1. Foreword . . . 3
2. Introduction . . . 3
3. Release notes . . . 3
4. Types of users . . . 4
5. For users . . . 5
5.1 Search administrator . . . 5
5.1.1 Adding a new language . . . 5
5.1.2 Configuration modification . . . 6
5.1.3 Crawling Files with ManifoldCF . . . 6
5.1.4 Extracting file content in ManifoldCF and apply OCR . . . 7
5.1.5 Manage Solr configuration with Zookeeper . . . 8
5.1.6 Open local files . . . 9
5.1.7 OpenSearch . . . 11
5.1.8 Security Management . . . 13
5.1.9 Solr configuration customization . . . 15
5.1.10 Speeding up search queries . . . 18
5.1.10.1 Highlighting configuration . . . 19
5.1.11 Optimising the index size . . . 19
5.1.12 Users Management . . . 21
5.1.13 Update Datafari . . . 23
5.2 Search experts . . . 23
5.2.1 Alerts management - Mail Configuration . . . 23
5.2.2 Analytics . . . 24
5.2.2.1 Dynamic analytics . . . 24
5.2.2.2 ELK analytics . . . 24
5.2.2.2.1 Configure ELK . . . 25
5.2.2.2.2 Configure ELK (Datafari 3.1 and above) . . . 29
5.2.2.2.3 Content Analysis . . . 31
5.2.2.2.4 Content Analysis Over Time . . . 31
5.2.2.2.5 Usage Statistics . . . 32
5.2.2.3 Search navigation analytics . . . 33
5.2.3 Autocomplete configuration . . . 34
5.2.4 Connectors configuration . . . 35
5.2.5 Crawling . . . 36
5.2.5.1 Add a JDBC connector (MySQL, Oracle) . . . 46
5.2.5.2 Add the JCIFS Connector to Datafari . . . 49
5.2.5.3 Advanced Crawling website . . . 50
5.2.5.4 Reset a MCF job . . . 55
5.2.6 Deduplication management . . . 57
5.2.7 Facets management . . . 57
5.2.8 Favorites management . . . 59
5.2.9 Fieldweights . . . 59
5.2.10 Likes management . . . 60
5.2.11 Promolinks management . . . 60
5.2.12 Query Elevator . . . 62
5.2.12.1 Admin UI feature . . . 62
5.2.12.2 Search View Feature . . . 63
5.2.13 Stopwords management . . . 64
5.2.14 Synonyms . . . 65
5.3 Search users . . . 68
5.3.1 Alerts . . . 69
5.3.2 Autocomplete . . . 70
5.3.3 Deduplication . . . 70
5.3.4 Facets . . . 71
5.3.5 Favorites . . . 71
5.3.6 Highlighting . . . 72
5.3.7 Likes . . . 72
5.3.8 Promolinks . . . 73
5.3.9 Searching . . . 74
5.3.10 Stopwords . . . 78
5.3.11 Alerts (from Datafari 3.1) . . . 79
6. For developers . . . 79
6.1 Add new icon for search results . . . 79
6.2 Ajaxfrancelabs graphical framework . . . 79
6.2.1 Displaying thumbnails for autocomplete and results list . . . 80
6.2.2 Framework architecture and mechanism . . . 82
6.2.3 Quickly changing the main colors . . . 90
6.2.4 Tutorial . . . 90
6.2.5 Widgets and Modules . . . 115
6.3 Add custom fields in Datafari and display them . . . 129
6.4 Alerts - Architecture and mechanism . . . 131
6.5 Alerts - technical doc (from 3.1) . . . 132

6.6 Architecture . . . 133
6.7 Custom Solr configuration . . . 134
6.8 Deduplication : Technical note . . . 135
6.9 ELK . . . 136
6.10 Faceting . . . 138
6.10.1 Manually adding facets to the Datafari UI . . . 139
6.11 Fieldweights - Architecture and mechanism . . . 146
6.12 Likes and Favorites . . . 147
6.13 Logging . . . 151
6.13.1 ELK logs exploitation . . . 152
6.14 Save search - technical doc . . . 153
6.15 Security . . . 153
6.15.1 CustomCombinedRealm - Architecture and mechanism . . . 154
6.15.2 Understanding Security Realms . . . 156
6.16 Semantics . . . 159
6.16.1 Link an ontology . . . 159
6.16.2 Promolinks - architecture and mechanism . . . 161
6.16.3 Synonyms and stopwords . . . 162
6.17 Setup Development Environments . . . 163
6.17.1 Setup development environnement with Eclipse (Linux) . . . 163
6.17.1.1 Setup development environment with Eclipse (Linux) for Datafari 2.0 . . . 169
6.17.1.2 Setup development environment with Eclipse (Linux) for Datafari 2.x - Maven . . . 176
6.17.1.3 Setup development environment with Eclipse (Linux) for Datafari 3.1 . . . 193
6.17.1.4 Setup development environment with Eclipse (Linux) for Datafari 3.x - Maven . . . 197
6.18 Unit Testing . . . 214
6.19 Utility logs . . . 216
6.19.1 Core Monitoring logs . . . 216
6.19.2 Statistics logs . . . 218
6.20 Widgets and CSSs good practices . . . 219
6.21 Search Index . . . 219
6.22 Data Schemas . . . 220
6.22.1 User data . . . 220
6.22.1.1 Likes and Favorites Schema . . . 220
6.22.1.2 Preferred language . . . 221
6.22.1.3 Roles . . . 222

Foreword

Datafari is the ideal product for those who want to analyse and search through their data, while using advanced open source technologies. Datafari combines Apache ManifoldCF and Apache Solr, to allow its users to search through many different and diverse data sources: their file shares, their cloud shares (dropbox, google drive…), their databases, but also their emails and many more. Available as community or enterprise version, Datafari has its specificities:

- Its open source licence is non aggressive as its under Apache v2 licence: you are free to do whatever you want with it, you just need to mention that you're using it.
- It combines several popular Apache projects: Solr, ManifoldCF, and Cassandra, which guarantees stability over time.
- It provides analytics capabilities, through the integration of Banana and ELK.

Introduction

Be it for users or for companies, the amount of data is increasing exponentially. On top of it, the challenge of the cloud multiplies the number and heterogeneity of systems hosting data. Search engines are here to tackle this challenge. They can connect to many systems, propose a single view on the entirety of available data. Contrarily to web search engines, enterprise search engines are 100% controlled by you, and have access to data which belong to you, and which are not necessarily public. These search engines guarantee the security of access to these data.

Among the many existing solutions, a majority is proprietary : you acquire a licence, you pay for support, and you buy the integration. However, a large chunk on the basic functionalities are now available as open source, and don't require massive investments. The big players of the web have made this choice: Linkedin, eBay, Twitter, Salesforce, Kelkoo… They all use open source tools.

However, the most well knowned tools, Apache Lucene and Apache Solr, are only the heart of a search solution. They do not provide any framework to manage the access to the data sources, they do not handle security, and they do not manage backup or monitoring activities. Other complementary open source projects are available, but the integration is not always easy. This is where Datafari is standing: it integrates these technologies, using as much as possible projects using an Apache licence (or equivalent), in order to remain non aggressive for companies.

We wanted to offer to the community an easy to use tool, affordable – even free – for many use cases, but also able to scale up in order to manage hundreds of millions of indexed documents, thanks to SolrCloud. You will find in this document an overview of Datafari, in order to better understand it, use it, even extend it. Obviously, we encourage the users community to help us in the evolution of this open source tool. In the architecture section, you will discover the architecture of Datafari and its main components. In the crawling section, we will detail the crawling section and its usage. In the indexing section, we present the content indexing part of the search engine. In the search section, we cover the search part of the search engine, which means how do the queries and search algorithm work. In the user interfaces section, we present the default user interface used by Datafari. In the security section, we cover the security challenges. In the analytics section, we will learn how to monitor Datafari. In the use cases section, we present use cases which you can use as a way to kickstart you own projects.

User documentation
In this user documentation section, you will learn all you need about Datafari. Whether you are the search manager, a standard or the Datafari system administrator, we cover in this section the functionnalities of Datafari.

Developer documentation
In this developer documentation section, you will learn how to set up a proper development environment, as well as the test environment into place. For now, only the development environment is documented.

Release notes

Target release	3.1
Document status	RELEASE
Document owner	Olivier Tavard
Designer	
Developers	Olivier Tavard, Aurélien Mazoyer, Julien Massiera
QA	Olivier Tavard

Major changes from v3.0

New and updates:

- Tika updated to version 1.14
- Manifold CF updated to version 2.5
- Improved security for Active Directory connections
- Improved security for Tomcat JNDI Realm
- Auto-configuration of ELK on the first start of Datafari
- New fields for EXIF images metadata
- New preferred language feature

Bug fixes: (check github for the detailed history):

- Bug fixes for Likes/Favorites
- Query elevator fix
- Synonyms fix (press enter bug)
- Fix Likes not inserted into Cassandra
- Fix for spellcheck and hyphen words

Major changes from v2.0

- Solrcloud activated by default
- Graphical monitoring of the Datafari clusters
- Query elevator admin functionality
- Remodeling and aggregation of the logs, centralised towards ELK.

Major changes from v1.0

- Ajaxfrancelabs has been completly revamped:
  - New responsive design
  - New widgets
- Apache Cassandra is integrated to handle user management
- Migration to Solr 5
- The admin UI has been completly revamped:
  - Harmonisation of the different components, leveraging the great Devoops V2 admin theme
- New stable functionnalities:
  - User management
  - Connection to AD through LDAP
  - Promolinks
  - Synonyms
  - Size limitation
  - Autocomplete configuration
- New experimental functionalities:
  - Likes and favorites
  - Alerts
  - Deduplication
  - OCR
  - Facets configuration
  - Field based boosts

Types of users

Starting with v2.0, Datafari proposes four distinct roles, which have access to different sets of functionalities.

Those four roles are: **anonymous search user**, **connected search user**, **search expert**, **search administrator**. The last two roles have been created to distinguish the activity of optimising the search engine from a relevance/usage perspective and from a technical perspective. Some details about what these four roles represent.

1. Anonymous search user :

This is the most basic type of user. This user can only use the search functionality, and gets only two possible displays: the search bar (including autocomplete), and the results list (including facets). He has no access to functionalities such as alerts, statistics, saving favorites, or liking results. Also, since he is anonymous, he can only search through public data. Anything that has been labelled as restricted in terms of visibility (e.g. through ACLs for files) will not be searchable. If searching through these secured documents is required, the search user will need to switch to the "connected search user" role.

2. Connected search user :

This is a role similar to the anonymous search user role, except that it proposes more search functionalities. Obviously, the user can search using the search bar (including autocomplete) and the results list (including facets). Contrarily to the anonymous user, he has access to functionalities such as alerts and saving favorites. To reach these functionalities, the searcher needs to go in his administration panel, where he will see the required tabs. Also, he can search through secured documents. This means that when searching, Datafari does a security check on the documents and the user, and displays in the search result all the documents that the connected search user is allowed to see. Check the pages detailing the alerts functionnality and likes and favorites functionality to learn how to use it.

3. Search expert :

The search expert is responsible for maintaining the relevance of Datafari, monitoring its usage, and managing its community of users. He ensures that queries with no clicks are not left alone. He checks the vitality of the system in terms of queries. He manages the promolinks. Beyond Datafari, he also ensures that people are aware of the tool, and he trains them if necessary. The expert has access to a range of functionalities from his administration interface: statistics through a graphical dashboard, promolinks management, synonyms and stopwords management, field based boosts, documents elevation, duplicate identifier, OCR activation, user management. We don't detail these functionalities here, they have dedicated sections in the documentation.

4. Search administrator :

The admin is responsible for maintaining the datafari system. He is the one in charge of restarting the server, monitoring the performances, checking the logs, ensuring that the hardware specs are still compatible with the number of documents indexed and the number of queries per second. He has a range of administration functionalities available from his admin panel: statistics through a graphical dashboard, logs search dashboard, system alerts (both for connectors and for the search engine), user alerts overall performance, user alerts scheduler activation, size limitations for the indexing phase. These functionalities are detailed in their respective sections of the documentation.

For users

Welcome to the users documentation about Datafari. Datafari is a packaged enterprise search solution, in open source under Apache v2 Licence, combining Apache Solr, Apache ManifoldCF, ELK and Ajaxfrancelabs. To learn more about Datafari as a product, you can go to the main page of Datafari. Or if you want to know more about the creators - and main contributors - of Datafari, visit the corporate site of France Labs.

For clarity, we have decided to split this documentation based on the types of users. This structure may not be optimal, but at least it gives a bit of structure.

Search administrator

This section lists the management functionalities of Datafari for the search administrator.

The admin is responsible for maintaining the datafari system. He is the one in charge of restarting the server, monitoring the performances, checking the logs, ensuring that the hardware specs are still compatible with the number of documents indexed and the number of queries per second. He has a range of administration functionalities available from his admin panel: statistics through a graphical dashboard, logs search dashboard, system alerts (both for connectors and for the search engine), user alerts overall performance, user alerts scheduler activation, size limitations for the indexing phase. These functionalities are detailed in their respective sections of the documentation.

Adding a new language

For now Datafari is preconfigured for English, French, Italian, Arabic and Brazilian Portuguese. Still, its aim is to have a global reach, so the steps to enable additional languages is rather straightforward and can be found here. In case you did it, please contact us so that we can integrate it in the next releases of Datafari !

Step-by-step guide

For the internationalization of the user interface, we use the i18n java library:

1. The folder to store the i18n config files is here : **datafari/WebContent/js/AjaxFranceLabs/locale**. Just open it and add your new language. For example if you add German translation, put here a file named de.json.
2. Add the new langage in the Java class com.francelabs.datafari.utils.LanguageUtils.java :

```
public static final List<String> availableLanguages =
Arrays.asList("en", "fr", "it", "ar");
```

3. Add the new language in WebContent/js/AjaxFranceLabs/i18njs.js :

```
availableLanguages : [ 'en', 'fr', 'it', 'ar' ],
```

4. Finally launch the ant script datafari-dev.xml to take into Datafari the modifications.

For the internationalization of the language detection, indexing and search by the Datafari Solr engine, follow these steps:

1. Modify the dedicated Solr updateprocessor which is declared in the **DATAFARI_HOME/solr/solr_home/FileShare/conf/solrconfig.xml**, that you can find at *updateRequestProcessorChainDatafari,* which detects the languages based on the fields content and title. By default, we use English and French. In order to add a new language, modify the new language to the element *"langid.whitelist*".
2. Modify the Solr schema to handle the new language, which you will find at **DATAFARI_HOME/solr/solr_home/FileShare/conf/schema.xml**. You will notice that we already have the following two fields which are language specific, namely content and title. Therefore, we have "*content_en*", "*title_en*", "*content_fr*", "*title_fr*". You need to create your specific "*content_xy*" and "*title_xy*" fields for your new language.
3. Modify the searchrequesthandler named select in the **DATAFARI_HOME/solr/solr_home/FileShare/conf/solrconfig.xml**. There, change the parameters *qf* et *pf* : put the following new fields: *title_xy* and *content_xy* to the existing chain of parameters.
4. Now you can restart your Datafari for the changes to be taken into account.

> You can either send us your new language either directly or using github, either way is fine by us as the modifications are not huge. We will send you a **cool Datafari T-Shirt** if you share that with us, so that you can show the community you are a real Datafarian 🙂

Related articles

- Adding a new language

Configuration modification

You can edit the file DATAFARI_HOME/tomcat/conf/datafari.properties to edit the configuration of Datafari.

- ALLOWLOCALFILEREADING

It is related to the content that you index with ManifoldCF : local filesystem crawling.
It allows you to change the way that you can open local files in the Datafari Search interface.

If the setting is at **true**, it means that you do not need any extension installed on your browser to open local files (see this page for information about it). Indeed Datafari opens the file for you and sends it to you directly when you click on the document link.

If the setting is at **false**, you need to have specific settings enabled on your browser to open the document because it becomes a normal link to the resource.

Crawling Files with ManifoldCF

⚠ **The date of LibreOffice documents is not crawled correctly.**
Dates extraction is not available on plain text documents.

⚠ **The local file crawler is a DEMO connector that shouldn't be used for production environments.**
It has a bug in the extraction of the modification date of the documents (making Date facet not working).

