Alfresco learning

keensoft

Day 2 - Integration

Angel Borroy

Founder member of Order of the Bee (http://orderofthebee.org))

- Programme Chair of BeeCon2016 (http://beecon.buzz)
- Participant in Cataloging Committee of addons

Speaker in international conferences

- Alfresco Summit 2013 (Barcelona)
- Alfresco Summit 2014 (London)
- BeeCon 2016 (Bruselas)
- Tech TalkLive #82 (https://www.youtube.com/watch?v=fy-dE9uOL-Y)
- Tech TalkLive #85 (https://www.youtube.com/watch?v=qz2LoVCU8Go)

Author in diferents contributions and addons to the platform

- GitHub (http://github.com/keensoft)

Angel Borroy

angel.borroy@keensoft.es

@AngelBorroy

GitHub

- http://github.com/keensoft
- http://github.com/angelborroy

Official community of Alfresco

- http://community.alfresco.com

Blogs

- http://orderofthebee.org
- http://angelborroy.wordpress.com
- http://www.keensoft.es/blog

Day 2 - Integration

CMIS

- Apache Workbench
- curl
- Cliente Java
- Others CMIS clients: PHP, Python, .NET, Objective-C, JavaScript

REST API

- Alfresco Web Scripts
- OpenAPIs / swagger.io

Aikau

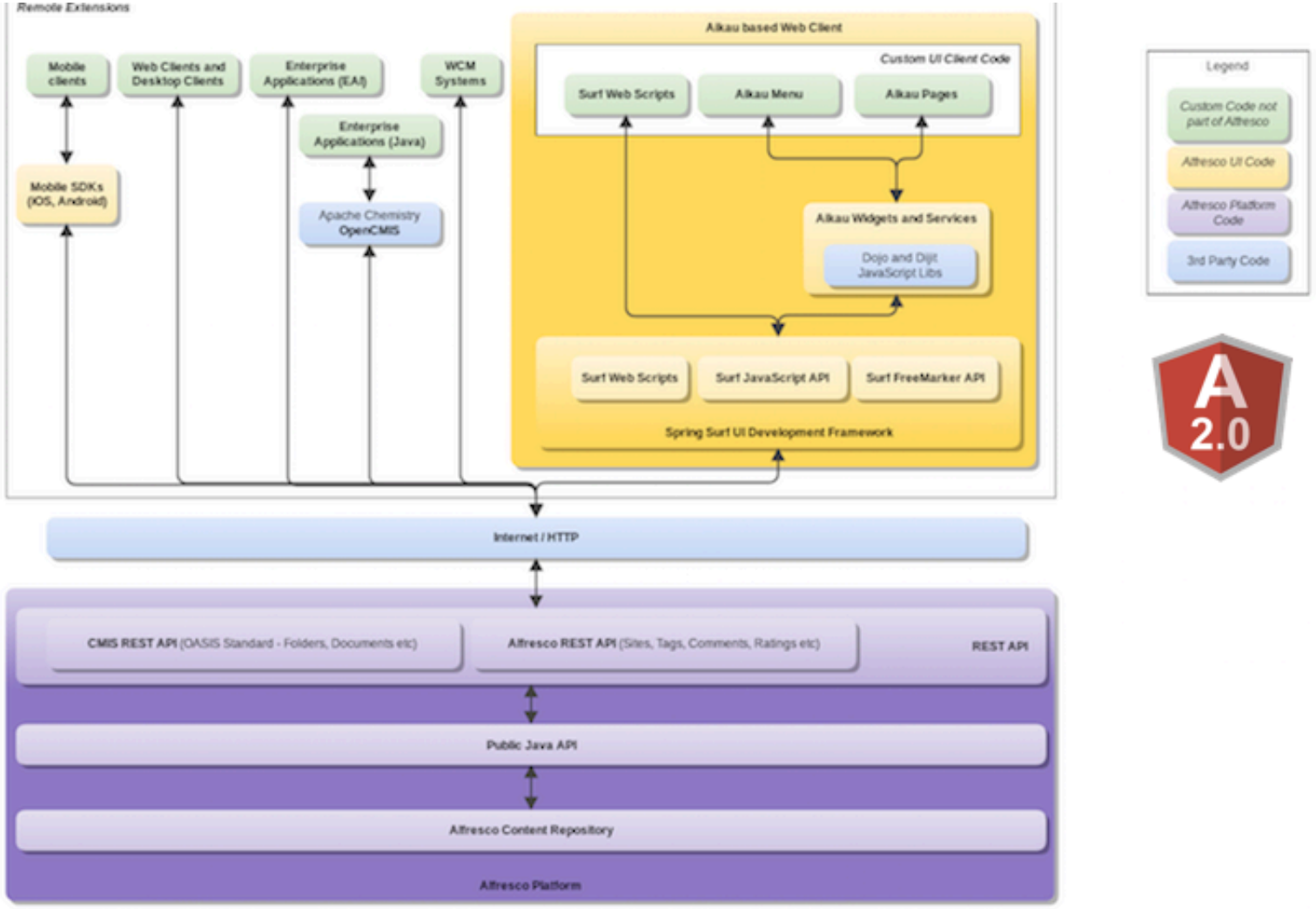
- Client Application

Alfresco Development Framework (ADF)

