2 //sync.to.cloud
case "13" => 0 //folder-added
case "14" => 0 //file-deleted
case "15" => 0 //file-created
case "16" => 0 //user-joined
case "17" => 8 //comment-created
case "18" => 0 //files-added
case "19" => 0 //site.create
case "20" => 0 //user.create
case "21" => 0 //user-role-changed
case "22" => 7 //content.download
case "23" => 0 //subscriptions.followed
case "24" => 0 //files-deleted
case "25" => 0 //profile.status-changed
case "26" => 0 //link-deleted
case "27" => 0 //post-created
case _ => 0
}
/** Compute RMSE (Root Mean Squared Error). */


*default_3.cql
Run using connection: localDoc in keyspace: rat with limit: 300
1 select value from node where key in (-1216494276,2038885305,-1697982464,1145606377,-40451577
2);
Results
Query Trace
value
fc17d531-78d0-4e9a-9f21-edc2f54f45fd
cd0bce67-6fea-4162-8911-0959f87c1ba9
ad6b1738-feee-4432-832c-5a73da60fe3c
58d5d3e2-9813-4450-a6bf-1f21bf82e4b0
60e167f4-85cf-4b00-9307-0e44a8542f28
56d2e872-3fb4-434a-95e1-e5ed2ff82aef
87708661-2647-4f4a-95a7-5933ff3b3ad5
e5eb0f86-d907-4f8d-acb9-2b3d96100d69
1c88c3a2-0b8b-4b9f-b987-797d2e3f61f3
17eacc65-28e5-40bb-8113-edb8c21d57a5
Schema: localDoc
rat
Tables
node
user
User defined types
User defined functions
User defined aggregates
Outline
default_3.cql
SELECT FROM node
Details

Metadata Series

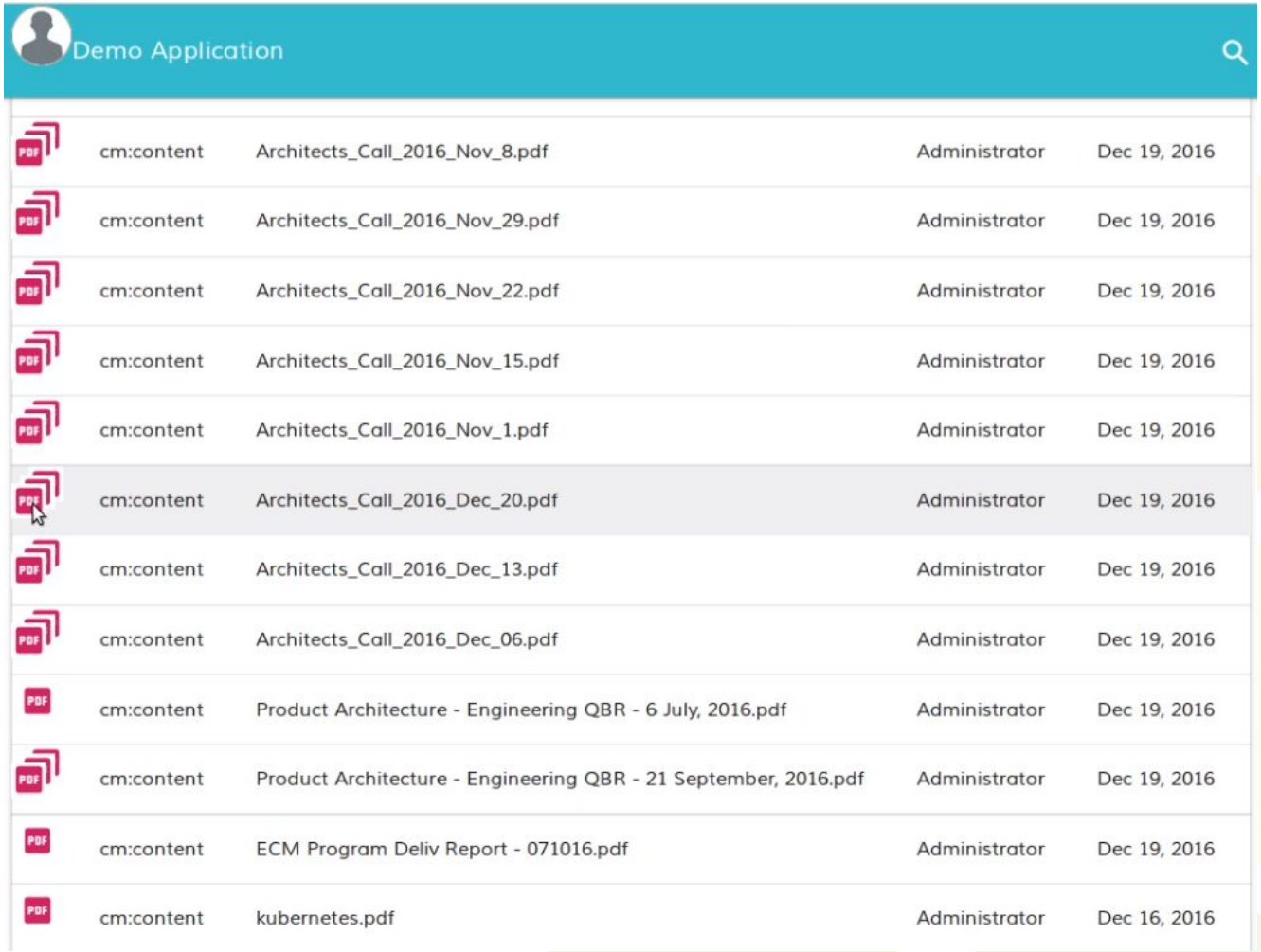
Determine related content based on only the metadata.