A good alternative to the local file crawler is the "Windows share" (a.k.a. JCIFS connector), that may be used to crawl files in a file shared directory (it works under Linux as well 🙂).
Documentation on how to setup a samba share on Linux here

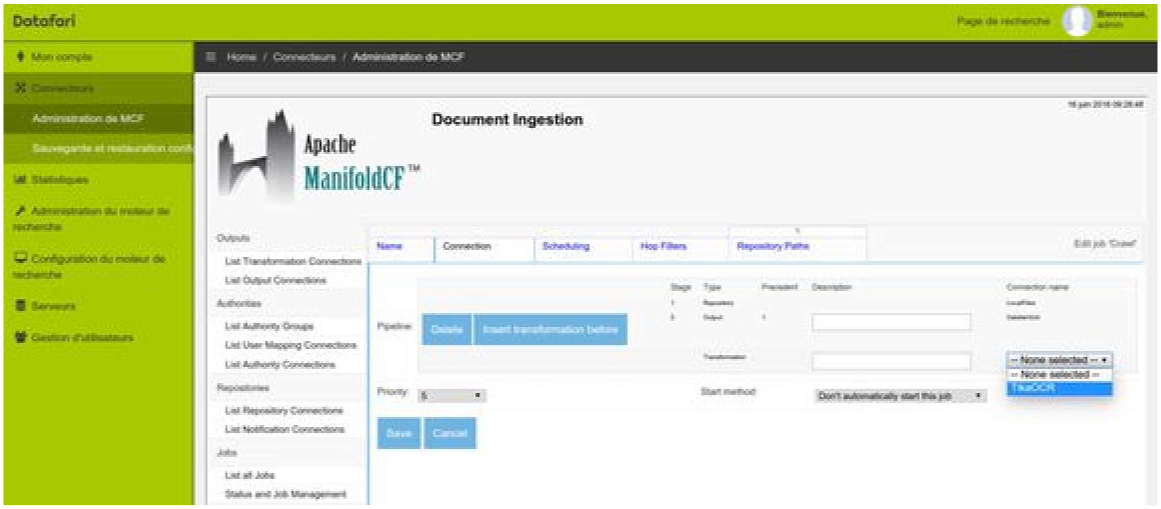
Download jcifs-1.3.xx.jar from http://jcifs.samba.org/src/ to DATAFARI_SOURCE_DIR/mcf/mcf_home/connector-lib-proprietary
Then edit the file DATAFARI_SOURCE_DIR/mcf/mcf_home/connectors.xml and uncomment the line :
<!--repositoryconnector name="Windows shares" class="org.apache.manifoldcf.crawler.connectors.sharedrive.SharedDriveConnector"/–>

To take into account your modification, do as follows:

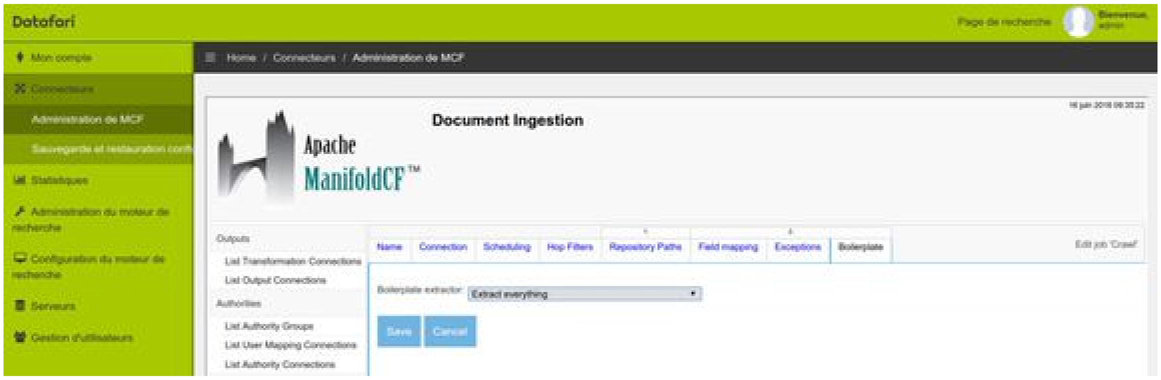
- If you have already installed Datafari:
  - stop Datafari if running,
  - change the installation status of Datafari to "STATE=installed" in the file DATAFARI_INSTALL_DIR/bin/common/init_state.properties,
  - start Datafari.
- If you are running Datafari in development environment:
  Restart Datafari.

Extracting file content in ManifoldCF and apply OCR

By default, Datafari uses the Apache Solr Extracting Handler (aka Solr Cell) that leverages Apache Tika to extract the content that will be indexed from the crawled files. To limit the resource consumption (especially the network if ManifoldCF is installed on an external server), it is possible to use Tika directly in ManifoldCF. In this case, the content is extracted directly in ManifoldCF, and only the content that should be indexed is sent to Apache Solr. A Tika Transformation Connection is configured by default in Datafari. To use it, you simply have to add it to your crawling job in ManifoldCF :



Then, click on "Insert Transformation Before". You can now go to the tab "Boilerplate" and select "Extract Everything" :



Boilerplate extractor: Extract everything

Now your crawling job is ready to use Apache Tika directly in ManifoldCF to extract the content of the crawled files. You can also use Tesseract OCR to analyse images in order to extract data from image and pdf files. Tesseract OCR is bundled in Datafari. In order to make the ManifoldCF TikaOCR transformation connector able to use Tesseract to do the OCR analysis, you have to change the property "**OCR**" in */opt/datafari/tomcat/conf/datafari.properties* from "**false**" to "**true**". Then restart Datafari.

Manage Solr configuration with Zookeeper

The search administrator has the task to manage the Solr configuration via upload it or download it from/to Zookeeper.

Indeed with Datafari 3+ and the SolrCloud mode by default we cannot directly edit the configuration on the file system and reload the core. The Solr configuration is now stored in ZK.

1.Modify the configuration, send it back to Zookeeper and reload the Solr collection

- Edit the Solr configuration
    - By the UI admin

You can easily modify some parameters of the Solr configuration by the UI admin like the synonyms.

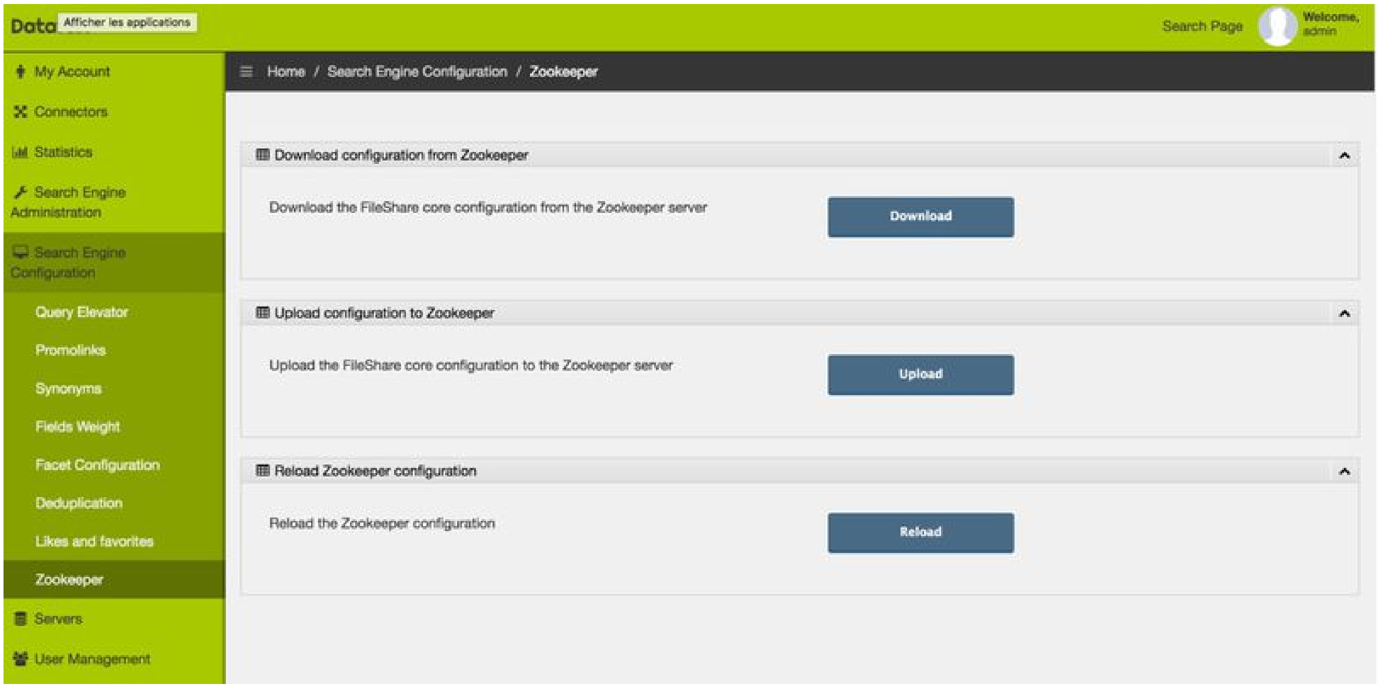
    - by the file system

You can change the configuration in the filesystem directory to change precise parameters that you cannot change directly in the UI for example. The path is : /opt/datafari/solr/solrcloud/FileShare/conf

- Send the new configuration to ZK

When you are done with your changes, you have to send the new configuration to ZK.

You can do it directly by the UI : click on the tab Zookeeper located in Search engine configuration.



Download configuration from Zookeeper

Download the FileShare core configuration from the Zookeeper server

Upload configuration to Zookeeper

Upload the FileShare core configuration to the Zookeeper server

Reload Zookeeper configuration

Reload the Zookeeper configuration

Now click on the button Upload configuration to ZK.

Doing that, your modified configuration is now stored in ZK.

The final step is to load it into Solr.

- Reload Collection

We now tell to the Solr collection FileShare to reload itself to have the latest version of Solr configuration from Zookeeper.

For that, click on the button Reload.

Be careful to not do it during heavy load of Solr, especially during indexing time.

2. Download Solr configuration from ZK

If you want to replace the Solr configuration present on the filesystem by the Solr configuration present in ZK, you can do it also by the Zookeeper Admin UI.

Click on the button Download. The Solr configuration present in /opt/datafari/solr/solrcloud/FileShare/conf will be replaced by the Solr configuration from the ZK server.

Note : if you add new files on the local file system they will not be deleted from the Solr configuration from the ZK server. in order to have a clean copy of the Solr configuration you can erase all the content of the folder /opt/datafari/solr/solrcloud/FileShare/conf and after that download the configuration from ZK server.

Open local files

1. Open local files with Chrome
2. Open local files with Firefox
3. Open local files with Internet Explorer

1. Open local files with Chrome

By default (and for security !), Chrome forbids the opening of urls pointing towards local files, from a web browser. There is a workaround using the Localinks extension:

https://chrome.google.com/webstore/detail/locallinks/jllpkdkcdjndhggodimiphkghogcpida

Then activate the extension.

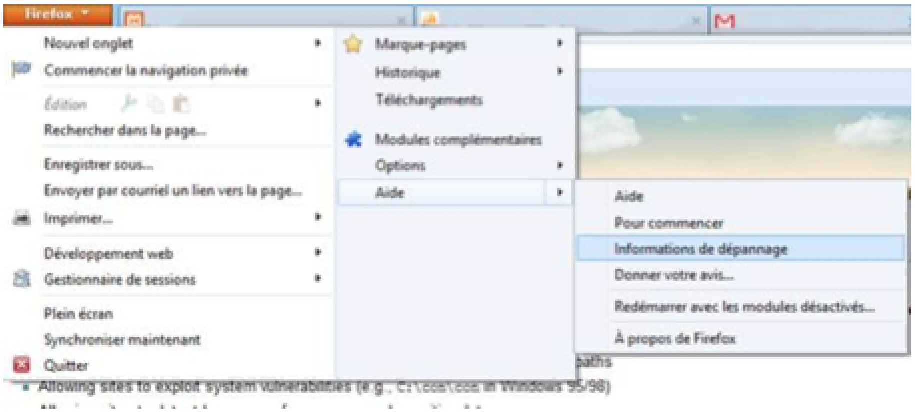
2. Open local files with Firefox

Again, by default and for security, Firefox forbids the opening of URLs pointing at local files, from a web browser. Here is a work around:

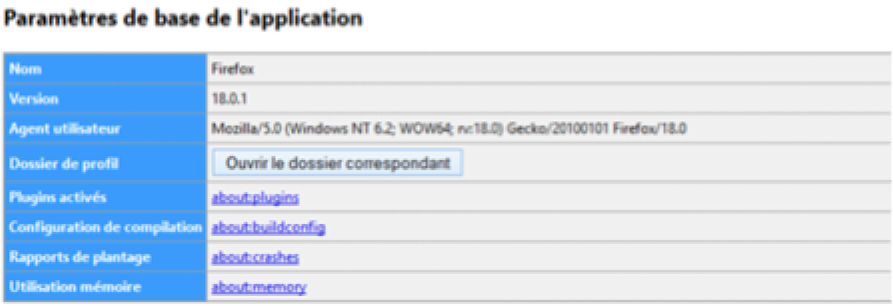
- *Open the AppData folder*

To locate it easily in Firefox, click on :

Firefox->Help->Debug help



Then click on Open the corresponding folder



Paramètres de base de l'application

Nom	Firefox
Version	18.0.1
Agent utilisateur	Mozilla/5.0 (Windows NT 6.2; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Dossier de profil	Ouvrir le dossier correspondant
Plugins activés	about:plugins
Configuration de compilation	about:buildconfig
Rapports de plantage	about:crashes
Utilisation mémoire	about:memory

The path sould like the following : C:\Users\Utilisateurs\AppData\Roaming\Mozilla\Firefox\Profiles\xxx.default

2 possibilities :

- *Either there is already a user.js file, to which you need to add the following lines :*

user_pref("capability.policy.policynames", "localfilelinks");

user_pref("capability.policy.localfilelinks.sites", "http://192.168.1.88:8080");

user_pref("capability.policy.localfilelinks.checkloaduri.enabled", "allAccess");

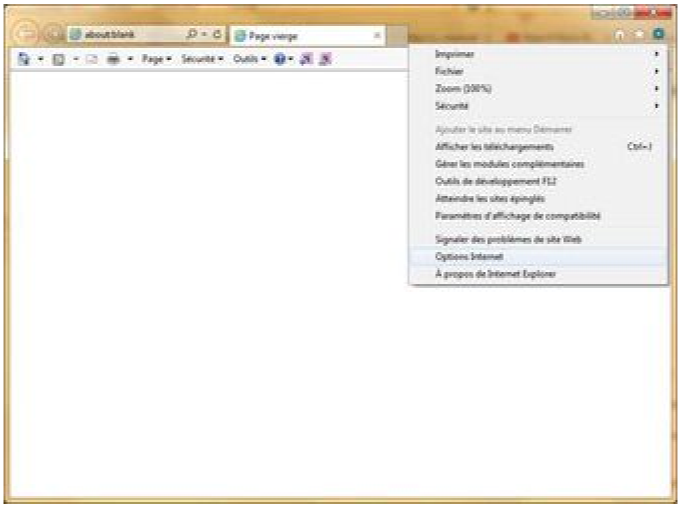
- *Or there is no file named user.js, in which case you need to create it and put the lines detailed above.*

You then need to copy user.js into the folder, to stop Firefox (check that the process has been killed) and restart it. The local files should open normally.

3. Open local files with Internet Explorer

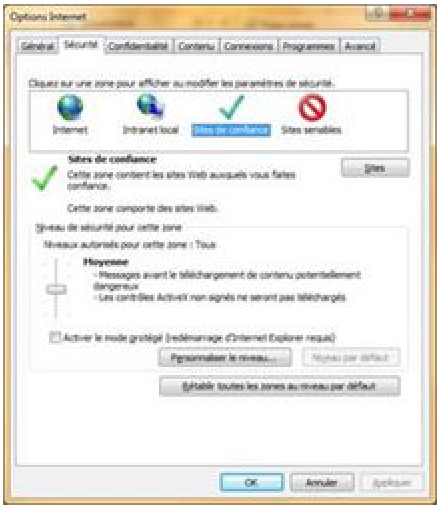
Open Internet Explorer

Go to Internet Options


about:blank
Page vierge
Page ▾
Sécurité ▾
Outils ▾
Imprimer
Fichier
Zoom (100%)
Sécurité
Ajouter le site au menu Démarrer
Afficher les téléchargements
Ctrl+J
Gérer les modules complémentaires
Outils de développement F12
Atteindre les sites épinglés
Paramètres d'affichage de compatibilité
Signaler des problèmes de site Web
Options Internet
À propos de Internet Explorer

Go to the security tab

Click on Trusted Sites



Cliquez sur une zone pour afficher ou modifier les paramètres de sécurité.

Sites de confiance

Cette zone contient les sites Web auxquels vous faites confiance.

Cette zone comporte des sites Web.

Niveau de sécurité pour cette zone

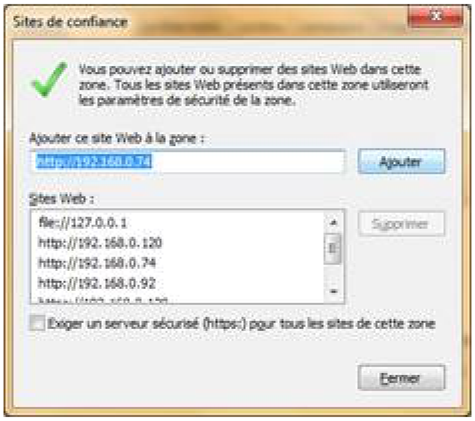
Niveaux autorisés pour cette zone : Tous

Moyenne

- Messages avant le téléchargement de contenu potentiellement dangereux
- Les contrôles ActiveX non signés ne seront pas téléchargés

☐ Activer le mode protégé (redémarrage d'Internet Explorer requis)

Click on Sites and add the ip address of your datafari, starting with http:// (do not add the port number, just the IP)



Vous pouvez ajouter ou supprimer des sites Web dans cette zone. Tous les sites Web présents dans cette zone utiliseront les paramètres de sécurité de la zone.

Ajouter ce site Web à la zone :

http://192.168.0.74

Sites Web :

file://127.0.0.1
http://192.168.0.120
http://192.168.0.74
http://192.168.0.92

☐ Exiger un serveur sécurisé (https:) pour tous les sites de cette zone

Restart your windows explorer, do another search and voila !

OpenSearch

In this page, we will show you how to generate OpenSearch standard responses from your Datafari's Solr indexes.

