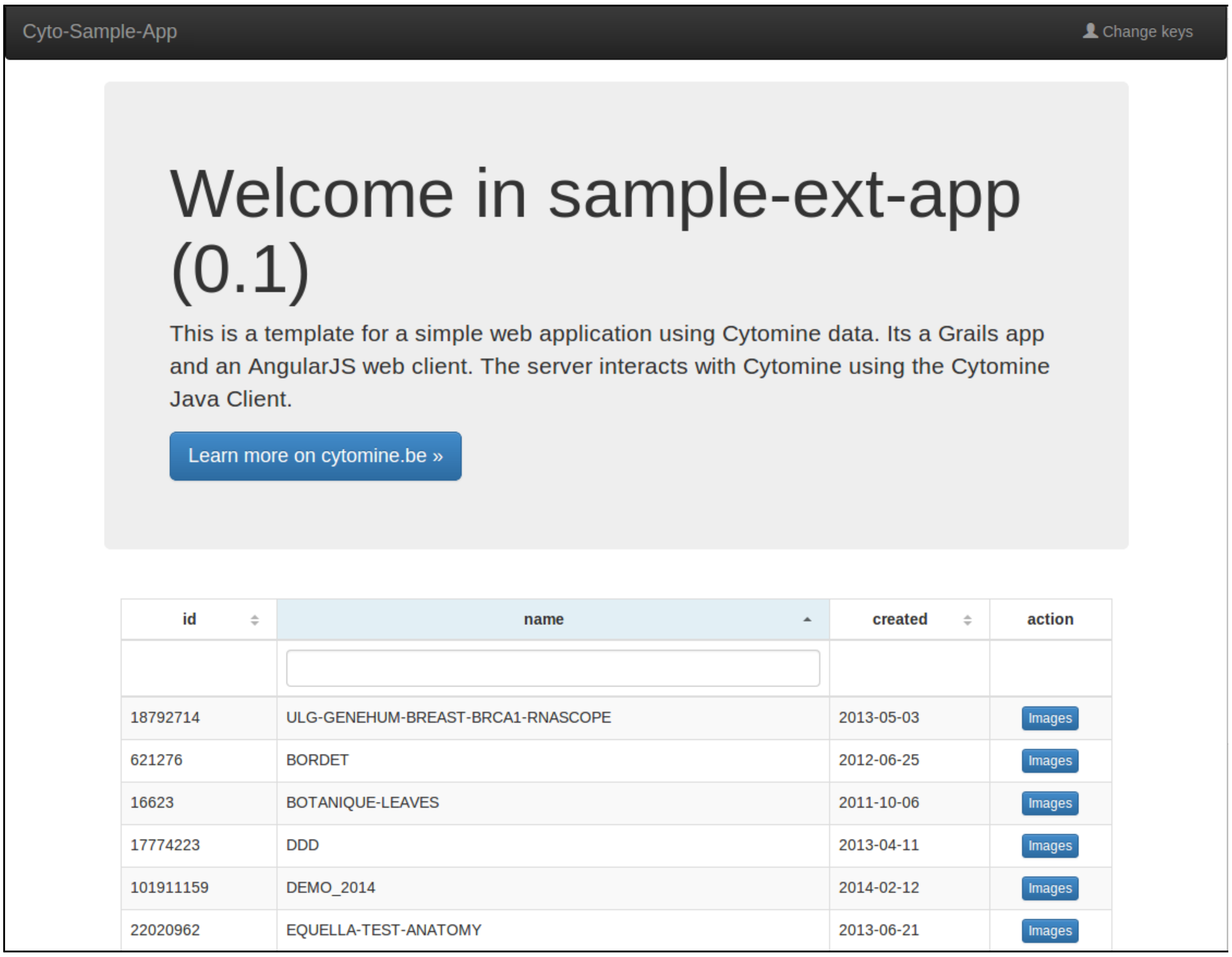
[DOC] Sample web app application using Cytomine

Introduction

The sample-ext-app is an external application that can interact with a Cytomine instance.
It is an almost empty web application to show you how to implement your own business functionnalities by creating an external app without modifying the Cytomine core.