Eg. date series



Group by Series

Stored in Solr



Demo Application

architect

Home

Docu

Data Series

Architects_Call_2016_Dec_20.pdf

VIEW

Architects_Call_2016_Dec_13.pdf

VIEW

Architects_Call_2016_Dec_06.pdf

VIEW

Architects_Call_2016_Nov_29.pdf

VIEW

Architects_Call_2016_Nov_22.pdf

VIEW

Architects_Call_2016_Nov_8.pdf

VIEW

Architects_Call_2016_Nov_1.pdf

VIEW

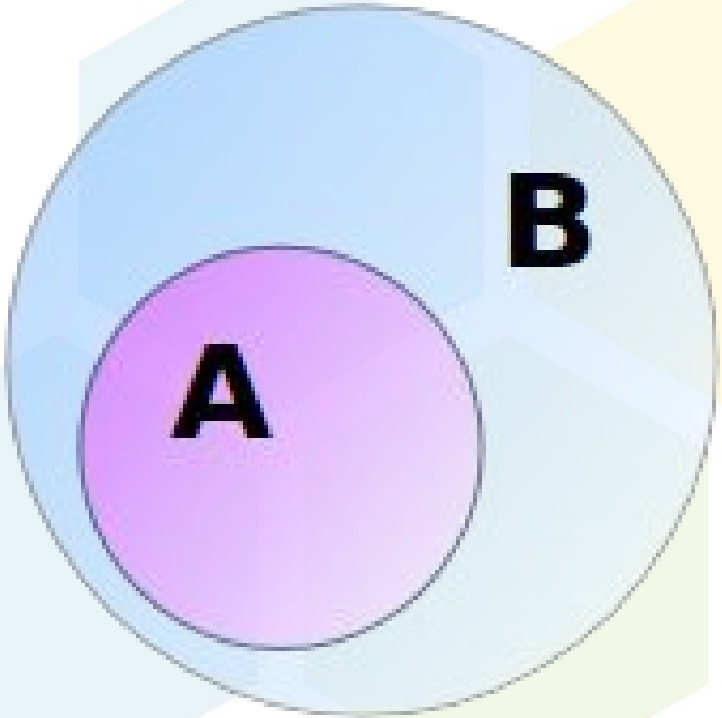
4. Anomaly Detection

Significant Terms with Solr

The significantTerms expression queries a collection but instead of returning the matching documents, it returns the significant terms in the matching documents.