- Alfresco JavaScript Library for Node.js

Integration mechanism

Integration mechanism



CMIS

CMIS

- Content Management Interoperability Services* OASIS standard that allows the access to repository contents
- ECM servers: Alfresco, ECM Documentum, HP Interwoven, IBM Content Manager y FileNet, Lotus Quickr, Microsoft SharePoint, OpenText, SAP
- WCM servers: Magnolia, Liferay, Drupal, Hippo, TYPE3, OpenCMS, docCMS
- Blogs: Wordpress
- Clients: LibreOffice, Adobe Drive, Atlassian Confluence, SAP ECM Integration, Pentaho Data Integration, SugarCRM, Trac, Kofax,...
- SOA: Mule ESB, Spring Integration
- Libraries: Apache Chemistry (Java, Python, PHP, .NET, Objective-C)

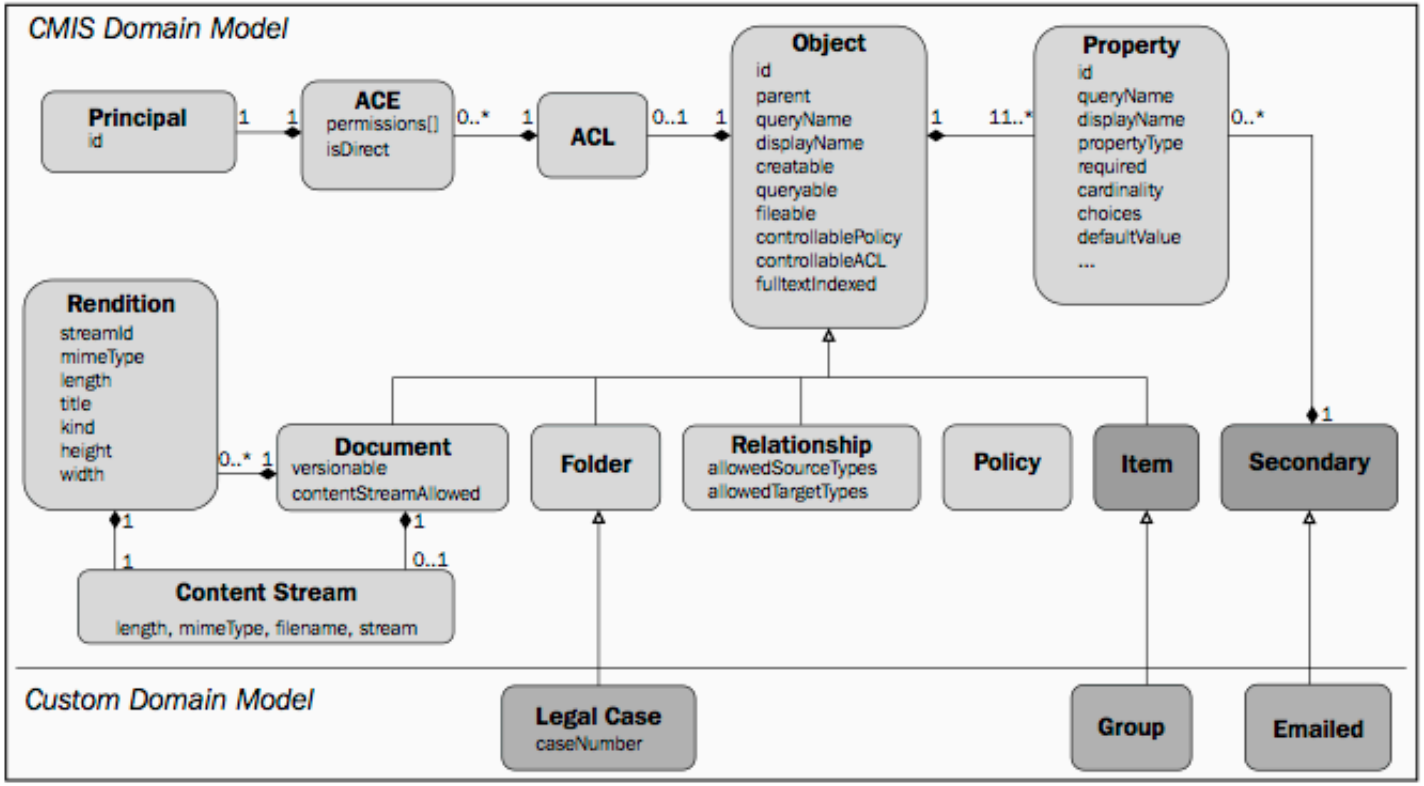
CMIS

Why use CMIS

- Technological neutrality: any language with support to HTTP requests and management of XML or JSON can be used to access the CMIS repository
- Platform independent: there is no requirement for the implementation of document repository. Alfresco is implemented in Java but Documentum is implemented in C
- Standard API for all platforms
- Search language based on SQL syntax
- Direct integration with BPM systems