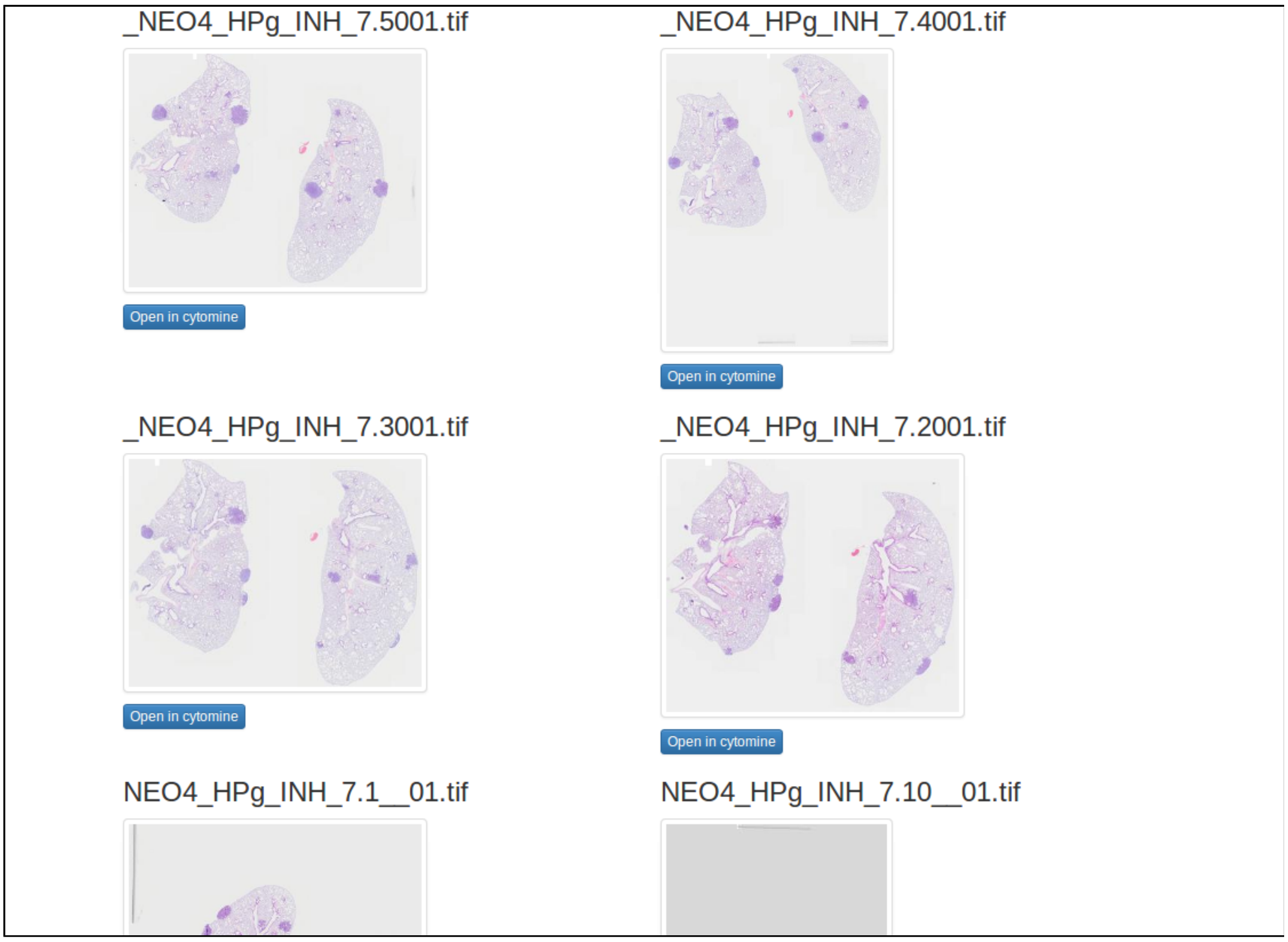
This is just a sample app that could serve as a basis for your own application using Cytomine. It first provides a form where you have to enter your Cytomine public and private keys (you can retrieve them on your Cytomine account page). Next the app lists all available projects. If you click on the image links, you will get the project images. You may click on each image button to open it on the Cytomine application (you must be logged on Cytomine). You can use similar principles to access any data stored in the main Cytomine core server instance (e.g. user annotations,...).



Welcome in sample-ext-app (0.1)

This is a template for a simple web application using Cytomine data. Its a Grails app and an AngularJS web client. The server interacts with Cytomine using the Cytomine Java Client.