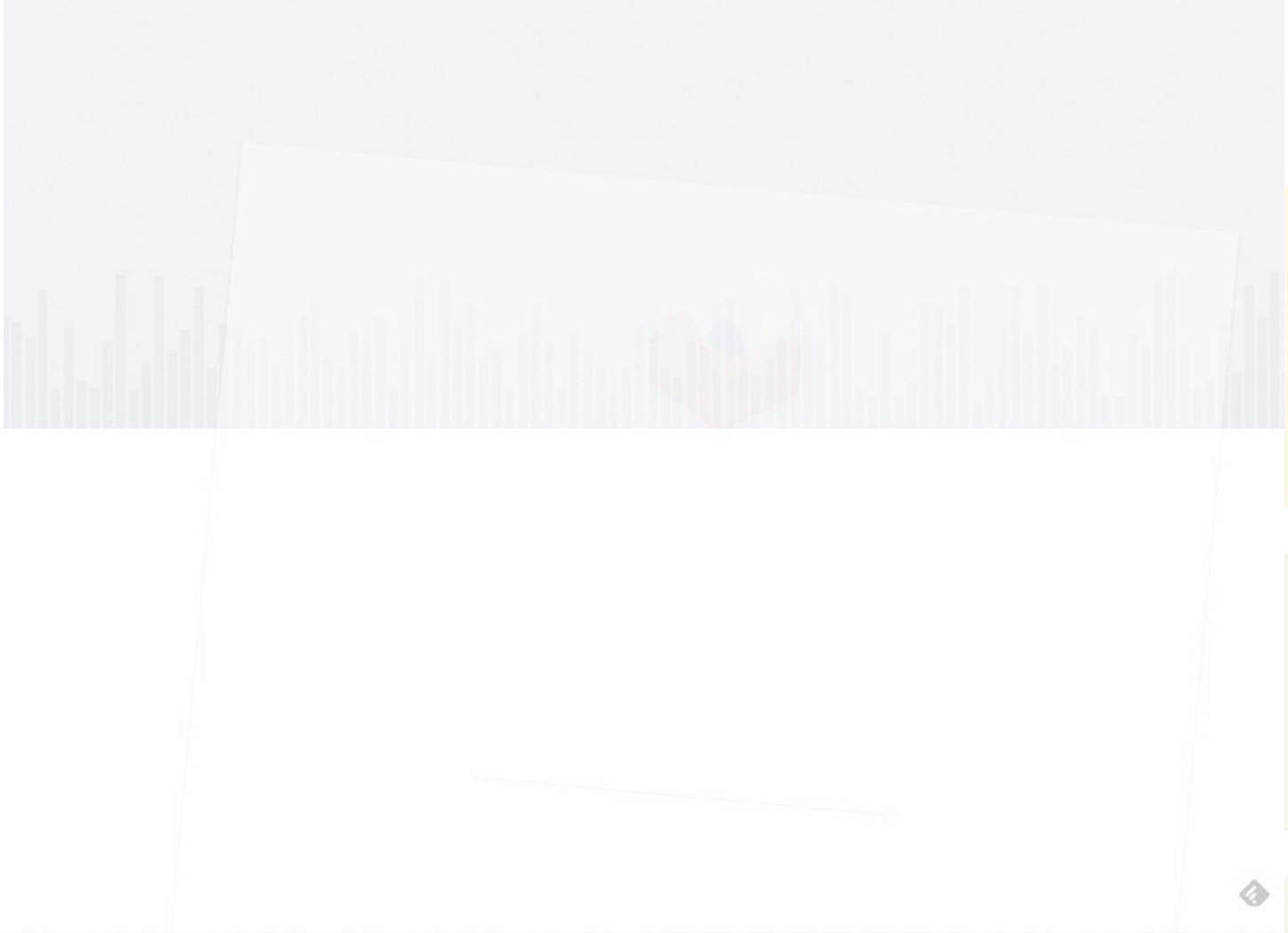
The **foreground** set is the result of a search. The **background** set is all the documents in the index.

It assigns higher scores to terms that are more frequent in the foreground set and rarer in the background set, in relation to other terms.