Therefore you can use it on your Windows explorer windows directly if you want.

You will need two files in order to do that :

- A OSDX file
- A XSLT file

The files are present in DATAFARI_SRC/dev-tools/opensearch

On your Datafari server

XSLT

We need now to transform the response sent by Solr with the XSLT writer to be compliant with the OpenSearch format.

To do that, we add the opensearch.xsl file in DATAFARI-SRC/solr/solr_home/FileShare/conf/

The fields displayed will be the title, the last modification date and the snippet highlighted for each document.

To take the file into account, you need either to reload the FileShare Solr core or restart Datafari.

On your Windows PC desktop

OSDX file

Copy the file datafari.osdx into your PC. This file contains the Solr connection information.

Edit the file to change the address of your Datafari server :

```
<Url type="application/rss+xml"
template="http://localhost:8983/solr/FileShare/opensearch?q={searchTerms
}&amp;wt=xslt&amp;tr=opensearch.xsl"/>
```

NB : in this example, we allow remote access of Solr and we do not use the search proxy which is not a recommanded way to do it in production.

To enable remote access of Solr, edit the file DATAFARI_SRC/solr/server/etc/jetty-http.xml and comment the line :

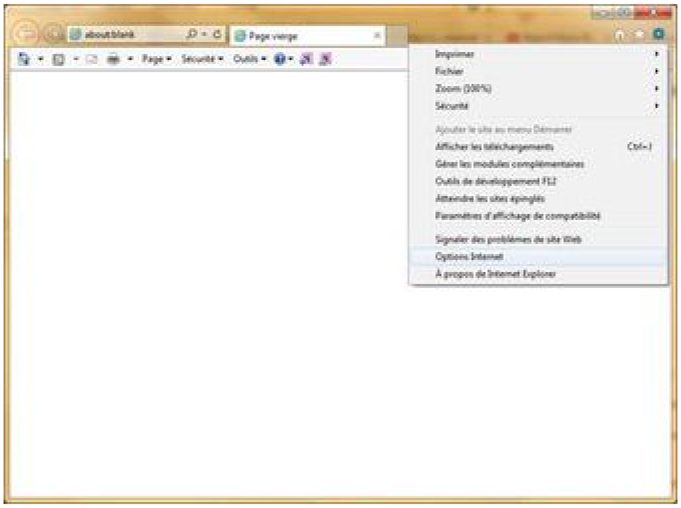
```
<Set name="Host">127.0.0.1</Set>
```

Then, to install it on your Windows system, just double click on it. You will see a new shortcut on your Windows explorer.

In case you get an “element blocked by Internet security parameters” error in windows explorer when searching, do the following:

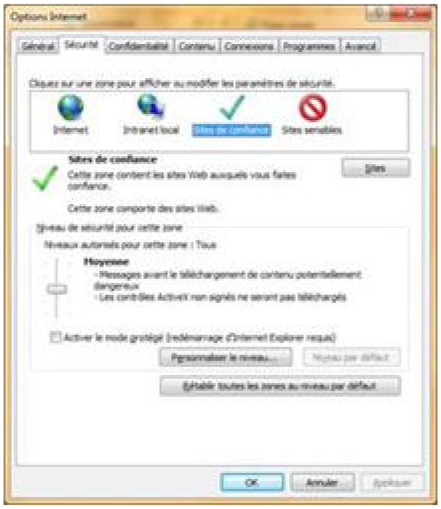
Open Internet Explorer

Go to Internet Options


about:blank
Page vierge
Page ▾ Sécurité ▾ Outils ▾
Imprimer
Fichier
Zoom (100%)
Sécurité
Ajouter le site au menu Démarrer
Afficher les téléchargements Ctrl+J
Gérer les modules complémentaires
Outils de développement F12
Atteindre les sites épinglés
Paramètres d'affichage de compatibilité
Signaler des problèmes de site Web
Options Internet
À propos de Internet Explorer

Go to the security tab

Click on Trusted Sites



Cliquez sur une zone pour afficher ou modifier les paramètres de sécurité.

Sites de confiance

Cette zone contient les sites Web auxquels vous faites confiance.

Cette zone comporte des sites Web.

Niveau de sécurité pour cette zone

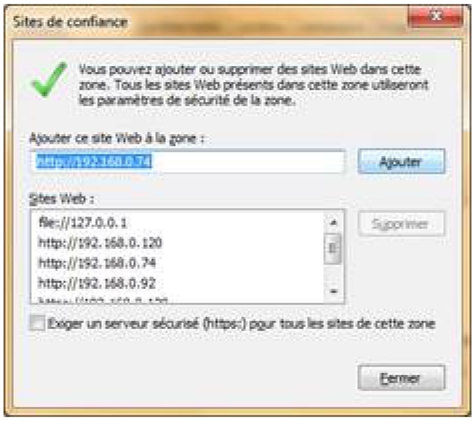
Niveaux autorisés pour cette zone : Tous

Moyenne

- Messages avant le téléchargement de contenu potentiellement dangereux
- Les contrôles ActiveX non signés ne seront pas téléchargés

☐ Activer le mode protégé (redémarrage d'Internet Explorer requis)

Click on Sites and add the ip address of your datafari, starting with http:// (do not add the port number, just the IP)



Vous pouvez ajouter ou supprimer des sites Web dans cette zone. Tous les sites Web présents dans cette zone utiliseront les paramètres de sécurité de la zone.

Ajouter ce site Web à la zone : http://192.168.0.74

Sites Web :

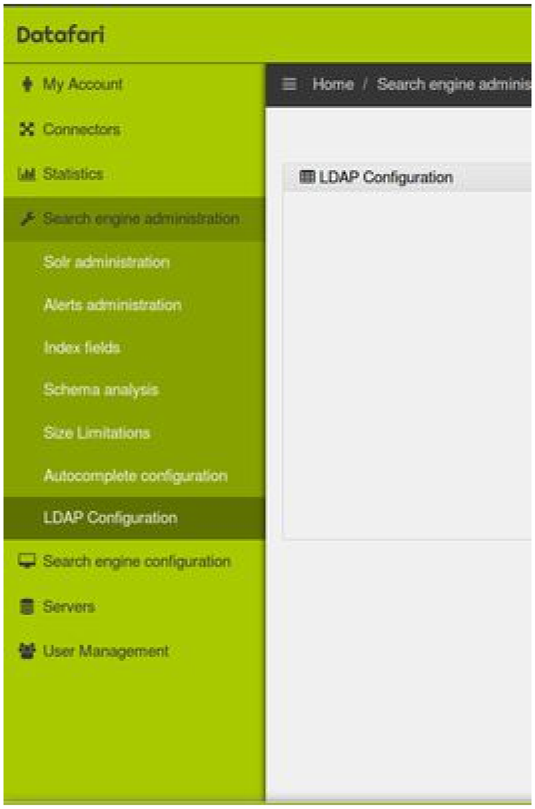
file://127.0.0.1
http://192.168.0.120
http://192.168.0.74
http://192.168.0.92

☐ Exiger un serveur sécurisé (https:) pour tous les sites de cette zone

Restart your windows explorer, do another search and voila !

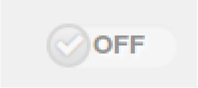
Security Management

As mentioned in the section about Realms in the Datafari developers documentation, the *JNDIRealm* allows Datafari to use an AD for authentification, via an LDAP call. This functionality requires a specific configuration for the LDAP used. To set that, the administrator can use the admin page “*LDAP Configuration*” in the section “*Search engine administration*” (see the image below).



This functionality requires that you have the Search admin role. Once logged in and into the admin environement, go to the left menu. In “*Search engine administration*”, click on the LDAP configuration submenu.

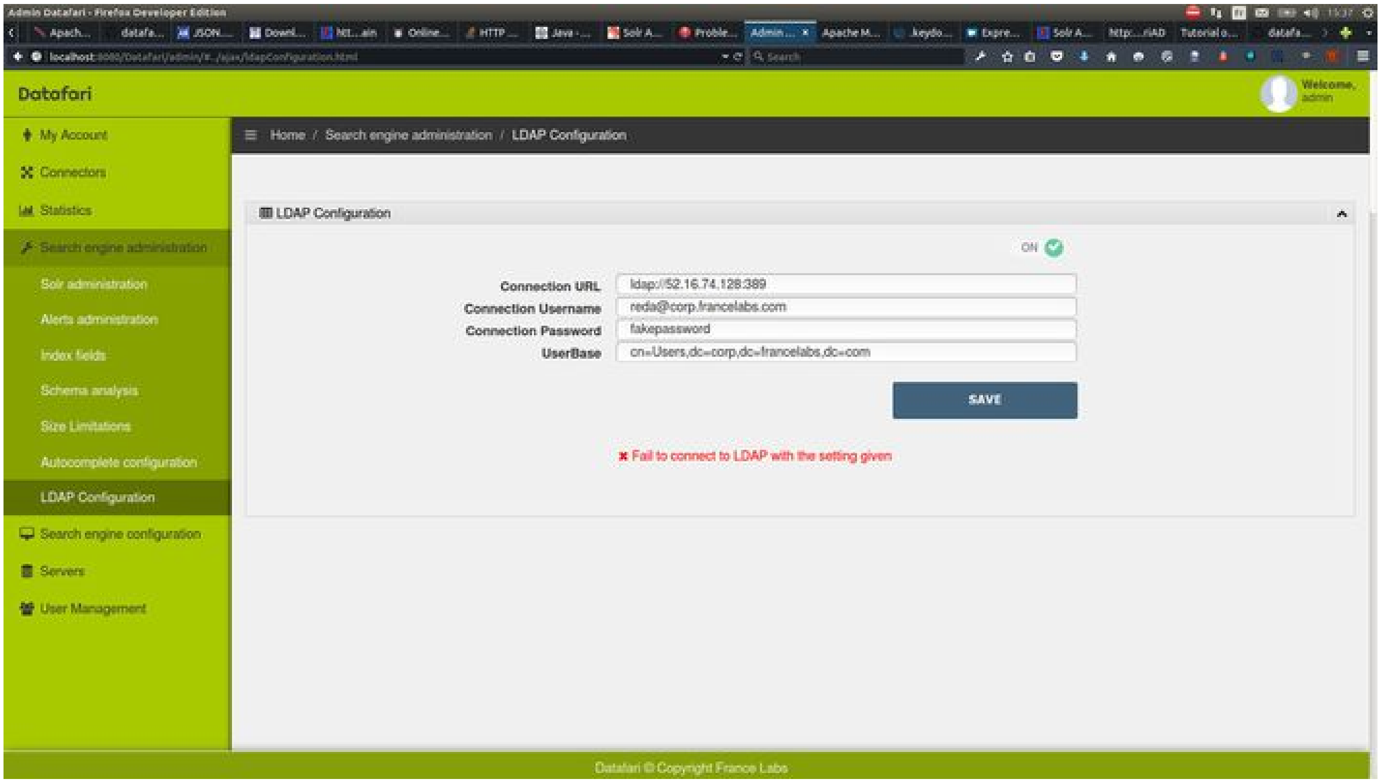
The first time you open this page, the switcher is off.



This means that the ADconfiguration is not used. By clicking on the switcher you will activate the process. Datafari will then be integrated with your AD. This click is not sufficient enough to make AD work, you also need to configure it correctly. Configure the following:

- “*connection URL”* following this syntax: ldap://URLTOSERVER:PORT.
- “*userBase”:* This is the path to root. Please follow this syntax: cn=GROUPOFUSERS,dn=ROOT2,dn=ROOT2...
- “*ConnectionUsername”*: admin username for the AD
- “*ConnectionPassword”*: admin password for the AD. This will allow Datafari to check the authentification of users by accessing the AD. It is also used to see the permission of the user in the AD so that for an example we can allow some files to be viewed in FileShare and some others not by checking the ACL permissions.

When the configuration is done and you click save, the server will test if the configuration is correct by attempting to connect to the AD. If the attempt is successful, then the modifications are saved and the corresponding files are changed. If the attempt fails, an error message is displayed below the Save button :



Connection URL: ldap://52.16.74.128:389

Connection Username: reda@corp.francelabs.com

Connection Password: fakepassword

UserBase: cn=Users,dc=corp,dc=francelabs,dc=com

✖ Fail to connect to LDAP with the setting given

Once the AD is correctly configured, a user will be able to connect with its AD identifiers without suffix. For instance, assuming there is an “admin” user and the domain is corp.francelabs.com, he can connect to Datafari by giving only admin as username instead of admin@corp.francelabs.com.

Solr configuration customization

> Use the files presented below to add whatever you want to the existing Solr configuration of Datafari. They will never be erased by an update in Datafari 2.x if you perform a modification, plus they are automatically included in the default conf. You can save them in order to backup your custom conf and easily port them on other Datafari installations.
>
> Keep in mind that if you chose not to use these files and to directly modify the standard solrconfig.xml file and/or schema.xml file, you will have troubles saving and restoring your modifications on a new Datafari installation or an update process.

In order to have a smooth update behavior for our users, the Solr configuration files of Datafari are configured by default to include custom files that represent specific parts of Solr. Here is the custom files list and their purpose:

- Location = *{Datafari_home}/solr/solr_home/FileShare/conf/customs_schema/*
    - **custom_fields.incl** - Used to add new fields to the FileShare Solr Core. Structure of the file:

```
<fields>
 <field name="my_field" indexed="true" stored="true"
type="string" multiValued="true"/>
 <field name="my_field2" indexed="true" stored="true"
type="string" multiValued="true"/>
 ....
</fields>
```

- **custom_copyFields.incl** - Used to add custom copy fields to the FileShare Solr Core. Structure of the file:

```
<copyField source="my_field" dest="my_copy" />
<copyField source="my_field2" dest="my_copy2" />
...
```

- **custom_dynamicFields.incl** - Used to add custom dynamic fields to the FileShare Solr Core. Structure of the file:

```
<dynamicField name="my_dynamic_field*" type="string"
multiValued="true"/>
<dynamicField name="my_second_dynamic_field*" type="string"
multiValued="true"/>
...
```

- **custom_fieldTypes.incl** - Used to add custom field types to the FileShare Solr Core. Structure of the file:

```
<fieldType name="myType" class="solr.TextField"
positionIncrementGap="100">
 <analyzer>
  <tokenizer class="solr.KeywordTokenizerFactory" />
  <filter class="solr.LowerCaseFilterFactory" />
 </analyzer>
</fieldType>

<fieldType name="myType2" class="solr.TextField"
positionIncrementGap="100">
 <analyzer>
  <tokenizer class="solr.KeywordTokenizerFactory" />
  <filter class="solr.LowerCaseFilterFactory" />
 </analyzer>
</fieldType>

...
```

- Location = *{Datafari_home}/solr/solr_home/FileShare/conf/customs_solrconfig/*
    - **custom_libs.incl** - Used to add additional libraries that Solr must load. Structure of the file:

      ```
      <lib dir="./lib/mylibs"/>
      <lib dir="./lib/mylibs2"/>
      ...
      ```

    - **custom_request_handlers.incl** - Used to add new request handlers for the FileShare Solr Core. Structure of the file:

      ```
      <requestHandler class="my.requestHandler"
      name="/myRequestHandler">
       <lst name="defaults">
        <str name="param1">true</str>
        <str name="param2">json</str>
       </lst>
      </requestHandler>

      <requestHandler class="my.requestHandler2"
      name="/myRequestHandler2">
       <lst name="defaults">
        <str name="param1">true</str>
        <str name="param2">json</str>
       </lst>
      </requestHandler>

      ...
      ```

    - **custom_search_components.incl** - Used to add new search components for the FileShare Solr Core. Structure of the file:

```
<searchComponent class="my.searchComponent"
name="mySearchComponent">
 <lst name="component">
  <str name="name">default</str>
  <str name="field">sample</str>
 </lst>
</searchComponent>

<searchComponent class="my.searchComponent2"
name="mySearchComponent2">
 <lst name="component">
  <str name="name">default</str>
  <str name="field">sample</str>
 </lst>
</searchComponent>

...
```

- **custom_search_handler.incl** - Used to replace the default Datafari search handler with a custom one. **This file must contain only one search handler which respect the following structure:**

```
<requestHandler class="solr.SearchHandler" name="/select">
 ....
</requestHandler>
```

When this file is filled with a custom search handler, you MUST comment the original one in the *{Datafari_home}/solr/solr_home/FileShare/conf/*solrconfig.xml file. Otherwise you will have troubles with Solr and Datafari !

- **custom_update_processors.incl** - Used to add new update processors for the FileShare Solr Core. Structure of the file:

```
<processor class="my.OwnUpdateProcessorFactory">
 <bool name="enabled">true</bool>
 <str name="param">something</str>
</processor>

<processor class="my.OwnUpdateProcessorFactory2">
 <bool name="enabled">true</bool>
 <str name="param">something</str>
</processor>

...
```

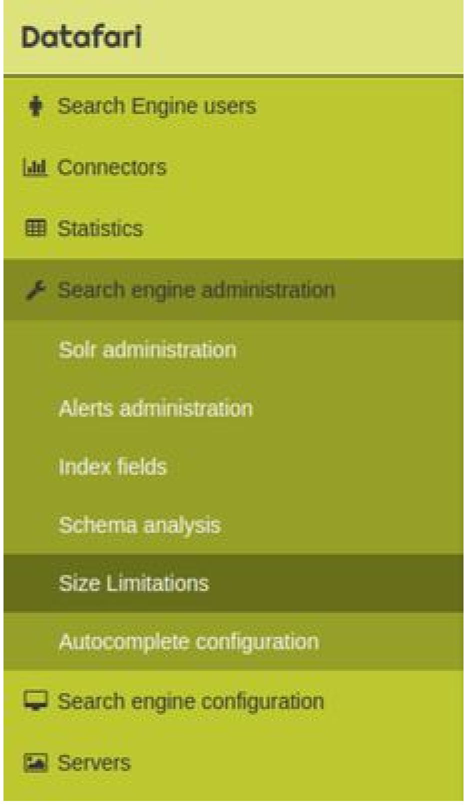
Speeding up search queries

This section lists the possibilities offered by Datafari to lower the response time at query time.