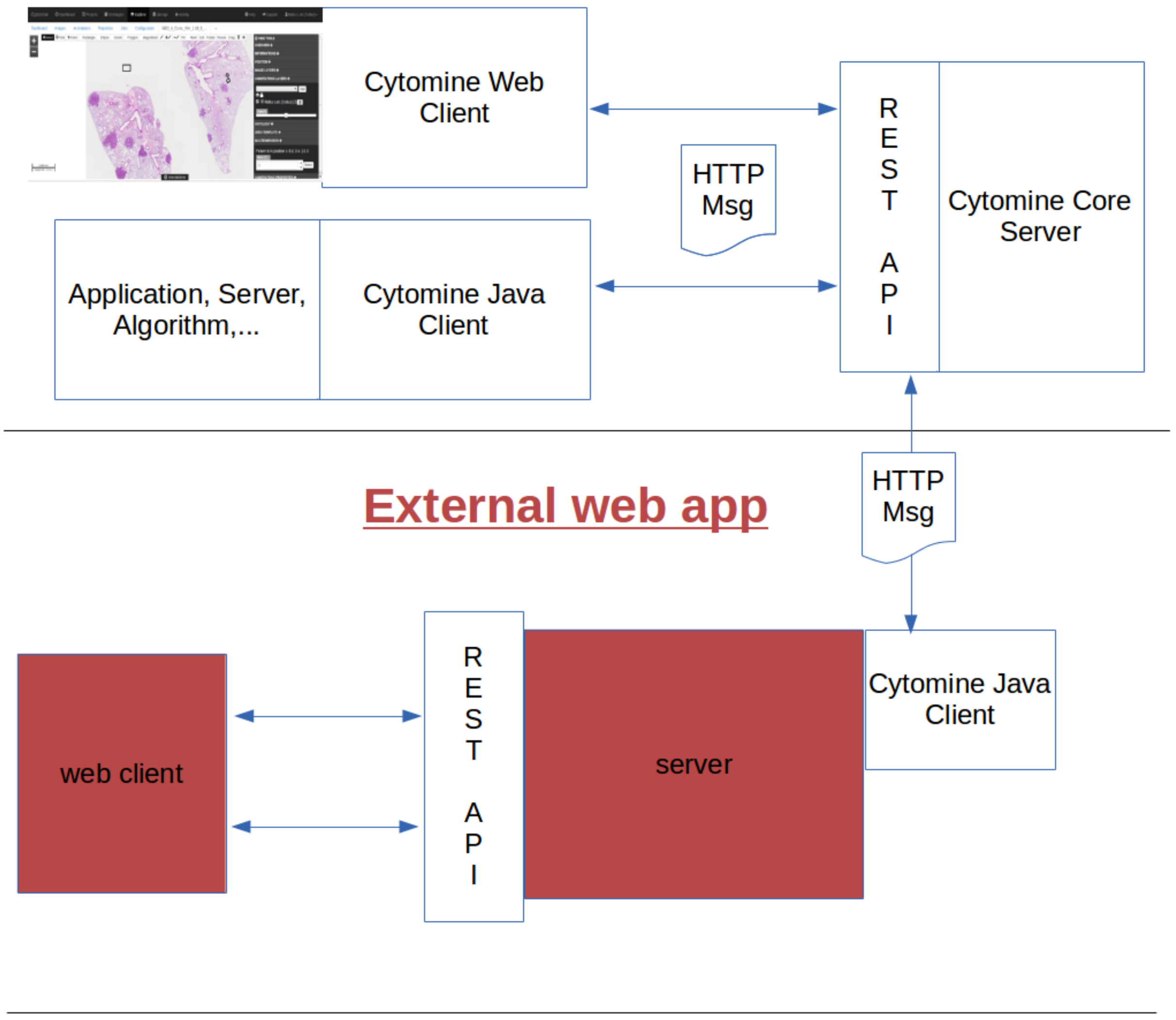
id	name	created	action
18792714	ULG-GENEHUM-BREAST-BRCA1-RNASCOPE	2013-05-03	Images
621276	BORDET	2012-06-25	Images
16623	BOTANIQUE-LEAVES	2011-10-06	Images
17774223	DDD	2013-04-11	Images
101911159	DEMO_2014	2014-02-12	Images
22020962	EQUELLA-TEST-ANATOMY	2013-06-21	Images



Before reading this article, you should:

- Be familiar with Grails web framework (following a simple tutorial will be enough),
- Be familiar with AngularJS (same as for Grails),
- Install Grails (version 2.3.5, higher should work too),
- Download the sample-ext-app sources

This is SAMPLE WEBAPP. We provide almost no error handling and no test. We provide a very simple authentification mechanism.