CMIS

Model domain



CMIS

Model domain

Document

- Object in the repository, for example a PDF

Folder

- Object that have other folders or documents, for each content in the forlder is created automatically a relation son father

Relationship

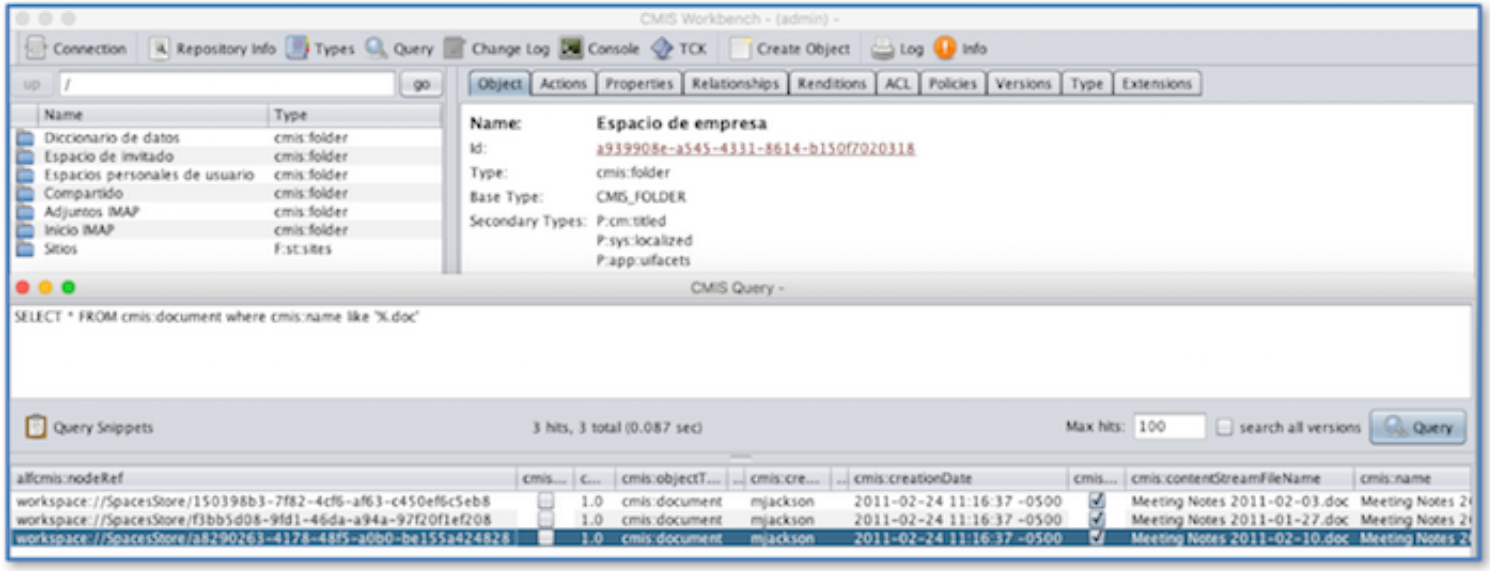
- Represents the relation between two objects origin and destiny. Optional: no implemented in Alfresco

Policy

- Policies in the management of objects, for example retention policy

Item

CMIS Workbench



http://chemistry.apache.org/java/developing/tools/dev-tools-workbench.html

CMIS

Services

- Repository: Information of repository and dictionary
- Navegation: Obtain children or hierarchies of children
- Objects: CRUD operations
- Discovery: Search for objects *wanted*
- Versioning: lock of objects and recover of versions
- Relations: relations of objects
- Policies: definiton and application of policies
- ACL: objects permissions

CMIS

Language SQL

```
-- All fields from documents
SELECT * FROM cmis:document

-- Some field from documents
SELECT cmis:name, cmis:description FROM cmis:document

-- Get documents having name 'like'
SELECT cmis:name FROM cmis:document WHERE cmis:name LIKE '%contract%'

-- Get documents by using FTS filter
SELECT * FROM cmis:document WHERE CONTAINS('alfresco')

-- Get documents in folder
SELECT * FROM cmis:document WHERE IN_FOLDER('folder id')

-- Get documents in hierarchy
SELECT * FROM cmis:document WHERE IN_TREE('folder id')

-- Get properties from different items
SELECT * FROM cmis:document d JOIN cm:titled t
          ON d.cmis:objectId = t.cmis:objectId
```

CMIS

Access protocols

Web Services SOAP 1.0

The requests are done using the SOAP protocol

https://alfresco.keensoft.es/alfresco/cmisws/cmis?wsdl

RESTful AtomPub binding 1.1

The requests are done using Atom XML feed or Atom XML Entry

https://alfresco.keensoft.es/alfresco/api/-default-/public/cmis/versions/1.1/atom