5. Querying by sql

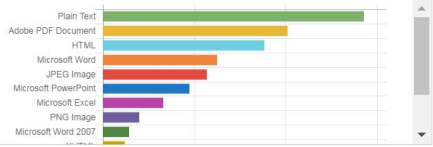
6. Charts



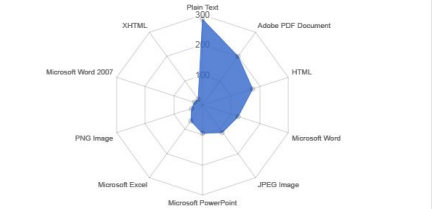
Reporting

Public

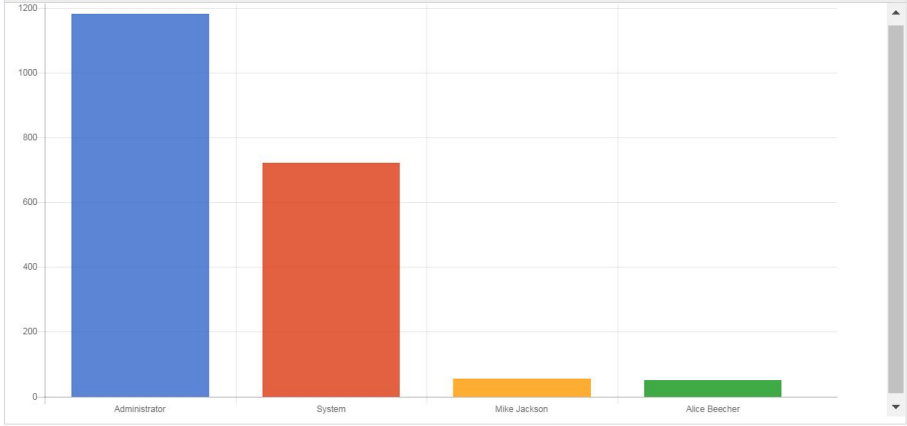
Mimetype



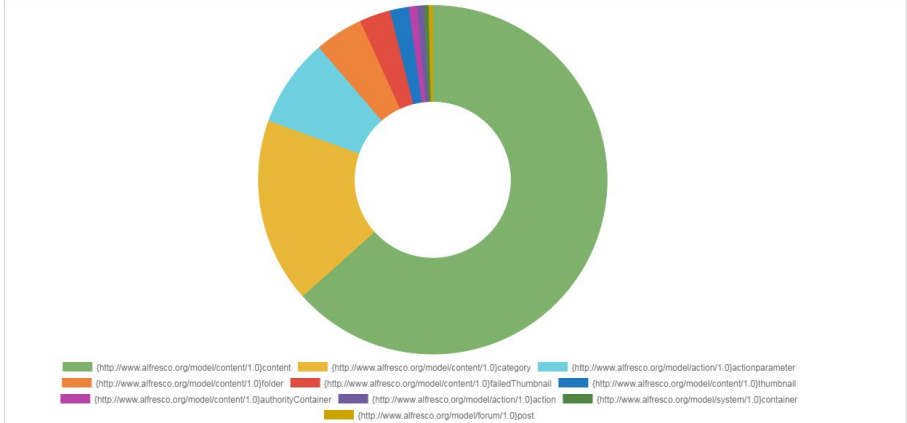
Mimetype Radar



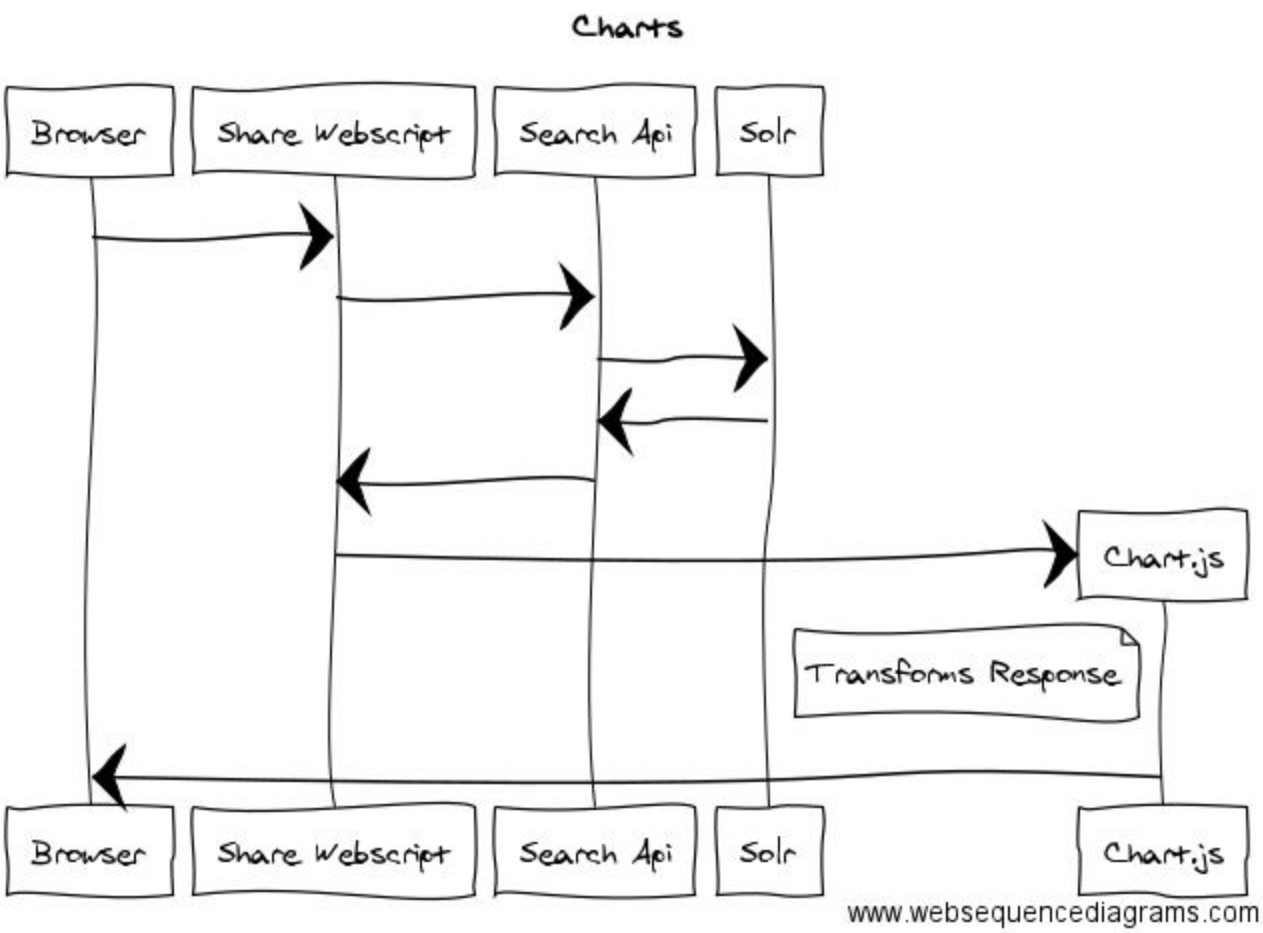
Top Content Creators

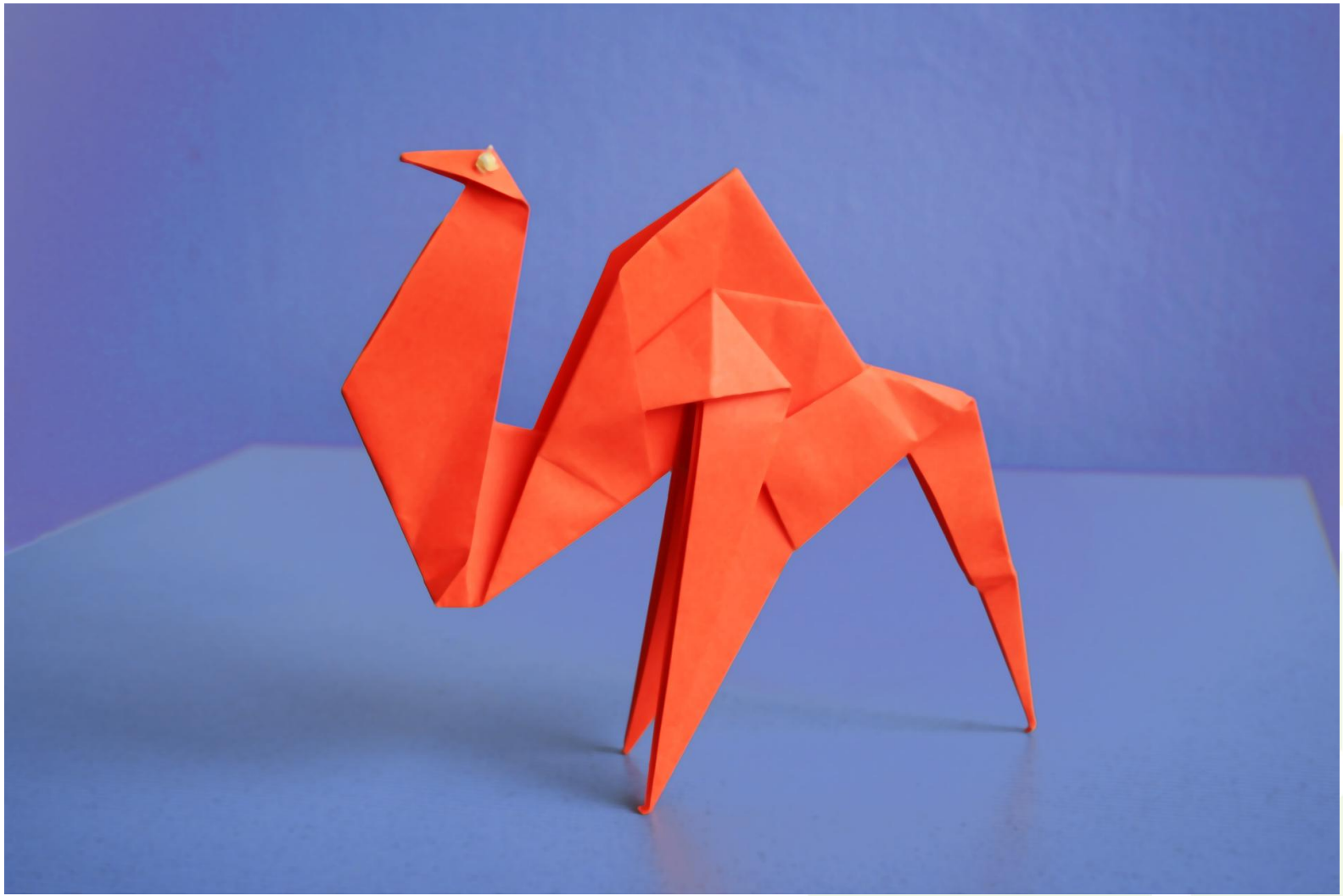


Content and Folder Types

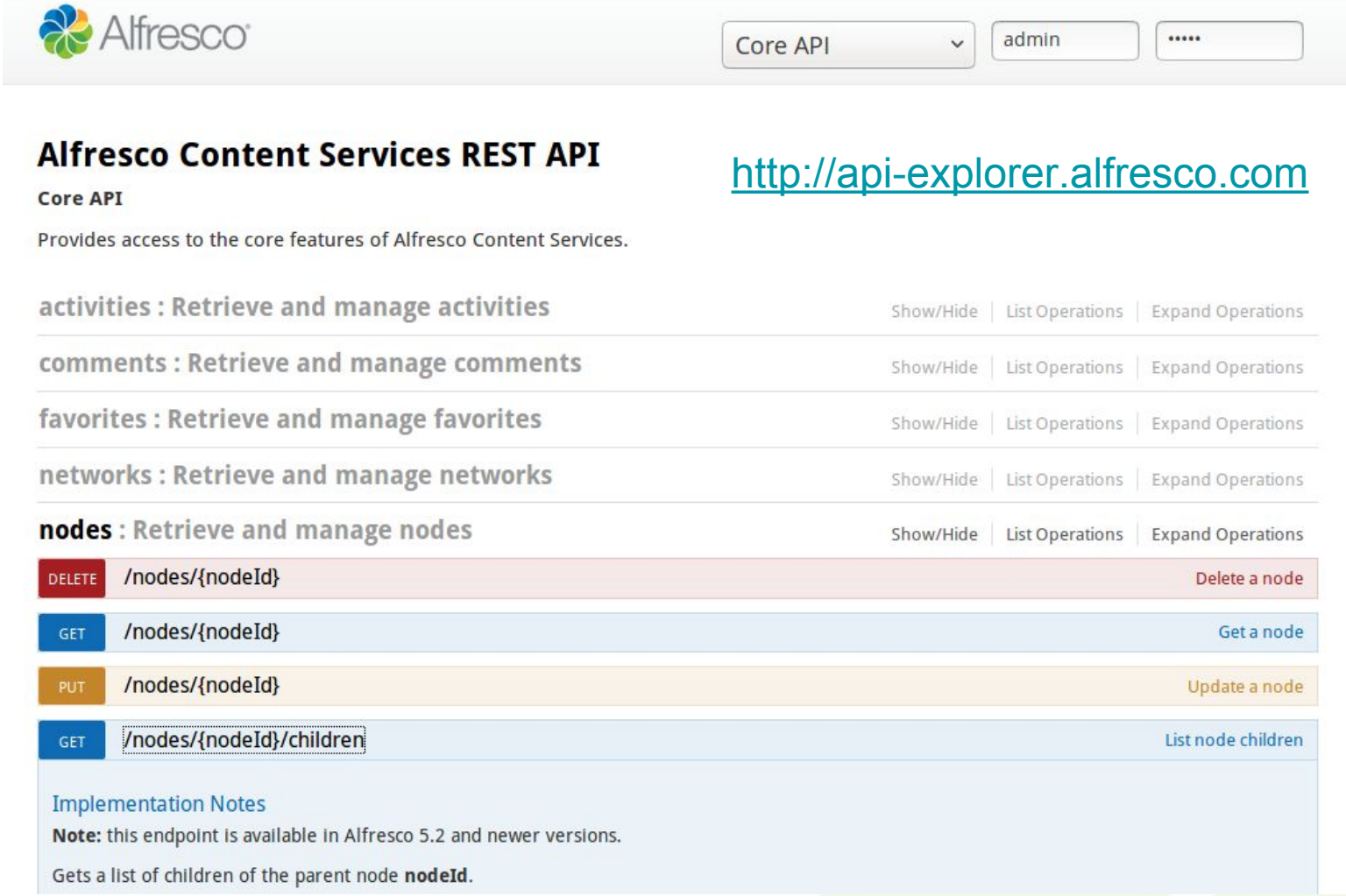


Alfresco
Licensed To: Enterprise License Holder
Alfresco Software, Inc. © 2005-2017 All rights reserved.


Charts
Browser
Share Webscript
Search Api
Solr
Chart.js
Transforms Response
Browser
Share Webscript
Search Api
Solr
Chart.js
www.websequencediagrams.com



7. Custom Restful Api


Alfresco
Core API
admin
Alfresco Content Services REST API
Core API
Provides access to the core features of Alfresco Content Services.