Technologies

We use these technologies for the server:

Name	Description	URL
Grails	Web framework to build the server	https://grails.org/
Cytomine Java Client	A client to easily interact with Cytomine REST API	No URL, the jar is in lib/ folder

For the client:

Name	Description	URL
AngularJS	Web framwork to build the client	https://angularjs.org/
Bootstrap	CSS-based design template	http://getbootstrap.com
ng-table	An external lib to easily build data tables with AngularJS	http://ngmodules.org/modules/ng-table

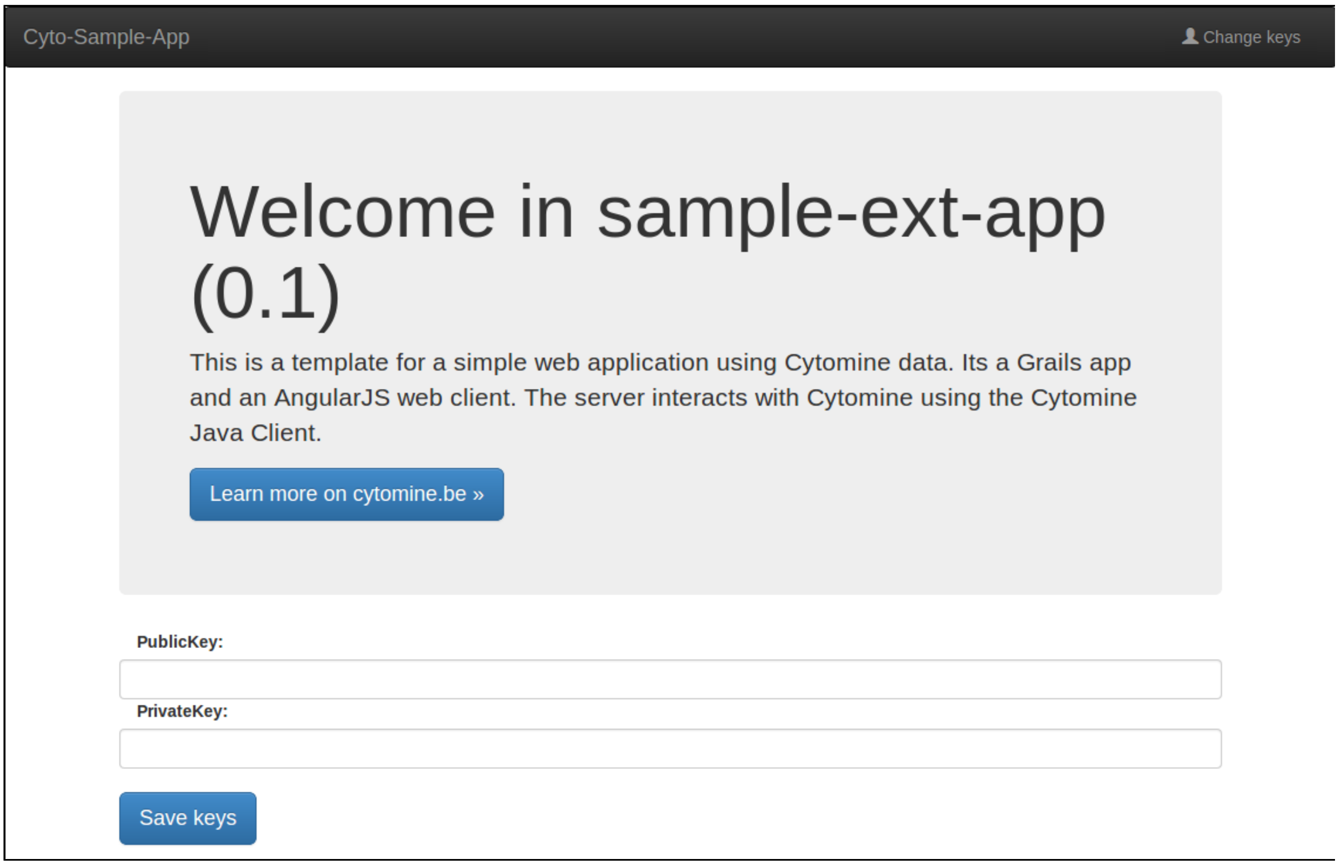
Execute

You first need a Grails 2.3.5 installed (higher versions version should be ok after upgrating).