RESTful Browser binding 1.1

The requests are done using JSON (instead of Atom XML feed or Atom XML Entry)

https://alfresco.keensoft.es/alfresco/api/-default-/public/cmis/versions/1.1/browser

CMIS

curl

```
# ATOM - Get repository information
$ curl -u admin:keensoft https://alfresco.keensoft.es/alfresco/cmisatom

# ATOM - Get repository ID
$ curl -u admin:keensoft https://alfresco.keensoft.es/alfresco/cmisatom \
| grep -o "<cmis:repositoryId.*repositoryId"

<cmis:repositoryId>10b21ede-f9b8-4f59-a34f-9a46f854c95a</cmis:repositoryId

# BROWSER - Get repository information
$ curl -u admin:keensoft https://alfresco.keensoft.es/alfresco/cmisbrowser

# BROWSER - Get repository ID
$ curl -u admin:keensoft https://alfresco.keensoft.es/alfresco/cmisbrowser\
|jq '.[] | .repositoryId'

"10b21ede-f9b8-4f59-a34f-9a46f854c95a"
```

CMIS

<) Install Alfresco for development

https://community.alfresco.com/docs/DOC-6296-community-file-list-201605-ga

Windows

```
alfresco-community-installer-201605-win-x64.exe
```

Mac

```
alfresco-community-installer-201605-osx-x64.dmg
```

Linux

```
alfresco-community-installer-201605-linux-x64.bin
```

CMIS

CMIS Java Client

Create a Maven project

```
$ mvn archetype:generate -DgroupId=es.keensoft.cmis -DartifactId=cmis-app \
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Add the dependecy of CMIS client in 'pom.xml'

```
<dependency>
  <groupId>org.apache.chemistry.opencmis</groupId>
  <artifactId>chemistry-opencmis-client-impl
  </artifactId>
  <version>0.14.0</version>
</dependency>
```

CMIS

CMIS Java Client

Create a session and recover the information from the *repository*

```
public class App {
  public static void main( String[] args ) {

    SessionFactory sessionFactory = SessionFactoryImpl.newInstance();
    Map<String, String> parameters = new HashMap<String, String>();
    parameters.put(SessionParameter.USER, "admin");
    parameters.put(SessionParameter.PASSWORD, "admin");
    parameters.put(SessionParameter.ATOMPUB_URL,
        "http://localhost:8080/alfresco/api/" +
            "-default-/public/cmis/versions/1.1/atom");
    parameters.put(SessionParameter.BINDING_TYPE, BindingType.ATOMPUB.value());
    parameters.put(SessionParameter.COMPRESSION, "true");
    parameters.put(SessionParameter.CACHE_TTL_OBJECTS, "0");

    Repository repository = sessionFactory.getRepositories(parameters).get(0);
    Session session = repository.createSession();

    System.out.println(session.getRepositoryInfo());

  }
}
```

CMIS

Use the service to navigate

```
Folder rootFolder = session.getRootFolder();
    for (CmisObject object : rootFolder.getChildren()) {

        if (object.getBaseTypeId() == BaseTypeId.CMIS_FOLDER) {
            Folder folder = (Folder) object;
            System.out.println("Folder: " + folder.getPath());

        } else if (object.getBaseTypeId() == BaseTypeId.CMIS_DOCUMENT) {
            Document document = (Document) object;
            System.out.println("Document: " + document.getName());

        } else {
            System.out.println(object.getBaseType().getDisplayName() + ": " +
                object.getName());
        }
    }
}
```

CMIS

Use the service of *objects*

```
Folder parent = (Folder) session.getObjectByPath("/Sitios/swsdp/documentLibrary");

Map<String, Object> properties = new HashMap<String, Object>();
properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
properties.put(PropertyIds.NAME, "myNewDocument.txt");

properties.put("cm:description", "My description");
properties.put("cm:title", "My title");

byte[] content = "Hello World!".getBytes();
InputStream stream = new ByteArrayInputStream(content);
ContentStream contentStream =
    new ContentStreamImpl(name, BigInteger.valueOf(content.length),
        "text/plain", stream);

Document doc =
    parent.createDocument(properties, contentStream, VersioningState.MAJOR);
```

CMIS

Use the service of *objects*

```
Map<String, Object> properties = new HashMap<String, Object>();
properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
properties.put(PropertyIds.NAME, "myNewDocument.txt");

List<Object> aspects = new ArrayList<Object>();
aspects.add("P:cm:titled");
properties.put(PropertyIds.SECONDARY_OBJECT_TYPE_IDS, aspects);

properties.put("cm:description", "My description");
properties.put("cm:title", "My title");

byte[] content = "Hello World!".getBytes();
InputStream stream = new ByteArrayInputStream(content);
ContentStream contentStream =
    new ContentStreamImpl(name, BigInteger.valueOf(content.length),
        "text/plain", stream);

Document doc =
    parent.createDocument(properties, contentStream, VersioningState.MAJOR);
```

CMIS

Use the service for *discover*