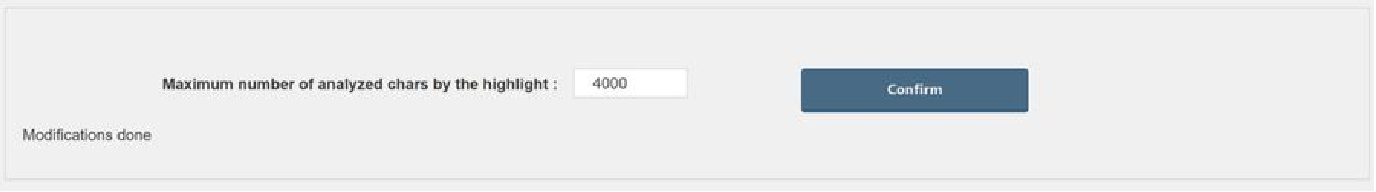
Highlighting configuration

Size limitations allow you to restrict the number of characters analyzed by the highlighter. The higher the number, the better the highlight and the slower the queries.

To set the size limitation, you have to be the search administrator. Go in the admin UI.



Now set the value to a positive integer, then click the “confirm” button :


Maximum number of analyzed chars by the highlight : 4000

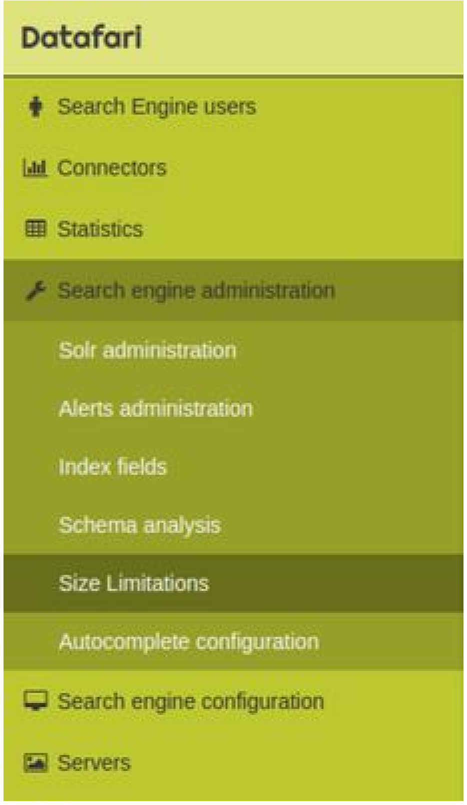
Modifications done

Optimising the index size

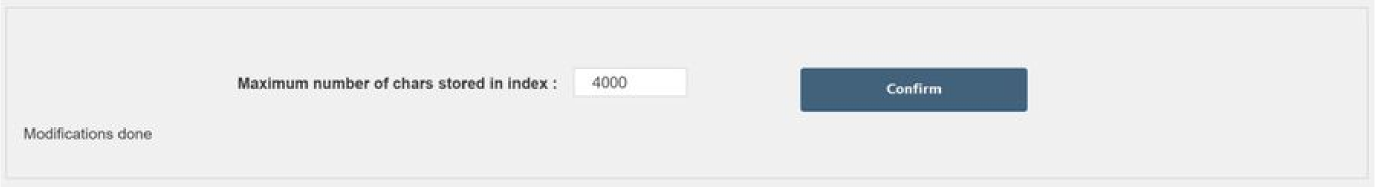
This section lists the parameters and configurations of Datafari that allow you to modify the size of the Datafari index (some modifications may have an impact on the search experience).

Set a size limit for the stored field

You can set a value for the limit of the number of characters stored in the index. This requires a full reindex of the data to be taken into account. For standard scenarios, this value can be equal to the number of analyzed characters for highlighting (Highlighting configuration). Using a low value can save disk space. Limiting stored field won't have an impact on the search results but can lead to uncorrect highlighting.To set the size limitation, you have to be the search administrator. Go in the admin UI:



And set the value in the following tab:

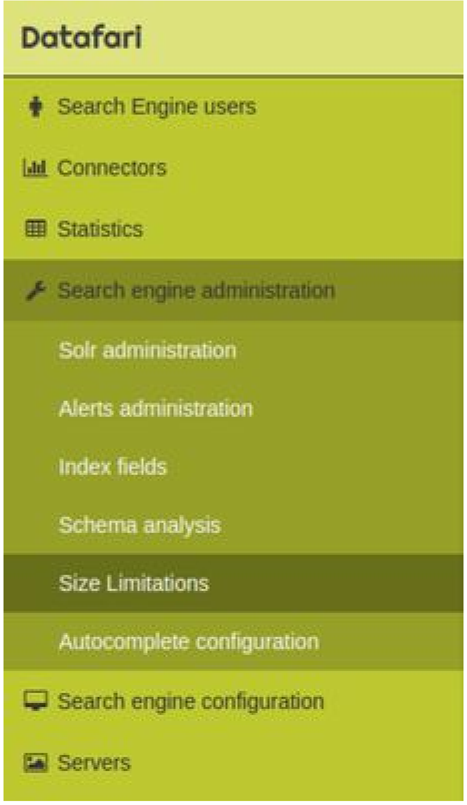


Maximum number of chars stored in index : 4000

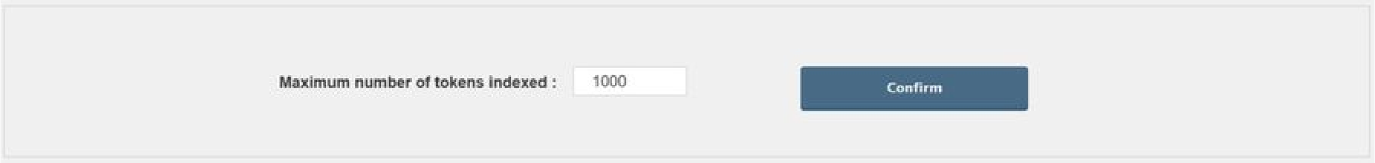
Modifications done

Set a number of token limit for fields:

You can set a value to limit the number of tokens that are indexed in the inverted index for each field of each document. This requires a full reindex of the data to be taken into account. Using a low value can save disk space. Be carefull, limiting the number of indexed tokens can have an impact on search results as inverted index is used to find the documents during a standard search query. To set the size limitation, you have to be the search administrator. Go in the admin UI:



And set the value in the following tab:



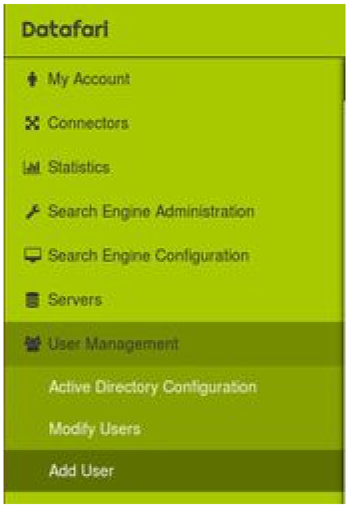
Maximum number of tokens indexed : 1000

You can find more information on the search index in the dedicated section Search Index.

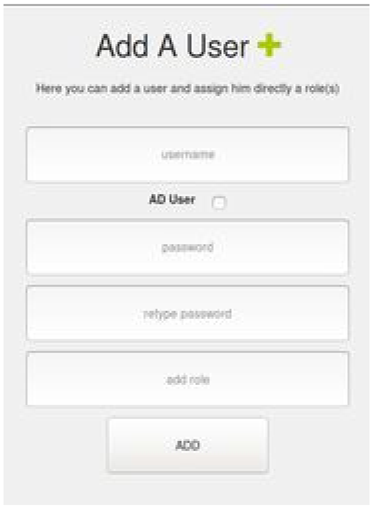
Users Management

A very important thing to know is that if you want to add Datafari roles to an existing Active Directory user, you need to 'Add' it in Datafari as you were adding a fresh new user ! Although, you don't need to add an Active Directory user through this functionality if you don't want to give him some Datafari roles. If you have configured the Active Directory in Datafari, all the AD users will be able to connect to Datafari without further configuration.

As a Search administrator you may want to add/remove or modify a user for Datafari. This can be done through the admin interface of Datafari, the 'User Management' tab allows you to 'Add' or 'Modify' users:


Datafari
My Account
Connectors
Statistics
Search Engine Administration
Search Engine Configuration
Servers
User Management
Active Directory Configuration
Modify Users
Add User

1) Let's start with the 'Add User'. This functionality is meant to add new users to Datafari or to add roles to an existing Active Directory user. If you click on this tab, a form will be displayed with some fields to fill :



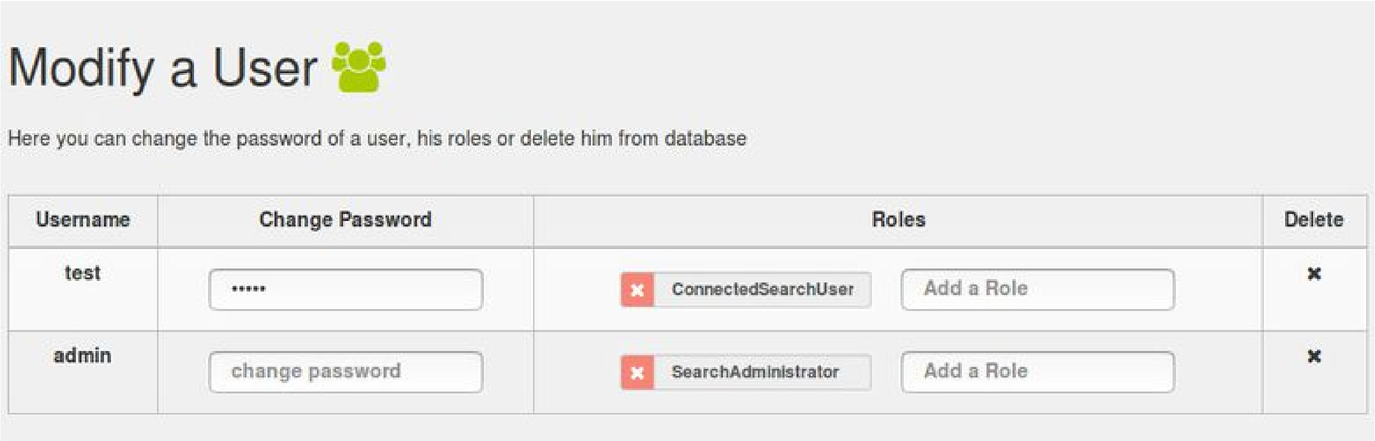
Add A User

Here you can add a user and assign him directly a role(s)

- username: Here you can enter the username you want to create or add to the datafari users in case it is an Active Directory user
- AD User: check it if you are adding an Active Directory user to give him roles. If this option is checked, the password fields will disappear.
- password: the user password (this field is not available if you have checked the 'AD User' option)
- retype password: confirm the user password (this field is not available if you have checked the 'AD User' option)
- add role: here you can add roles to the user. By default, 3 roles are available:
  - SearchAdministrator: this role give a full access to all the features of Datafari
  - SearchExpert: this role give an access to all the SearchExpert features
  - ConnectedSearchUser: this is the basic user role that allows the user to connect to Datafari
  To add a role, type the first characters of the role, a pop-up suggestion will appear, select the suggestion to confirm the wanted role.
  If you want to remove a role, simply click on the red cross associated to the role.

Once you have fulfilled all the fields, you can click on the 'ADD' button. If everything is correct, you will have a green message saying 'User Successfully Saved', otherwise you will have a red message telling you what's wrong.
When adding an Active Directory user, Datafari will use the AD configuration to test if the user exists before adding it. So a 'User Successfully Saved' message indicates that the user has been found in the Active Directory and the selected Datafari roles added to it (in the Datafari system, not as real Active Directory roles)

2) The 'Modify Users' tab shows all the users in Datafari that have roles. All the Active Directory users that don't have any role are not displayed in this view:



Modify a User

Here you can change the password of a user, his roles or delete him from database

Username	Change Password	Roles	Delete
test	•••••	✖ ConnectedSearchUser Add a Role	✖
admin	change password	✖ SearchAdministrator Add a Role	✖

In this view you can:

- Change the password of a user by typing a new password and press the 'Return' key on your keyboard
- Add or remove roles to a user as in the 'Add User' interface. Adding or deleting a role is immediately taken in account.

Update Datafari

By default Datafari does not provide support for updates and updates are not possible between two major version of Datafari (ex : Datafari 1.X to Datafari 2.x). Although, an update between two minor versions of Datafari should work (ex: Datafari 2.0 to Datafari 2.X), but we strongly recommend you to backup your current version before doing it.
To update Datafari, you only need to run the 'dpkg -i [path to the Datafari .deb file]' command. All the files that have changed in the new version will be updated but if you have followed our recommendations concerning the Widgets and CSSs good practices and the Custom Solr configuration, none of your modifications should be affected and neither the Solr index or the data should have changed.

Search experts

This section presents the configuration, management and analytics tools that the search expert can leverage in Datafari.

The search expert is responsible for maintaining the relevance of Datafari, monitoring its usage, and managing its community of users. He ensures that queries with no clicks are not left alone. He checks the vitality of the system in terms of queries. He manages the promolinks. Beyond Datafari, he also ensures that people are aware of the tool, and he trains them if necessary. The expert has access to a range of functionalities from his administration interface: statistics through a graphical dashboard, promolinks management, synonyms and stopwords management, field based boosts, documents elevation, duplicate identifier, OCR activation, user management. We don't detail these functionalities here, they have dedicated sections in the documentation.

Alerts management - Mail Configuration

Search admin: managing the alerts mechanism

Using the search admin role, you first need to configure the alert mechanism so that it uses the smtp server to send emails.

The Alerts Administration UI allows to graphically edit the config file alerts.properties which is stored in InstallationDirectoryDatafari/tomcat/conf/

It needs the following information:

- ***SMTP*** = the name of the SMTP host
- ***Address*** = the sender address (exemple@exemple.com)
- ***User*** = the username (usually it will be the address)
- ***Password*** = the password of the mail address

For Instance :

- smtpHost = smtp.gmail.com;
- from = datafari.test@gmail.com;
- username = datafari.test@gmail.com;
- password = ********;

Once this is configured and saved, Datafari will automatically restart the alerts to take into account your modifications.

The search admin has more access rights than a connected search user. He gets the same type of information, except that he sees it for everyone. The only thing he cannot do is to create alers on behalf of other users.

Analytics

Datafari proposes by default two analytics page:

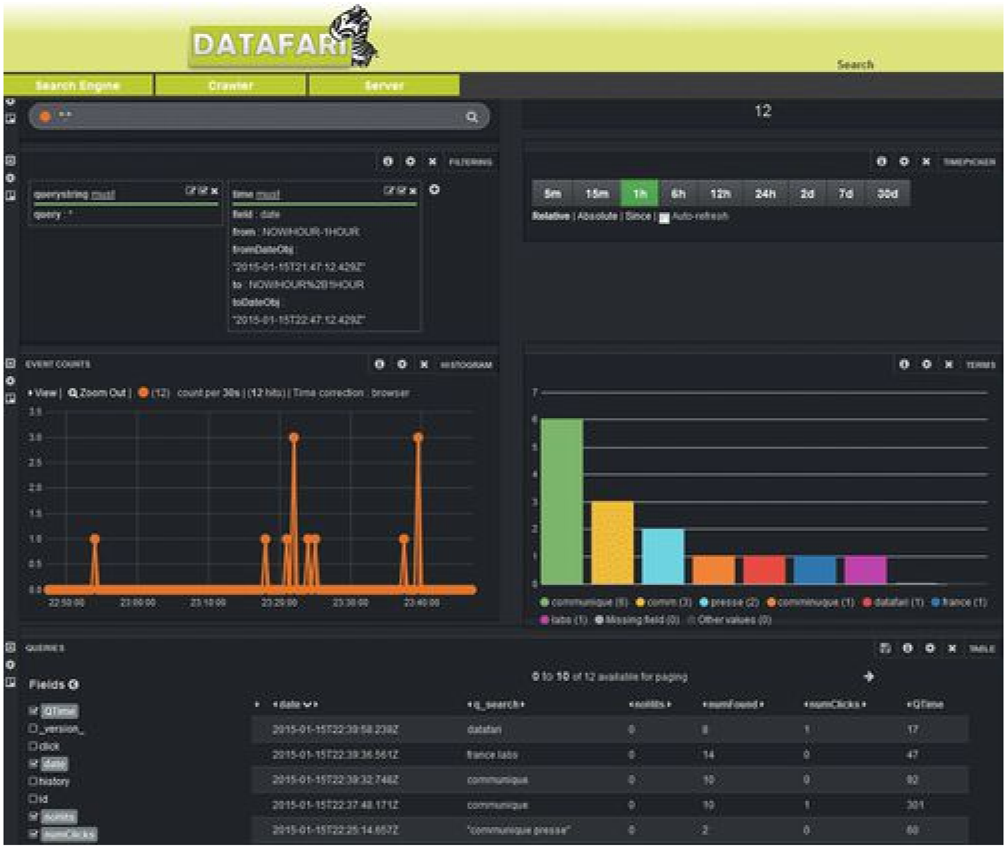
- a "static" analytics UI, focusing on search behavior, taking into account a notion of "navigation" of users when they do their search.
- a "dynamyc" analytics UI, focusing on the search log, allowing the user to create new dashboards, more flexible but with no notion of "navigation" of users.