- Move to the app directory (directory containing application.properties file),
- Update the ***grails.cytomine.host*** config option from the *grails-app/conf/Config.groovy* file if necessary (by default: "http://beta.cytomine.be").
- Run:

```
grails run-app
```

- By default, the server listens on port 8080. You should see something like this:
  Server running. Browse to http://localhost:8080/index.html
- Go to this link. You should see the form to enter your Cytomine credentials (keys). To get your keys, go to Cytomine, in your account page.



Welcome in sample-ext-app (0.1)

This is a template for a simple web application using Cytomine data. Its a Grails app and an AngularJS web client. The server interacts with Cytomine using the Cytomine Java Client.

PublicKey:

PrivateKey:

Server

This is a very light Grails server.

Config

All config data are in *grails-app/conf/Config.groovy*. These config data may be externalized in a real config file (see http://grails.org/doc/latest/guide/single.html#configExternalized). For the sample app, there are hard-coded.

```
grails.cytomine.host = "http://beta.cytomine.be"
```

The *grails-app/conf/DataSource.groovy* file contains config data for the database connection. We keep default value since we don't need a database for the sample app.

Startup

When you run the server (using *run-app* or running the war in container), Grails launch the *grails-app/conf/BootStrap.groovy* init method. This method is empty for our sample app.

Request

Each HTTP request is first processed by the *grails-app/conf/UrlMappings.groovy*. This file contains mapping rules. A mapping rule defines that for an HTTP request on path with a specific verb, the request must trigger a specific controller method.

```
"/api/project.$fomat"(controller:"cytomine"){
    action = [GET: "projects"]
}
```

This rule simply tells that when you do a GET on /api/project.json, you call the method *CytomineController.projects()* (controller name starts with a lowercase and is written without “Controller” prefix).

We have defined a filter in *grails-app/conf/SecurityFilters.groovy*. Before each request, we retrieved the cytomine credentials and we init the Cytomine connection. Each request using Cytomine data (e.g. get project lists, get images from project) must have publicKey and privateKey as parameters.

```
String publicKey = params.get("publicKey")
String privateKey = params.get("privateKey")
request['cytomine'] = new
Cytomine(grailsApplication.config.grails.cytomine.host,publicKey,privateKey,"./")
```

In *grails-app/controllers/be/cytomine/sample.CytomineController*, we've defined some methods using the Cytomine connection.

```
def projects() {
    render request['cytomine'].getProjects().list as JSON
}
```

This method simply call the *getProjects()* method from the Cytomine client instance and render the content in JSON as the request response.

To recap, if a GET method is done on */api/project.json* with the good credentials (Run http://localhost:8080/api/project.json?privateKey=...&publicKey=... in your browser):

1. The url mapping finds the good rule and calls *CytomineController.projects()* method,
2. The filters occurs just before the method call, extract keys from URL params and create a *Cytomine* instance (stored in *request['cytomine']*),
3. The method *projects()* uses the Cytomine Client to retrieve your projects. The method *getProjects()* from the client simply does a GET */api/project.json* on the Cytomine instance and returns the result as a list,
4. The method returns the project list as a JSON

Web Client

Its a basic AngularJS app. The entry point is the *web-app/index.html* file. All the routes are defined in web-app/*mainapp.js*.

URL	Content for <ng-view>	Description
By default	views/projects.html	Print projects listing view (or the keys form if public/private key are null)
/index.html#/home	views/projects.html	Same as before
/index.html#/project/:idProject/image	views/images.html	Print the project images view

The interesting part from the *index.html* file is:

```
<div ng-show="publicKey">
    <ng-view></ng-view>
</div>
<div ng-hide="publicKey" ng-include src="'views/keys.html'"></div>
```

The *ng-view* adds the content from the *mainapp.js* mapping (e.g. content from web-app/views/projects.html if /home). This content is shown only if publicKey is set in the AngularJS scope (from *mainController.js*). If not, we set the content from keys.html (a form to enter public and private keys). Keys are stored using HTML5 local storage.

Views file (html) are link with AngularJS controllers (*web-app/controllers*). Service files (*web-app/services*) are used to interact with Cytomine.