http://api-explorer.alfresco.com
activities : Retrieve and manage activities
Show/Hide | List Operations | Expand Operations
comments : Retrieve and manage comments
Show/Hide | List Operations | Expand Operations
favorites : Retrieve and manage favorites
Show/Hide | List Operations | Expand Operations
networks : Retrieve and manage networks
Show/Hide | List Operations | Expand Operations
nodes : Retrieve and manage nodes
Show/Hide | List Operations | Expand Operations
DELETE /nodes/{nodeId} Delete a node
GET /nodes/{nodeId} Get a node
PUT /nodes/{nodeId} Update a node
GET /nodes/{nodeId}/children List node children
Implementation Notes
Note: this endpoint is available in Alfresco 5.2 and newer versions.
Gets a list of children of the parent node nodeId.

Basic example of developing a Rest API on the Alfresco Platform that conforms to the api standards.

Edit

Add topics

9 commits | 1 branch | 0 releases | 1 contributor | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

covolution Updated the README — Latest commit c8800b5 3 minutes ago

src	Added category DELETE endpoint	8 minutes ago
.gitignore	Initial generation and added the .gitignore	8 minutes ago
LICENSE	Initial commit	14 minutes ago
README.md	Updated the README	3 minutes ago
debug.bat	Initial generation and added the .gitignore	8 minutes ago
debug.sh	Initial generation and added the .gitignore	8 minutes ago
pom.xml	Switched to Java 8 and added remote-api as a dependency	8 minutes ago
run.bat	Initial generation and added the .gitignore	8 minutes ago
run.sh	Initial generation and added the .gitignore	8 minutes ago

README.md

https://github.com/covolution/quick-api

Alfresco Quick Rest API Example

This is a very basic example of how to start developing a Rest API on the Alfresco Platform that conforms to the api standards.

It uses the Alfresco SDK.

8. Smart TV

9. Sharding by regex

Property based sharding (regex)

This sharding method uses any d:date, d:datetime, or d:text property

http://docs.alfresco.com/5.2/concepts/solr6-shard-approaches.html

```
prop.put("shard.method", ShardMethodEnum.PROPERTY.toString());
prop.put("shard.regex", "^[A-Za-z0-9._%+-]+@([A-Za-z0-9.-]+\\.[A-Za-z]{2,6})$");
prop.put("shard.key", ContentModel.PROP_EMAIL.toString());
```

10. Talking to Alfresco

Alexa Alfresco

https://github.com/melahn/alexa-alfresco

Alexa Skills Kit

Lambda

Open Alfresco:

Is Alfresco up?

List My Sites

List My Tasks

Approve Task {task}

Reject Task {task}

Read document

presentations@orderofthebee.org