Dynamic analytics

This page allows for a flexible analysis of search related information. It consumes a datafari solr core dedicated to Solr logs. It leverages the Banana analytics framework, provided by the Lucidworks team, and which is a port of the Kibana analytics framework. It provides a graphical tool to perform analytics on search logs. For a deeper explanation of Banana, refer to its reference documentation.

We detail here the panels we have pre-configured for Datafari v0.9

- The first panel on the upper left, which shows a search bar, is preconfigured with the ***:*** value. This means that the remainder of the widgets refer to the empty query. As for Google, an empty query means that all documents are returned. Since we use a search index that stores all the queries done on Datafari, this empty query returns all the queries that users have done on this Datafari.
- The second panel, below the first panel on the left, is a technical widget. Please refer to the Banana documentation for details about it.
- The next panel, on the right of the second panel, allows to configure the time window used by some of the widgets. It goes from 5 min to 30d. For instance, it impacts the Event Counts widgets, which presents a timeline.
- Event counts: this chart presents a timeline of the number of search queries that have been triggered on the server. One count corresponds to one search query. If you left the empty query on the first panel, you get the total number of queries done on the server. If you have entered another value, you will only get on the timeline the queries that contain this term.
- The next panel, on the right of "Event counts", presents with a colored histogram all the terms that were co-occuring with the term of the first panel in the search queries. If you left the empty query in the first panel, this histogram gives you a ranking of the most popular terms. If you put a term in the first panel, you will get a ranking of the most popular terms colocated with this term. This widget is dynamic. If you click on one bar, which means one term, you will get an new histogram that shows the most popular terms that are colocated with both the term of the first panel, and the term of the colored bar you've clicked on.
- The last panel at the bottom, is a chart presenting the content of the search queries index. Each line thus represents a user search query. You can play with the left column, which lets you pick which fields of the index you want to see on the results list.



:

querystring must
query : *

time must
field : date
from : NOW/HOUR-1HOUR
fromDateObj :
"2015-01-15T21:47:12.429Z"
to : NOW/HOUR%2B1HOUR
toDateObj :
"2015-01-15T22:47:12.429Z"

☑ QTime
☐ _version_
☐ click
☑ date
☐ history
☐ id
☑ nohits
☑ numClicks

date	q_search	nohits	numFound	numClicks	QTime
2015-01-15T22:39:58.239Z	datafari	0	8	1	17
2015-01-15T22:39:36.561Z	france labs	0	14	0	47
2015-01-15T22:39:32.748Z	communique	0	10	0	82
2015-01-15T22:37:48.171Z	communique	0	10	1	301
2015-01-15T22:25:14.657Z	"communique presse"	0	2	0	60

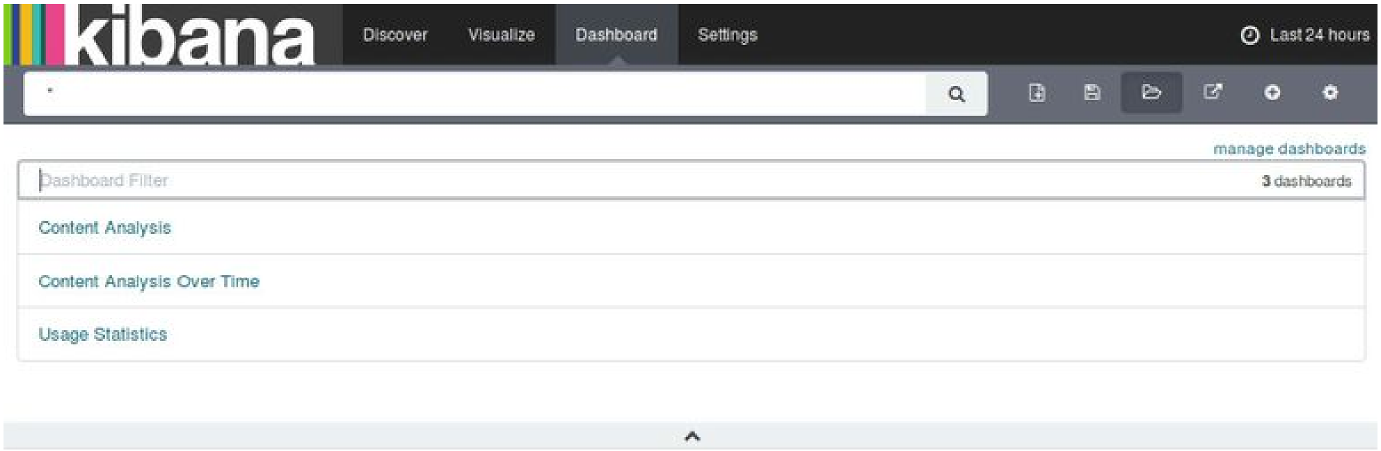
ELK analytics

The 2.2 release of Datafari introduces the ELK layout (Elasticsearch, Logstash, Kibana) to provide a full customizable analytic UI through Kibana.

By default, three dashboards are provided with Datafari:

- Content Analysis
- Content Analysis Over Time
- Usage Statistics

To open or switch between the available dashboards, click on the "Dashboard" tab, then click on the folder button next to the filter bar and select your dashboard in the list that appears

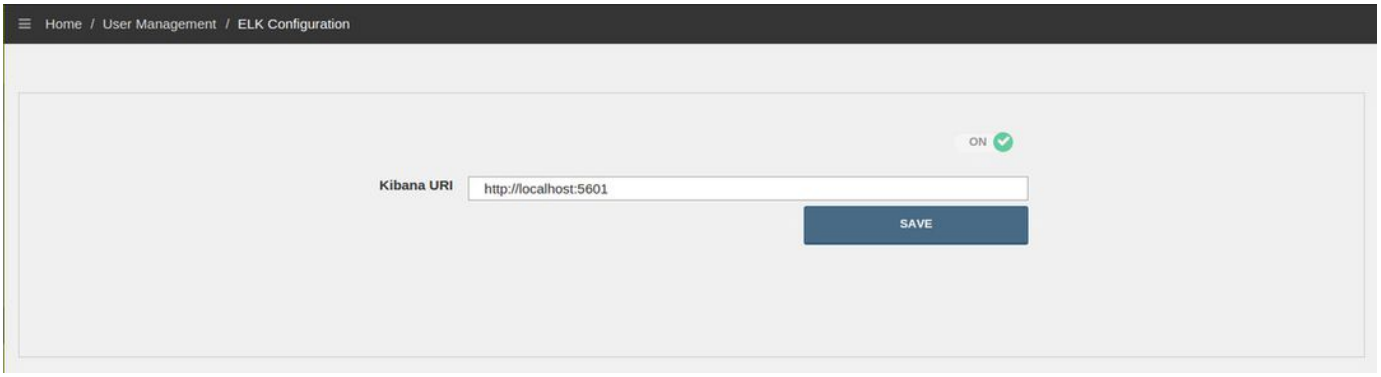


Configure ELK

This feature is only available from the v2.2 of Datafari !

By default, aside from Kibana, Elasticsearch and Logstash are automatically configured by Datafari on the first start, to fit with your installation directory and be ready to run.
If for some reasons (like an architecture decision), you move one or more of these components to a different place than the installation directory of Datafari, you must modify the ELK environment parameters located in [DATAFARI_HOME]/elk/scripts/set-elk-env.sh and maybe also the [DATAFARI_HOME]/elk/scripts/start-elk.sh and [DATAFARI_HOME]/elk/scripts/stop-elk.sh scripts.

The first time you run Datafari, go to the admin UI and to the 'ELK Configuration' page (Statistics/ELK Configuration). You will find this UI:

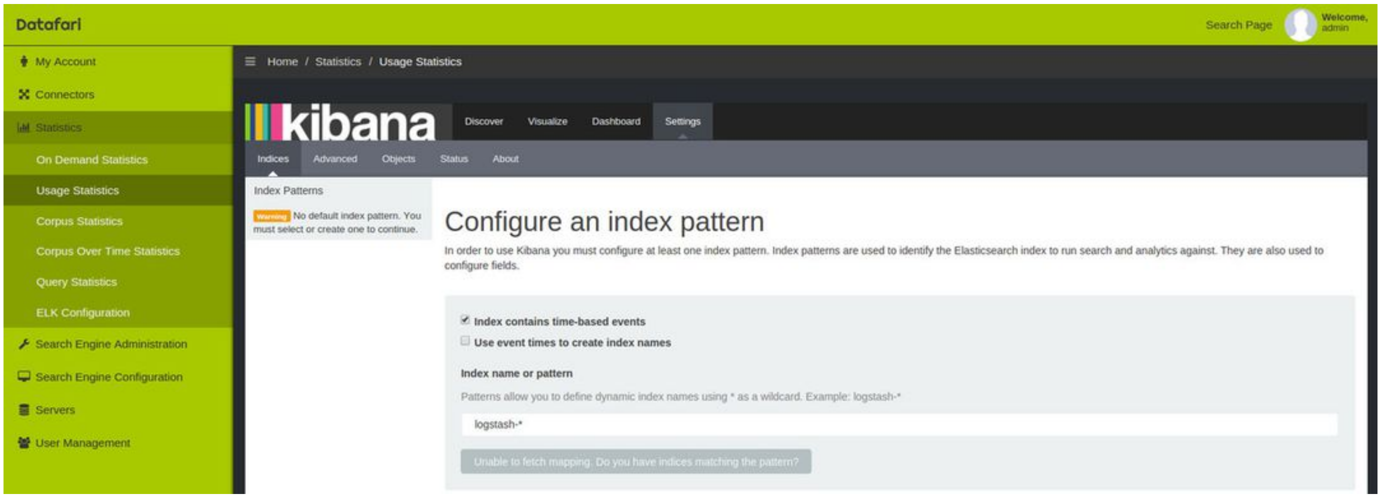


Kibana URI: http://localhost:5601

Let us go through the 2 options:

- **"ON/OFF" button :** Should ELK start when Datafari is running ? (Note that if you swich it ON, it will immediately start ELK)
- **Kibana URI :** The URI to reach Kibana. By default it is http://localhost:5601

Once you have started ELK (be aware that the average time for ELK to be 100% operational is approx. 15 seconds on a "standard" PC), click on any statistic tab ('Usage Statistics', 'Corpus Statistics' or 'Corpus Over Time Statistics'). You should see the following page :



Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names

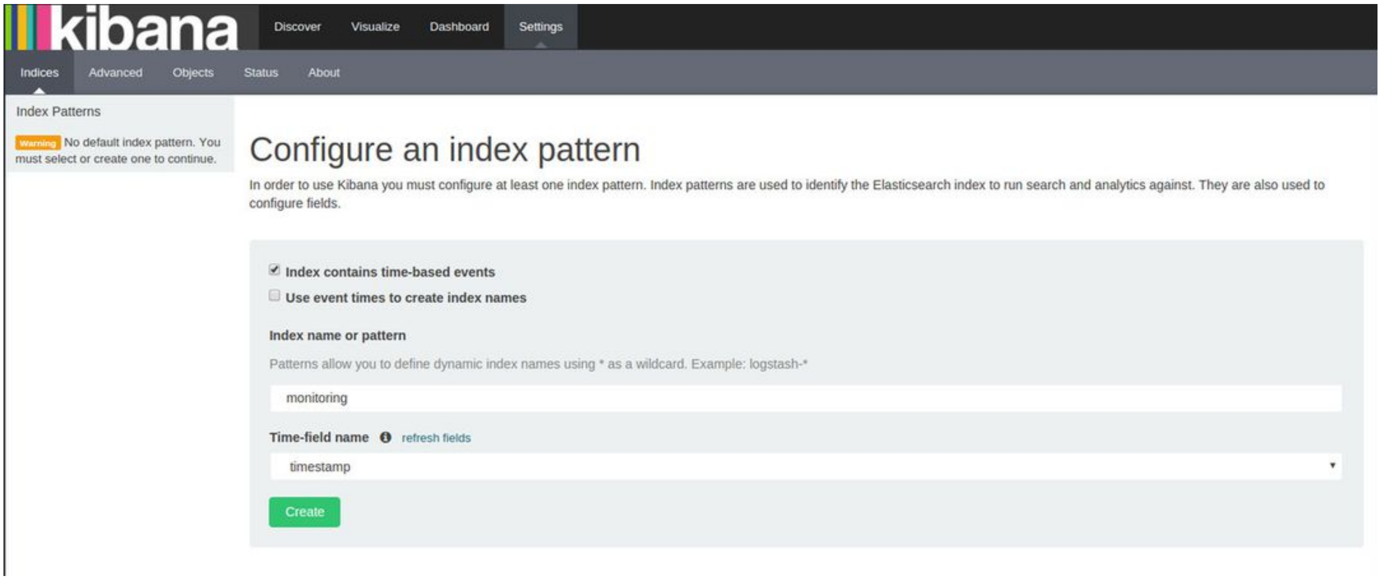
Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

Unable to fetch mapping. Do you have indices matching the pattern?

You need to configure the Elasticsearch indexes which will be used for the dashboards. To do so, type 'monitoring' as Index name and click 'Create':



Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

□ Use event times to create index names

Index name or pattern

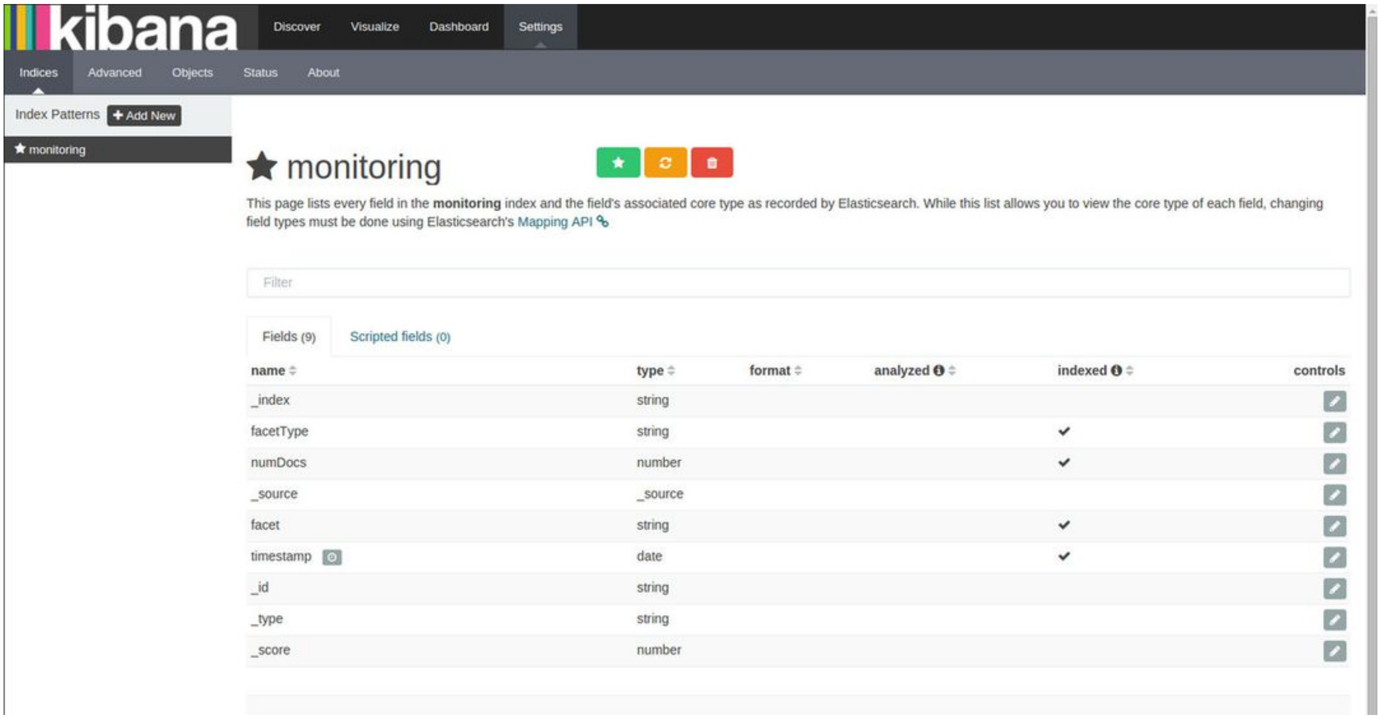
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

monitoring

Time-field name

timestamp

The index should have been found by Kibana and you should see the following page:

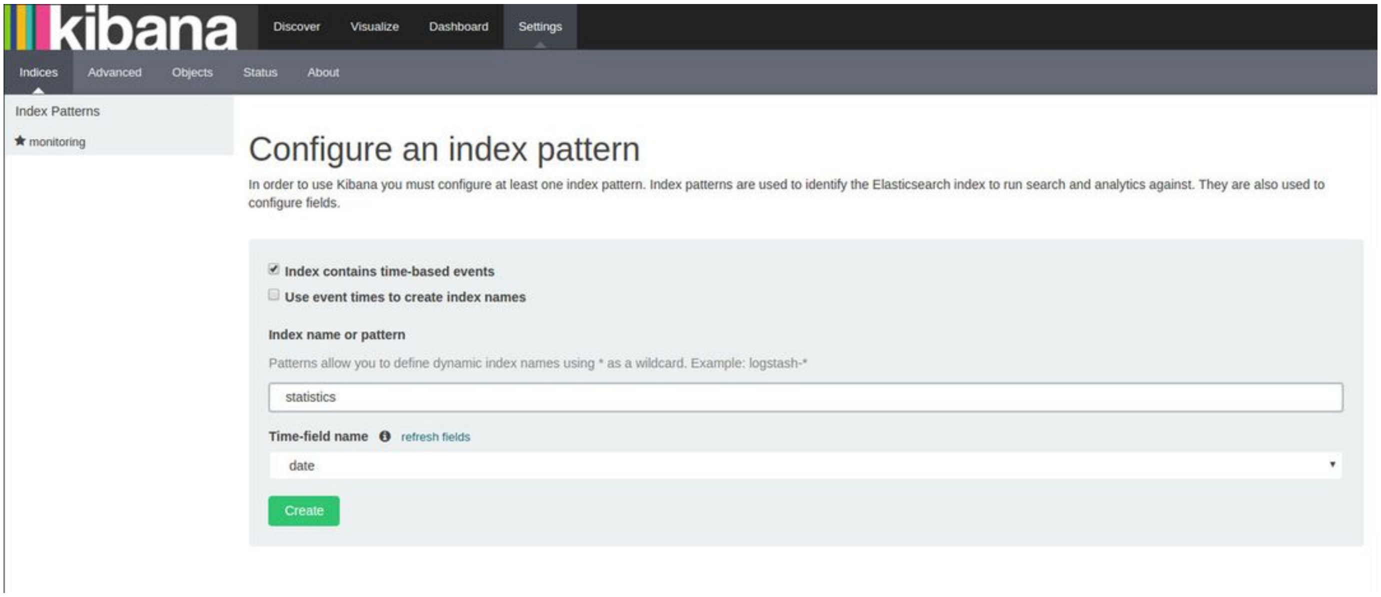


★ monitoring

This page lists every field in the **monitoring** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

name	type	format	analyzed	indexed	controls
_index	string				
facetType	string			✓	
numDocs	number			✓	
_source	_source				
facet	string			✓	
timestamp	date			✓	
_id	string				
_type	string				
_score	number				

Now click on the "Add New" button, and repeat the same steps with the index name 'statistics':



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ **Index contains time-based events**

☐ **Use event times to create index names**

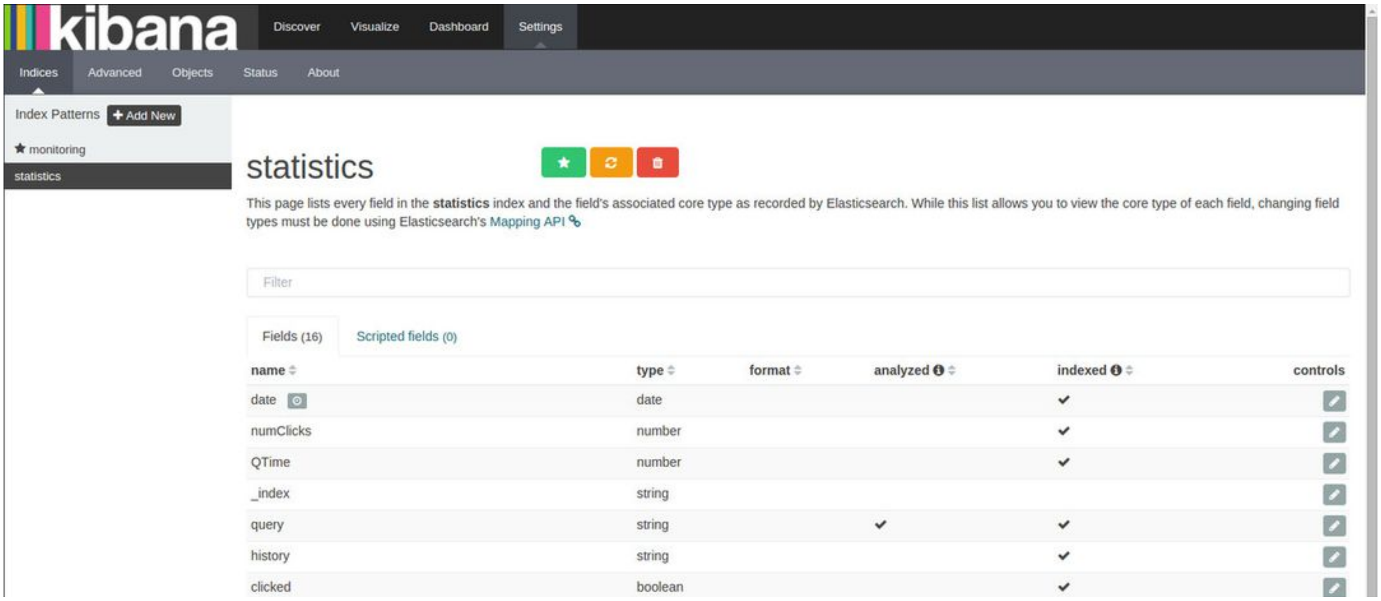
Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

statistics

Time-field name refresh fields

date

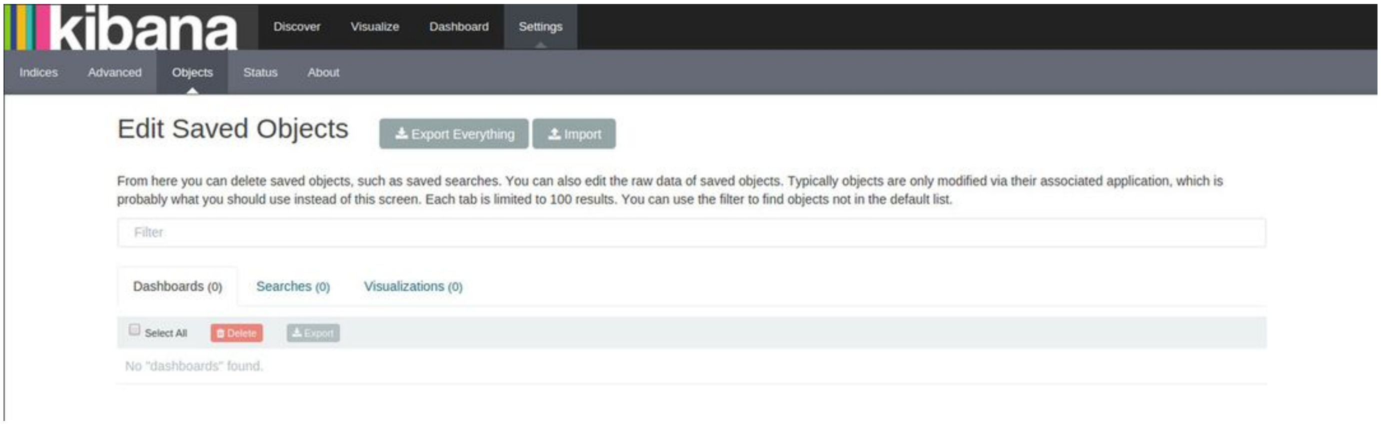


statistics

This page lists every field in the **statistics** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

name	type	format	analyzed	indexed	controls
date	date			✓	
numClicks	number			✓	
QTime	number			✓	
_index	string				
query	string		✓	✓	
history	string			✓	
clicked	boolean			✓	

Now click on the "Objects" tab and then on the "Import" button:

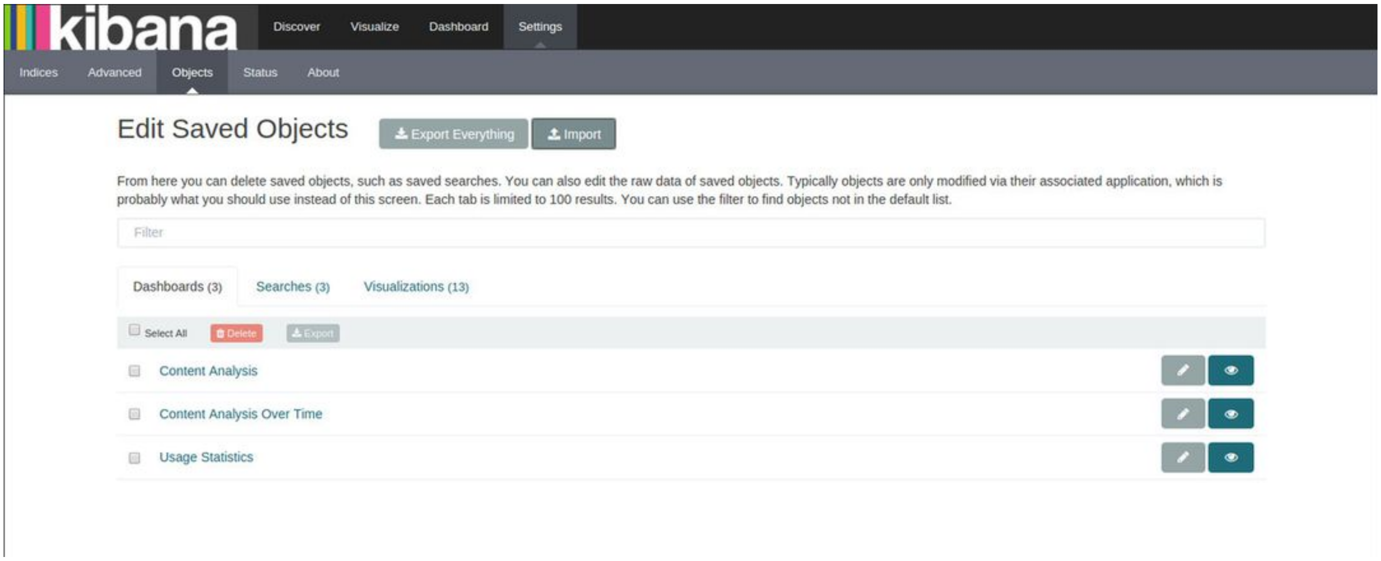


Edit Saved Objects

From here you can delete saved objects, such as saved searches. You can also edit the raw data of saved objects. Typically objects are only modified via their associated application, which is probably what you should use instead of this screen. Each tab is limited to 100 results. You can use the filter to find objects not in the default list.

No "dashboards" found.

And import the [DATAFARI_HOME]/elk/save/datafari-kibana.json file. If you don't have access to the folders local to the server, you can retrieve it from our github repository https://github.com/francelabs/datafari/raw/zk/datafari-elk/save/datafari-kibana.json :

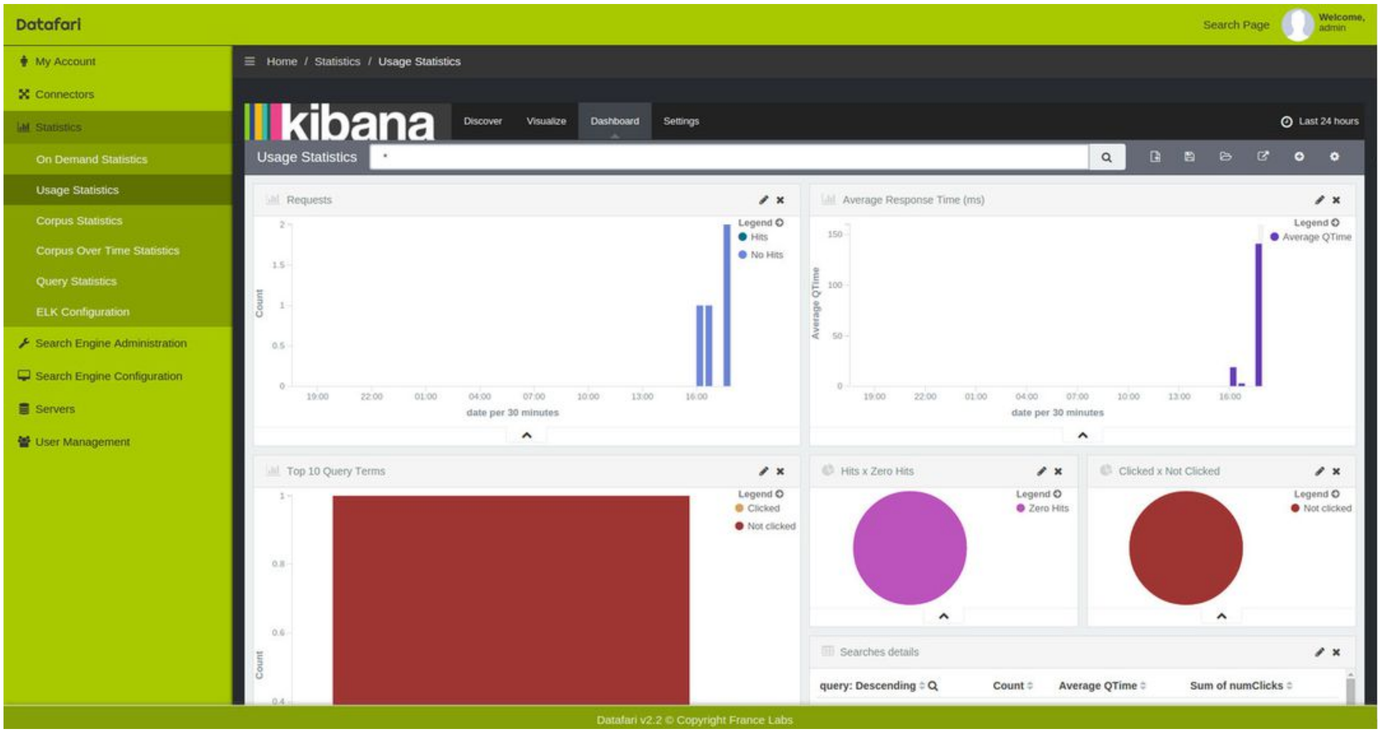


Edit Saved Objects

From here you can delete saved objects, such as saved searches. You can also edit the raw data of saved objects. Typically objects are only modified via their associated application, which is probably what you should use instead of this screen. Each tab is limited to 100 results. You can use the filter to find objects not in the default list.

- Content Analysis
- Content Analysis Over Time
- Usage Statistics

Check that you have exactly the same things as the screenshot below: 3 Dashboards, 3 Searches and 13 Visualizations. If it is the case congratulations ! Your ELK is now fully configured and if you click, in the Datafari admin UI, on any statistic tab ('Usage Statistics', 'Corpus Statistics' or 'Corpus Over Time Statistics'), you will now see the corresponding dashboard:

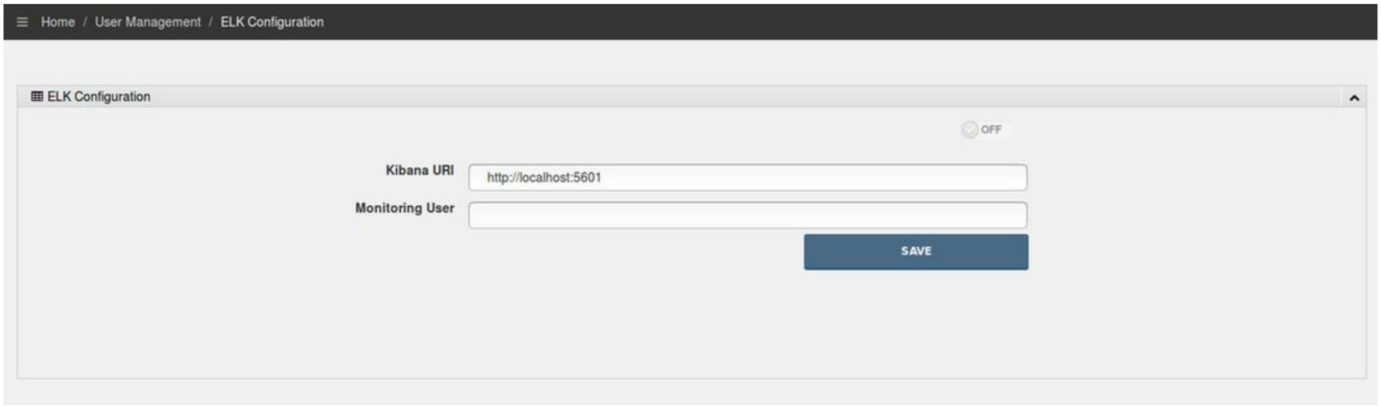


Configure ELK (Datafari 3.1 and above)

This feature is only available from the v3.1 of Datafari !

By default, aside from Kibana, Elasticsearch and Logstash are automatically configured by Datafari on the first start, to fit with your installation directory and be ready to run.
If for some reasons (like an architecture decision), you move one or more of these components to a different place than the installation directory of Datafari, you must modify the ELK environment parameters located in [DATAFARI_HOME]/elk/scripts/set-elk-env.sh and maybe also the [DATAFARI_HOME]/elk/scripts/start-elk.sh and [DATAFARI_HOME]/elk/scripts/stop-elk.sh scripts.