```
// false = only last version
ItemIterable<QueryResult> results =
   session.query("SELECT * FROM cmis:document WHERE cmis:name LIKE '%New%'",
       false);

for (QueryResult result : results) {

    String objectId =
        result.getPropertyById("cmis:objectId").getFirstValue().toString();

    Document docResult =
        (Document) session.getObject(session.createObjectId(objectId));

    System.out.println(docResult.getName());

}
```

CMIS

Use the service to *version*

```
Document docResult =
    (Document) session.getObjectByPath(
    "/Sitios/swsdp/documentLibrary/myNewDocument.txt");

ObjectId pwcId = docResult.checkOut();
Document pwc = (Document) session.getObject(pwcId);

Map<String, Object> properties = new HashMap<String, Object>();
properties.put("cm:title", "My title (version)");

byte[] content = "Hello World (versioned)!".getBytes();
InputStream stream = new ByteArrayInputStream(content);
ContentStream contentStream =
    new ContentStreamImpl(name, BigInteger.valueOf(content.length),
        "text/plain", stream);

// true = Major version
pwc.checkIn(true, properties, contentStream, "Versioned!");

versions = docResult.getAllVersions();
for (Document version : versions) {
    System.out.println("Version: " + version.getVersionLabel());
}
```

CMIS

Others CMIS clients

PHP

https://chemistry.apache.org/php/phpclient.html
https://github.com/keensoft/Basic-CMIS

Python

https://chemistry.apache.org/python/cmislib.html

.NET

https://chemistry.apache.org/dotnet/portcmis.html

Objective-C

https://chemistry.apache.org/objective-c/objectivecmis.html

JavaScript

https://chemistry.apache.org/javascript/parts.html (jQuery)

CMIS

<) Revision: List all categories of the branch Regions

Root node can be obtained by using FTS PATH

```
ItemIterable<QueryResult> regionRoot =
    session.query(
        "SELECT * FROM cm:category C WHERE " +
        "CONTAINS(C, 'PATH:\"/cm:generalclassifiable/cm:Regiones\"')", false);
```

Using the method `IN_TREE` for get all the nodes inside of category 'Regions'

```
ItemIterable<QueryResult> regions =
    session.query("...", false);
```

Print the result list

```
for (QueryResult region : regions) {
    Object nodeRef = region.getPropertyValueById("alfcmis:nodeRef");
    String nameCategory = region.getPropertyValueById("cm:name").toString();
    System.out.println("NodeRef: " + nodeRef + ", name: " + nameCategory);
}
```

REST API

REST API

Alfresco 5.1 or before

Native API developed using Web Script technology

https://alfresco.keensoft.es/alfresco/s/index/uri/

Only the ones that start with **/api** are considered from the public API

Alfresco 5.2 o after

Based on OpenAPIs standard and defined in YAML

https://alfresco.keensoft.es/api-explorer-1.2/

https://github.com/Alfresco/rest-api-explorer/tree/master/src/main/webapp/definitions

REST API

Native API

Sites

GET https://alfresco.keensoft.es/alfresco/s/api/sites

GET https://alfresco.keensoft.es/alfresco/s/api/sites/addons

GET | POST | PUT
https://alfresco.keensoft.es/alfresco/s/api/sites/addons/memberships

GET | POST | PUT | DELETE
https://alfresco.keensoft.es/alfresco/s/api/sites/addons/memberships/admin

GET https://alfresco.keensoft.es/alfresco/s/api/sites/addons/roles

GET | POST https://alfresco.keensoft.es/alfresco/s/api/sites/addons/invitations

GET https://alfresco.keensoft.es/alfresco/s/api/sites/addons/export

REST API

Native API

People

GET | POST https://alfresco.keensoft.es/alfresco/s/api/people

GET https://alfresco.keensoft.es/alfresco/s/api/people/admin

GET https://alfresco.keensoft.es/alfresco/s/api/people/admin/sites

GET | POST | DELETE
https://alfresco.keensoft.es/alfresco/s/api/people/admin/preferences

POST https://alfresco.keensoft.es/alfresco/s/api/person/changepassword/admin

REST API

Native API

Group

GET https://alfresco.keensoft.es/alfresco/s/api/groups

GET | PUT | DELETE
https://alfresco.keensoft.es/alfresco/s/api/groups/ALFRESCO_ADMINISTRATORS

GET
https://alfresco.keensoft.es/alfresco/s/api/groups/ALFRESCO_ADMINISTRATORS/children

GET
https://alfresco.keensoft.es/alfresco/s/api/groups/ALFRESCO_ADMINISTRATORS/children/a

GET
https://alfresco.keensoft.es/alfresco/s/api/groups/ALFRESCO_ADMINISTRATORS/parents

REST API

Native API

Tags

GET | POST
https://alfresco.keensoft.es/alfresco/s/api/tags/workspace/SpacesStore

DELETE
https://alfresco.keensoft.es/alfresco/s/api/tags/workspace/SpacesStore/ejemplos

GET
https://alfresco.keensoft.es/alfresco/s/api/tags/workspace/SpacesStore/ejemplos/nodes

REST API

Native API **Nodes**

GET
https://alfresco.keensoft.es/alfresco/s/api/node/workspace/SpacesStore/7e49b7c5-067b-4cfb-8eec-ba46d7908ba7/metadata

POST
https://alfresco.keensoft.es/alfresco/s/api/metadata/node/workspace/SpacesStore/7e49b7
067b-4cfb-8eec-ba46d7908ba7

GET | POST
https://alfresco.keensoft.es/alfresco/s/api/node/workspace/SpacesStore/7e49b7c5-067b-4cfb-8eec-ba46d7908ba7/comments

GET | POST
https://alfresco.keensoft.es/alfresco/s/api/node/workspace/SpacesStore/7e49b7c5-067b-4cfb-8eec-ba46d7908ba7/tags

GET
https://alfresco.keensoft.es/alfresco/s/api/node/workspace/SpacesStore/7e49b7c5-067b-4cfb-8eec-ba46d7908ba7/ruleset/rules

REST API

Native API

Content

GET
https://alfresco.keensoft.es/alfresco/s/api/node/content/workspace/SpacesStore/755647d1bc54-4c2f-9235-61e200653b24

POST https://alfresco.keensoft.es/alfresco/s/api/upload