The first time you run Datafari, go to the admin UI and to the 'ELK Configuration' page (Statistics/ELK Configuration). You will find this UI:



OFF

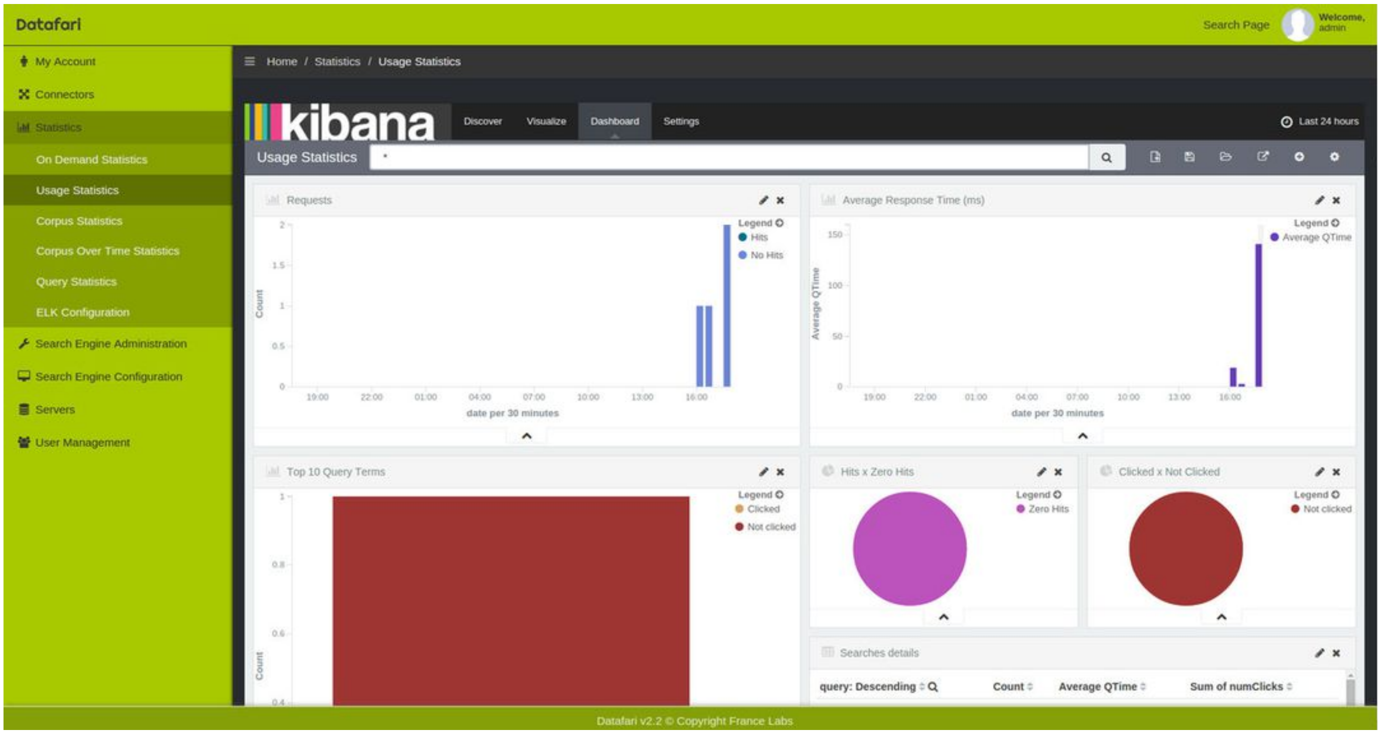
Kibana URI: http://localhost:5601

Monitoring User:

Let us go through the 3 options:

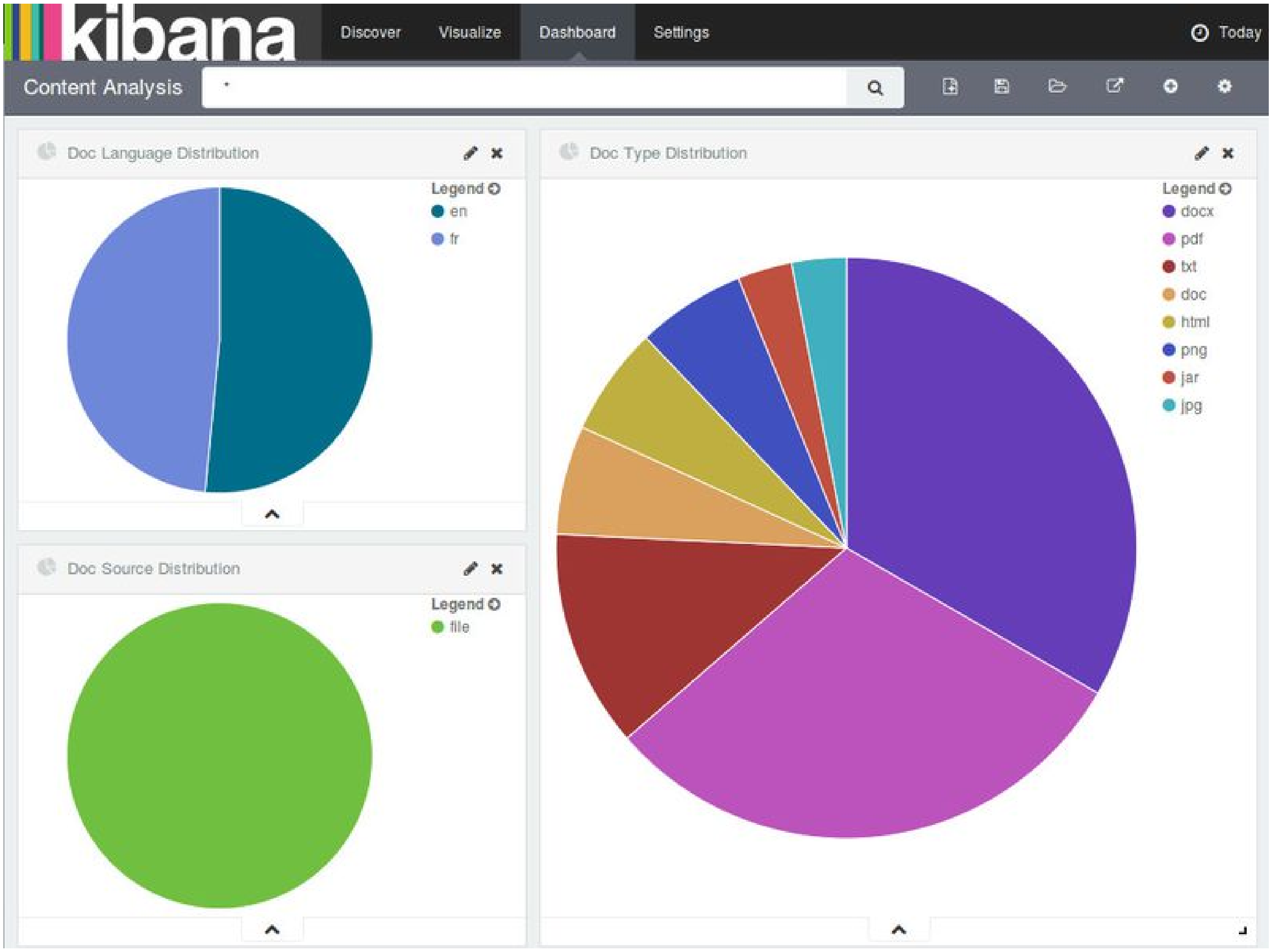
- **"ON/OFF" button :** Should ELK start when Datafari is running ? (Note that if you swich it ON, it will immediately start ELK)
- **Kibana URI :** The URI to reach Kibana. By default it is http://localhost:5601
- **Monitoring User :** This parameter should only be filled if you use ACLs for search. In that case, enter here the user used to crawl the files. Otherwise, ELK won't be able to generate statistics on the corpus.

Once you have started ELK (be aware that the average time for ELK to be 100% operational is approx. 15 seconds on a "standard" PC), click on any statistic tab ('Usage Statistics', 'Corpus Statistics' or 'Corpus Over Time Statistics'), you will now see the corresponding dashboard:



Content Analysis

This dashboard is based on the Core Monitoring logs generated by Datafari, which are inserted by Logstash in Elasticsearch under the "monitoring/logs" index.



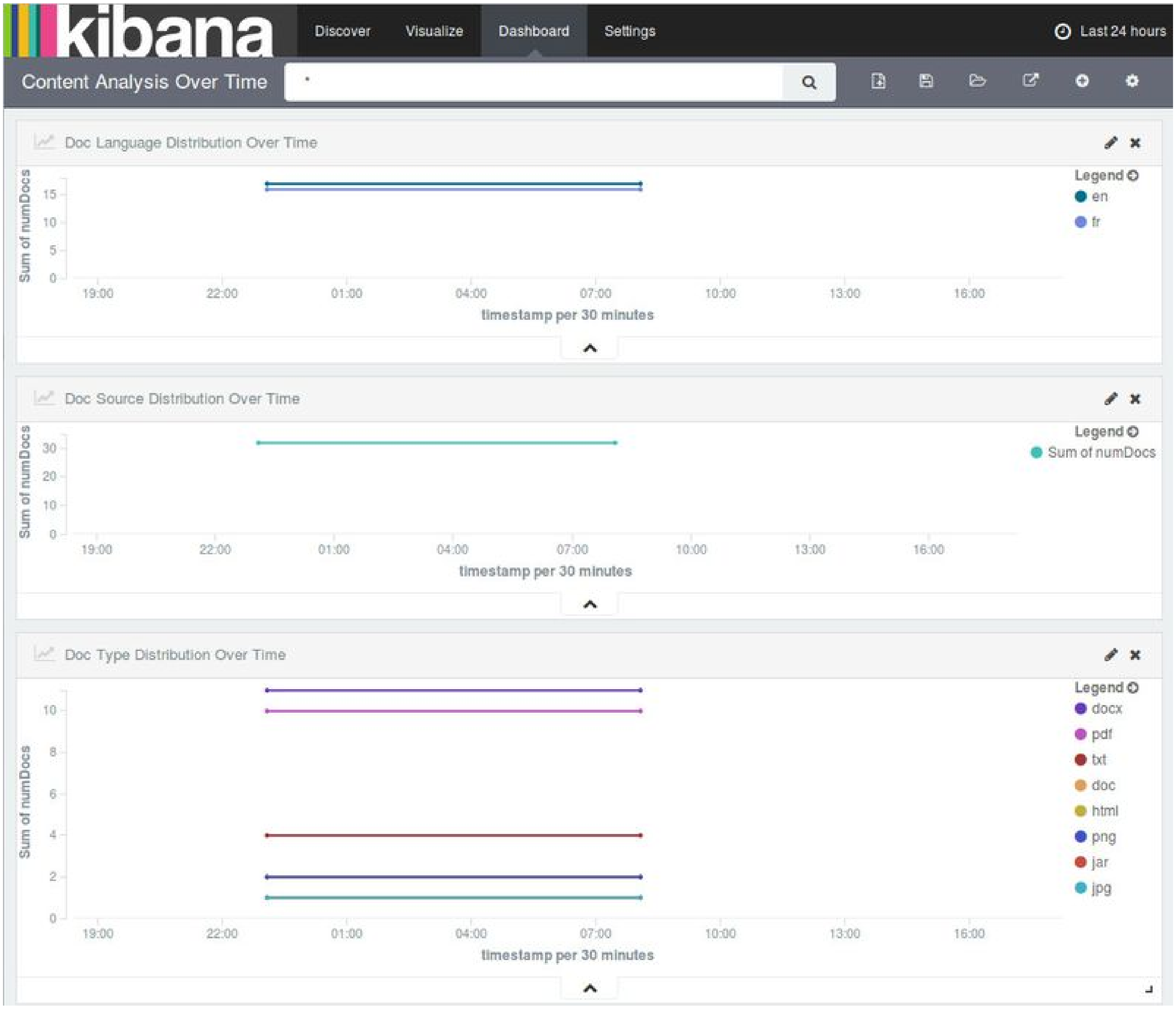
It shows the documents distribution in the core.
There is one pie chart per distribution type, and by default the types are doc language, doc type and doc source. As you may have understood, each pie is based on a facet : language, source and extension. You can create more charts, based on other facets by learning how Kibana works, and how our ELK works as well.

The important thing on this dashboard is the selected time frame. You can see it at the top right of the Kibana, it is set to "Today" by default. There is a really good reason why it is set to this value. Kibana works with time events and we decided by default to have one time event by day. It means that we have only one Elasticsearch document by facet value per day. Thus, those "daily documents" are updated each hour by Datafari when a monitoring iteration happens, and we keep the data displayed in Kibana uptodate. This also means that **if you change the time frame of this dashboard to another value than "Today", the data used by the charts will be wrong**. Indeed, the time frame of this dashboard must be set to one time event (so the current day if the time event is daily, the current hour if the time event is hourly, and the current minute if the time event is minutely)

For the moment, the Kibana time event is not configurable from the admin UI of Datafari but only by the code (see ELK for a better understanding).

Content Analysis Over Time

This dashboard is based on the Core Monitoring logs generated by Datafari, which are inserted by Logstash in Elasticsearch under the "monitoring/logs" index.

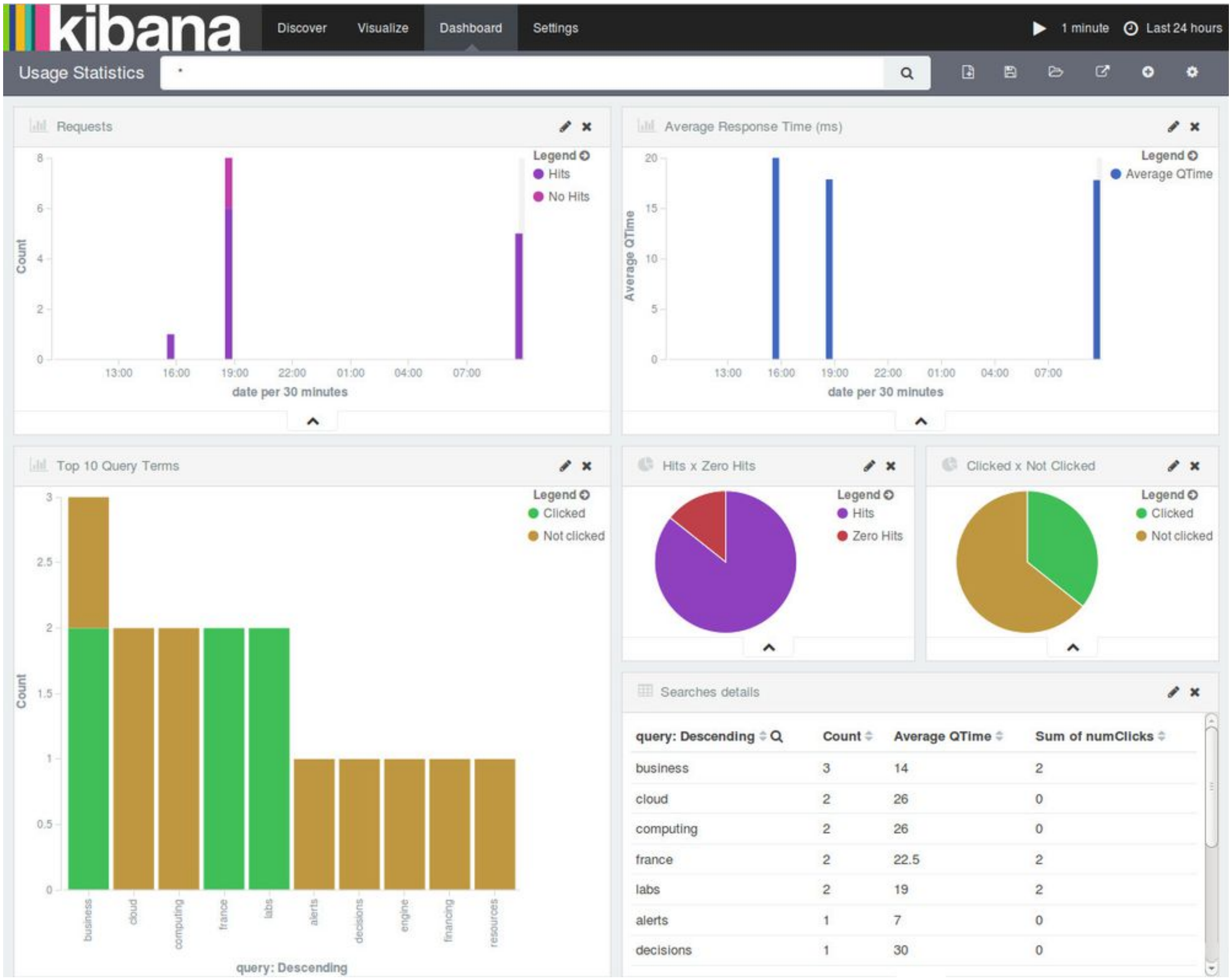


It shows, like the Content Analysis dashboard, the documents distribution in the core, but over the time.
There is one line chart by distribution type which are by default doc language, doc type and doc source. As you may have understood, each chart is based on a facet : language, source and extension.
You can create more charts, based on other facets by learning how Kibana works, and how our ELK works as well.

Kibana works with time events and we decided by default to have one time event by day. It means that we have only one Elasticsearch document by facet value by day. Thus, those "daily documents" are updated each hour by Datafari when a monitoring iteration happens, and we keep the data to display in Kibana up to date. This also means that **you will only see one dot per day for each value**. If you want more values per day, you need to change the Kibana time event which is yet only configurable in the code of Datafari.