```
$ curl -u admin:admin -v POST -F filedata=@test.txt -F siteid=swsdp \
-F containerid=documentLibrary -F uploaddirectory=/ \
http://localhost:8080/alfresco/service/api/upload

{
   "nodeRef": "workspace://SpacesStore/5dab255b-1187-4ea0-8e9a-19fa6772ef40",
   "fileName": "test.txt",
   "status":
   {
      "code": 200,
      "name": "OK",
      "description": "File uploaded successfully"
   }
}
```

REST API

Native API

Tasks

GET https://alfresco.keensoft.es/alfresco/s/api/workflow-definitions

GET https://alfresco.keensoft.es/alfresco/s/api/workflow-instances

GET https://alfresco.keensoft.es/alfresco/s/api/node/workspace/SpacesStore/7e49b7c5-067b-4cfb-8eec-ba46d7908ba7/workflow-instances

POST https://alfresco.keensoft.es/alfresco/s/api/workflow/task/end/4185

REST API

Native API

Cliente Java

Add the dependence of HTTP client to the file **pom.xml** using the previous example

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpcore</artifactId>
    <version>4.1</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpmime</artifactId>
    <version>4.1</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.1</version>
</dependency>
```

REST API

Native API

Java Java

Create a method `main` that gets a ticket from an Alfresco session (1)

```
// HTTP request

JSONObject au = new JSONObject();
au.put("username", "admin");
au.put("password", "admin");

HttpClient httpClient = new DefaultHttpClient();
HttpPost postRequest =
        new HttpPost("http://localhost:8080/alfresco/service/api/login");

StringEntity input = new StringEntity(au.toJSONString());
input.setContentType("application/json");
postRequest.setEntity(input);

HttpResponse response = httpClient.execute(postRequest);
```

Next >>

REST API

Native API

Java Client

Create a method `main` that gets the ticket of Alfresco session (2)

```
// HTTP response

JSONParser jsonParser = new JSONParser();
JSONObject ticket =
        (JSONObject) jsonParser.parse(
                IOUtils.readAllLines(response.getEntity().getContent()
                )
            );
JSONObject data = (JSONObject) ticket.get("data");
String alfrescoTicket = data.get("ticket").toString();
System.out.println(alfrescoTicket);

httpClient.getConnectionManager().shutdown();
```

REST API

Native API

Java Client

Invoke the service of sites list

```
httpClient = new DefaultHttpClient();
httpClient.getParams().setParameter("http.protocol.version",
    HttpVersion.HTTP_1_1);
httpClient.getParams().setParameter("http.protocol.content-charset", "UTF-8");

HttpGet getRequest =
        new HttpGet("http://localhost:8080/alfresco/s/api/sites?alf_ticket=" +
            alfrescoTicket);

response = httpClient.execute(getRequest);
JSONParser jsonParser = new JSONParser();
JSONArray body =
    (JSONArray) jsonParser.parse(IOUtils.readAllLines(
        response.getEntity().getContent()));

for (Object element : body) {
    JSONObject site = (JSONObject) element;
    System.out.println(site.get("shortName"));
}
```

REST API

<) Revision: Obtain the list of all `userName` in the system with the Java client

http://localhost:8080/alfresco/s/api/people


```
{
  "people" : [
    {
        "url": "\/alfresco\/s\/api\/people\/guest",
        "userName": "guest",
        "enabled": false,
        "firstName": "Guest",
        "lastName": ""
    }
    ,
    {
        "url": "\/alfresco\/s\/api\/people\/abeecher",
        "userName": "abeecher",
        "enabled": false,
        "firstName": "Alice",
        "lastName": "Beecher"
    }
  ]
}
```

REST API

OpenAPIs

Explore and test the API

https://alfresco.keensoft.es/api-explorer-1.2

Explore the definition of the services in YAML

https://github.com/Alfresco/rest-api-explorer/tree/master/src/main/webapp/definitions

Create a Java client for the core

https://raw.githubusercontent.com/Alfresco/rest-api-explorer/master/src/main/webapp/definitions/alfresco-core.yaml

http://editor.swagger.io/

```
Generate Client > Java
```

REST API

OpenAPIs

Get the list of sites

```
ApiClient apiClient = Configuration.getDefaultApiClient();
apiClient.setBasePath(
"https://alfresco.keensoft.es/alfresco/api/-default-/public/alfresco/versions/1");
apiClient.setUsername("admin");
apiClient.setPassword("keensoft");

SitesApi api = new SitesApi();
api.setApiClient(apiClient);
SitePaging sites = api.listSites(null, null, null, null, null, null);
for (SiteEntry site : sites.getList().getEntries()) {
    System.out.println(site.getEntry().getId());
}
```

REST API

<) Revision: Get the list of all `userName` in the system using Swagger cliente

Use the *handler* generated by Swagger `PeopleApi`

The URL REST that invokes is the follow:

https://alfresco.keensoft.es/alfresco/api/-default-/public/alfresco/versions/1/people

Aikau

Aikau

Aikau is a web development technology created by Alfresco to be used over the Alfresco Share and Alfresco Record Management.

This technology allows to create web applications over Alfresco in a quick way through a set of preconfigured components.

Components can be discovered at http://dev.alfresco.com/resource/docs/aikau-jsdoc/

Aikau

An Aikau web application can be created using Maven

```
$ mvn archetype:generate -DgroupId=es.keensoft.cmis -DartifactId=aikau-app \
-DarchetypeCatalog=https://artifacts.alfresco.com/nexus/content/groups/public/arch
-DarchetypeGroupId=org.alfresco -DarchetypeArtifactId=aikau-sample-archetype \
-DarchetypeVersion=RELEASE -DinteractiveMode=false
```

Once is configured, compile and start in the folder `aikau-app`

```
$ mvn install
$ mvn jetty:run
```

By default it is required the Alfresco repository be available at the port 8080

http://localhost:8080/alfresco

Once is started, is accessible using the following URL

http://localhost:8090/aikau-sample/

Aikau

`/src/main/webapp/WEB-INF/webscripts/pages/home.get.js`

The pages are builded declaring *widgets* based on the constructions components of Aikau...

```
// Add more widgets here !!!
,{
  name: "alfresco/documentlibrary/AlfDocumentList",
  config: {
    rootNode: "alfresco://user/home",
    rawData: true,
    widgets: [
      {
        name: "alfresco/documentlibrary/views/AlfSimpleView"
      }
    ]
  }
}
```

... and associating them to data services

```
// Add more services here !!!
,"alfresco/services/DocumentService"
```

Aikau

Detailed examples on using each component can be found at Aikau Sandpit

https://aikau-sandpit.alfresco.com/aikau-sandpit/page/na/ws/home

There are others tutorials available

https://github.com/Alfresco/Aikau/blob/master/tutorial/chapters

And a extensive collection of articles

https://community.alfresco.com/community/ecm/blog/tags#/?tags=aikau

The YUI technology of Alfresco Share is being replaced by Aikau pages

https://community.alfresco.com/community/ecm/blog/2016/11/23/create-and-edit-site-customization

Aikau

Through authentication mechanims like Kerberos SSO, the Aikau web applications can be integrated transparently in organizations

https://angelborroy.wordpress.com/2016/10/05/sso-support-for-aikau-apps/

https://github.com/angelborroy/aikau-kerberos-sso

ADF

ADF

Application Development Framework is the new platform for Alfresco development based in *Angular 2* and *Google Material*

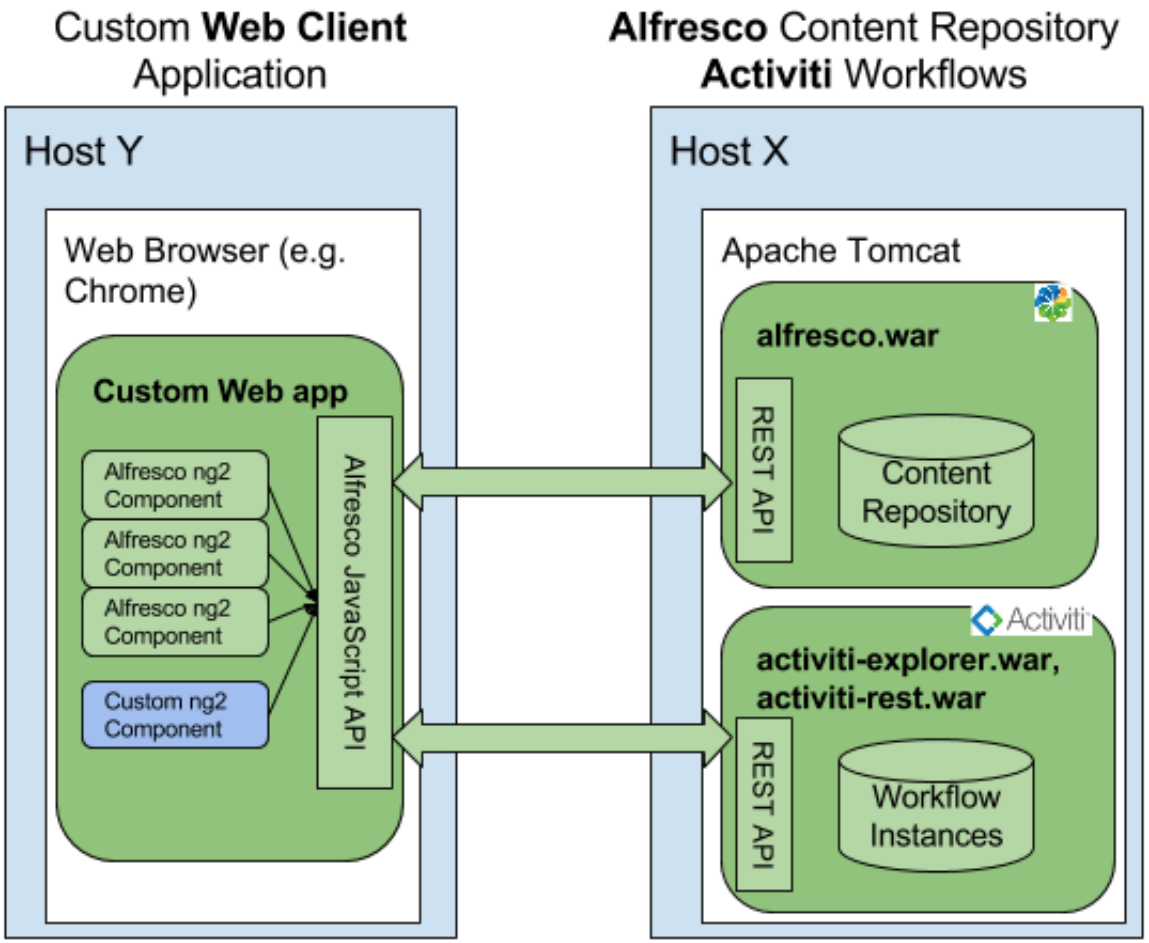
It will be available after Alfresco 5.2, since it works on the new API REST based in *OpenAPIs*

It allows the contruction of Angular 2 applications by using prefabricated web components:

- Core library
- DataTable
- DocumentList
- Viewer
- Login
- Upload

Components reference: http://devproducts.alfresco.com/

ADF


Custom **Web Client** Application
Alfresco Content Repository **Activiti** Workflows
Host Y
Host X
Web Browser (e.g. Chrome)
Apache Tomcat
Custom Web app
Alfresco ng2 Component
Alfresco ng2 Component
Alfresco ng2 Component
Custom ng2 Component
Alfresco JavaScript API
alfresco.war
REST API
Content Repository
Activiti
activiti-explorer.war, activiti-rest.war
REST API
Workflow Instances

ADF

Applications developed with ADF will be deployed on a different server than the one running Alfresco, since they are different technologies

For this reason, it is needed to activate CORS in the Alfresco Server

```
$ wget https://artifacts.alfresco.com/nexus/service/local/repositories/releases/
content/org/alfresco/enablecors/1.0/enablecors-1.0.jar
$ cp enablecors-1.0.jar /opt/alfresco/modules/platform
$ service alfresco restart
```

It requires a `node` version after 5.12

```
$ node -v
v6.2.2
```

For the application construction Yeoman generator is used, that needs to be installed in the development machine

```
$ npm install -g yo
```

ADF

Alfresco provides a application generator based on Yeoman

```
$ npm install -g generator-ng2-alfresco-app
```

Once installed, can be built the test application.

```
$ yo ng2-alfresco-app
? What's the name of your App? ng2-test
Your generator must be inside a folder named ng2-test
I'll automatically create this folder.
? How would you describe the app? Alfresco Angular 2 Application Example
? Author's Name Angel Borroy
? Author's Email angel.borroy@keensoft.es
? Author's Homepage
? Package keywords (comma to split)
? What is your Alfresco platform server URL? http://localhost:8080
? What is your Activiti platform server URL? http://127.0.0.1:9999
? GitHub username or organization
? Do you want include the User info component? Yes
? Do you want include a drawer bar? Yes
? Do you want include a search bar? Yes
? Do you want include a Document List? Yes
? Do you want include a Tasks List? Yes
? Which license do you want to use? Apache 2.0
```

ADF

Once generated, the application can be launched using *Node.js*

```
$ cd ng2-test
$ npm update
$ npm start
```

http://localhost:3000

For further reference

https://community.alfresco.com/docs/DOC-4595-getting-started-with-alfresco-application-development-framework

Alfresco learning

keensoft

Day 2 - Integration