Usage Statistics

This dashboard is based on the Statistics logs generated by Datafari, which are inserted by Logstash in Elasticsearch under the "statistic/logs" index.



query: Descending	Count	Average QTime	Sum of numClicks
business	3	14	2
cloud	2	26	0
computing	2	26	0
france	2	22.5	2
labs	2	19	2
alerts	1	7	0
decisions	1	30	0

It shows, various informations about the behaviour of users on Datafari.

In the default dashboard, six visualizations are available:

- **Requests**: represents the number of requests over the time, with, in addition, the distribution between the requests that had at least one hit and those that did not have any hit
- **Average Response Time (ms)**: shows the requests average response time in milliseconds over the time
- **Top 10 Query Terms**: it is the top 10 of the most searched terms in Datafari. For each term the chart shows the total number of time it was queried composed by the number of queries where users have clicked on a result and the number of queries where they did not.
- **Hits x Zero Hits**: shows the distribution between requests that had at least one hit and those that did not have any
- **Clicked x Not Clicked**: shows the distribution between queries where users have clicked at least on one result and those where they did not.
- **Searches details**: it is a data table showing all terms queried with some infos

The data used to build this dashboard are based on data corresponding to the selected time frame, which is 24 hours by default.

You can create more charts, based on other data, by learning how Kibana works, and how our ELK layout works as well.

Search navigation analytics

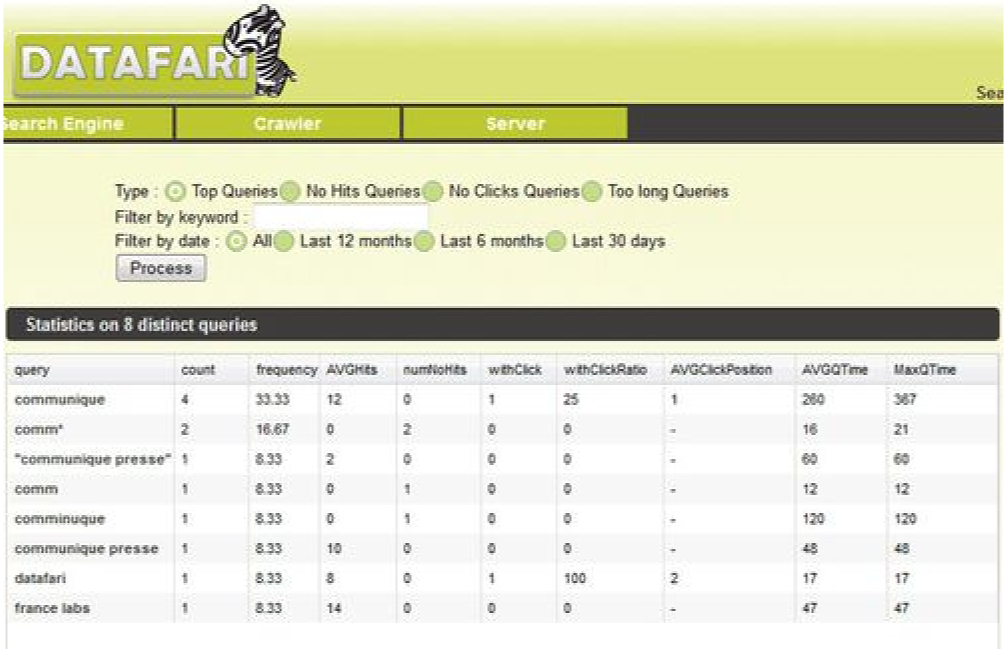
This static page focuses on search analysis, with a notion of "navigation" from users.

Going through the functionnalites, you can first decide which type of analysis you want to see, but selecting it with the buttons on the first line titled "Type":

- **Top Queries**: it will display in the chart below only the most frequent queries. This allows you to identify trends in terms of what users are searching for.
- **No Hits Queries**: it will display in the chart below only the most frequent queries where the search engine found no corresponding

documents in its index. This may be a sign that you may have missed a relevant data source repository (and then you should add it in ManifoldCF), or maybe some documents have not been taken into account at the indexing phase, in which case you should contact your system administrator to investigate the issue.

- **No Clicks Queries**: it will display in the chart below only the most frequent queries with results, but where users did not click on any result from the list. This may be a sign that users are not satisfied with the documents displayed. If you have an idea of which document should be displayed for a given query, you can use the admin functionnality that allows to promote documents based on identified queries.
- **Too long Queries**: it will display in the chart below the queries sorted by their execution time, from the slowest to the fastest. This is can be used to evaluate the performance of the engine. But it can also be used as hint for the system administrator if some specific queries unexpectedly consume a very long time to be executed.



Type : ◉ Top Queries ○ No Hits Queries ○ No Clicks Queries ○ Too long Queries
Filter by keyword :
Filter by date : ◉ All ○ Last 12 months ○ Last 6 months ○ Last 30 days

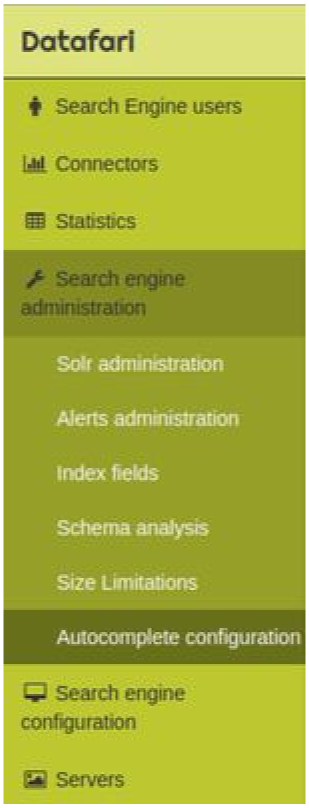
Statistics on 8 distinct queries

query	count	frequency	AVGHits	numNoHits	withClick	withClickRatio	AVGClickPosition	AVGQTime	MaxQTime
communique	4	33.33	12	0	1	25	1	260	367
comm*	2	16.67	0	2	0	0	-	16	21
"communique presse"	1	8.33	2	0	0	0	-	60	60
comm	1	8.33	0	1	0	0	-	12	12
comminuque	1	8.33	0	1	0	0	-	120	120
communique presse	1	8.33	10	0	0	0	-	48	48
datafari	1	8.33	8	0	1	100	2	17	17
france labs	1	8.33	14	0	0	0	-	47	47

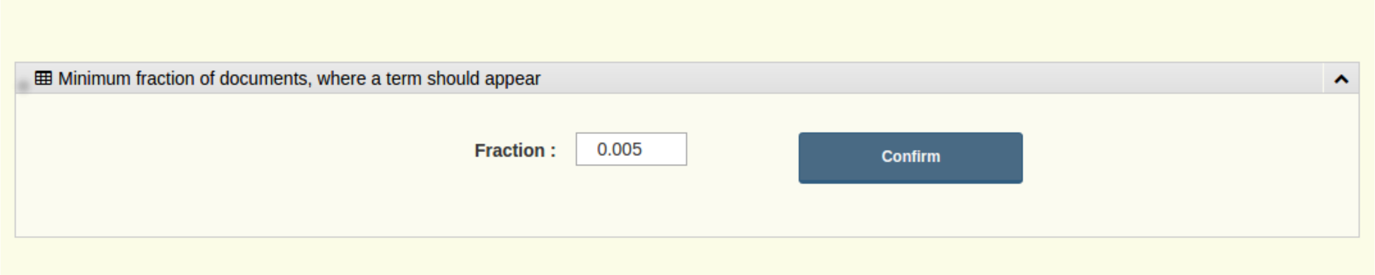
Autocomplete configuration

Auto-complete configuration allows you to set the proportion of documents in the index where a term should appear to be suggested in the auto-complete. The higher this proportion, the less terms are going to be suggested.

In order to configure auto-complete, you have to be the search administrator, and access to the UI.



Now you can set the value to something between 0 and 1, then click the “confirm” button.



Fraction : 0.005

Core reload required
Keep in mind that Datafari takes account of the changes only after a reload of the core or a complete restart.

Connectors configuration

Datafari relies on Apache ManifoldCF for the connection to data sources and retrieval of the data. It has a standalone documentation, which you can refer to in order to properly setup your connectors: Apache ManifoldCF official documentation.

Crawling

The crawling phase focuses on contacting the remote data repositories, retrieving the data, pre-processing it, and handing it over to the indexing engine. Using Apache ManifoldCF, it handles full or partial data retrieval, can connect to many different types of data repositories, and handles the security phase of indexing and searching. The full documentation for the crawl administrator is naturally located on the apache manifoldCF website, more precisely on the ManifoldCF 1.5.1 documentation.

⚠ **The date of LibreOffice documents is not crawled correctly.**
Date metadata is not extracted from plain text documents.

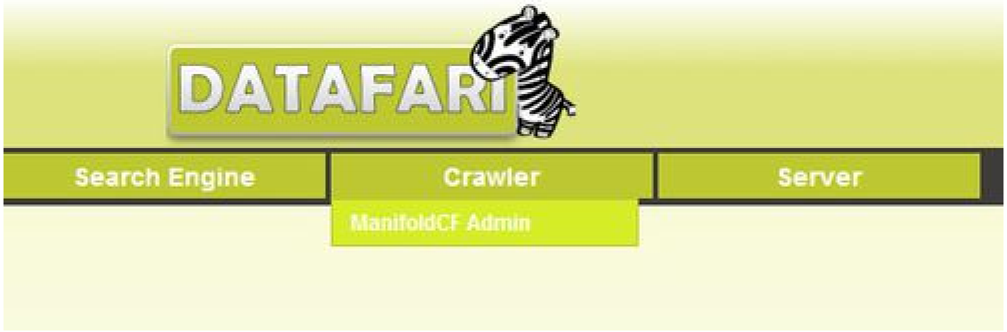
We detail here how to kick off quickly in Datafari with a crawling of a local file folder.

Local and shared files crawling

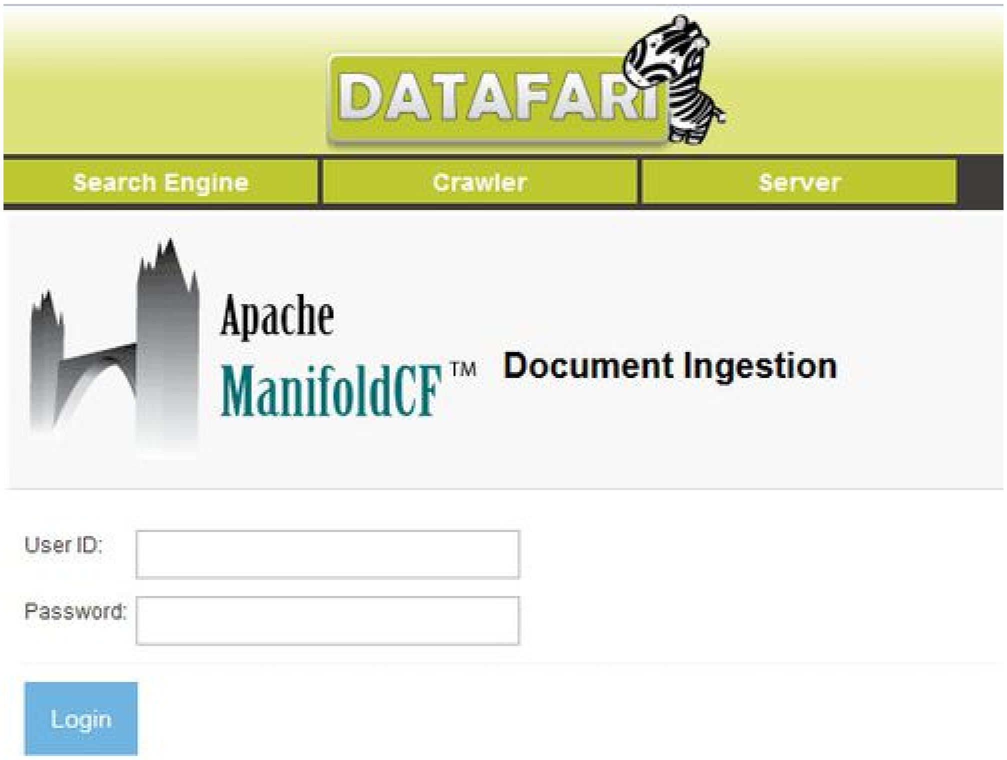
⚠ **The local file crawler is a DEMO connector that shouldn't be used for production environments.**

A good alternative to the local file crawler is the "Windows share" (a.k.a. JCIFS connector), that may be used to crawl files in a file shared directory (it works under Linux as well 🙂). You can check its official documentation on the Apache ManifoldCF website.
Documentation on how to setup a samba share on Linux here.
Look at Add the JCIFS Connector to Datafari to learn how to use the recommended JCIFS connector.

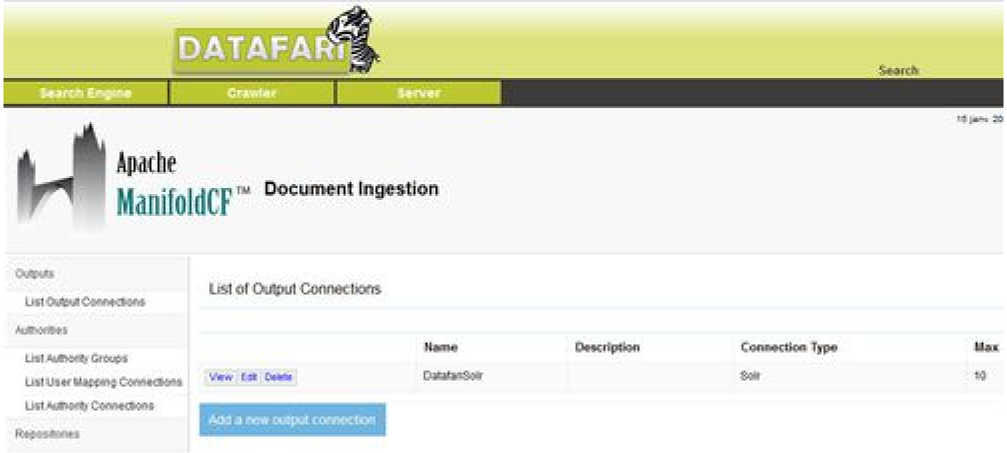
From the admin menu, click on Crawler and then on ManifoldCF Admin.



ManifoldCF then asks you for your credentials, which you have setup either in the config xml file, or in the install phase if you used it.



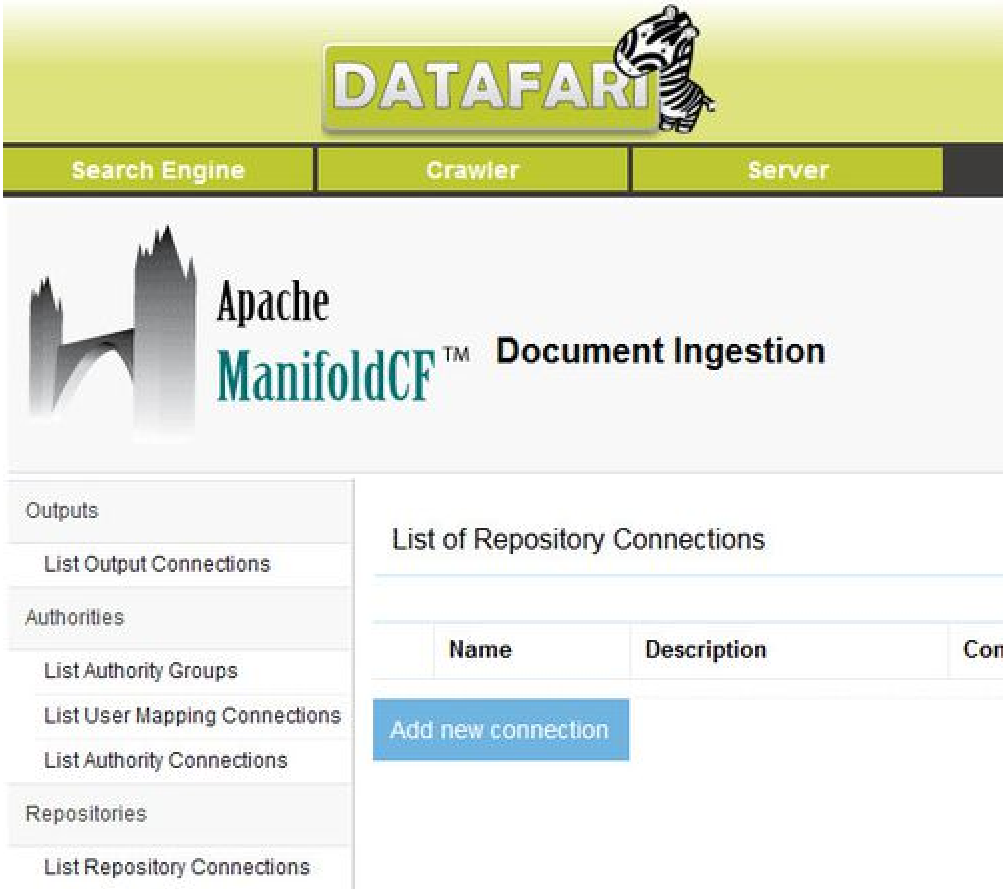
You are now in the MCF admin page. Since this is the MCF of Datafari, an output connection is already configured to point to the Solr of Datafari. You can check that by clicking on "List Output Connections" on the left menu. You will then see on the right, the DatafariSolr output connection.



List of Output Connections

	Name	Description	Connection Type	Max
View Edit Delete	DatafariSolr		Solr	10

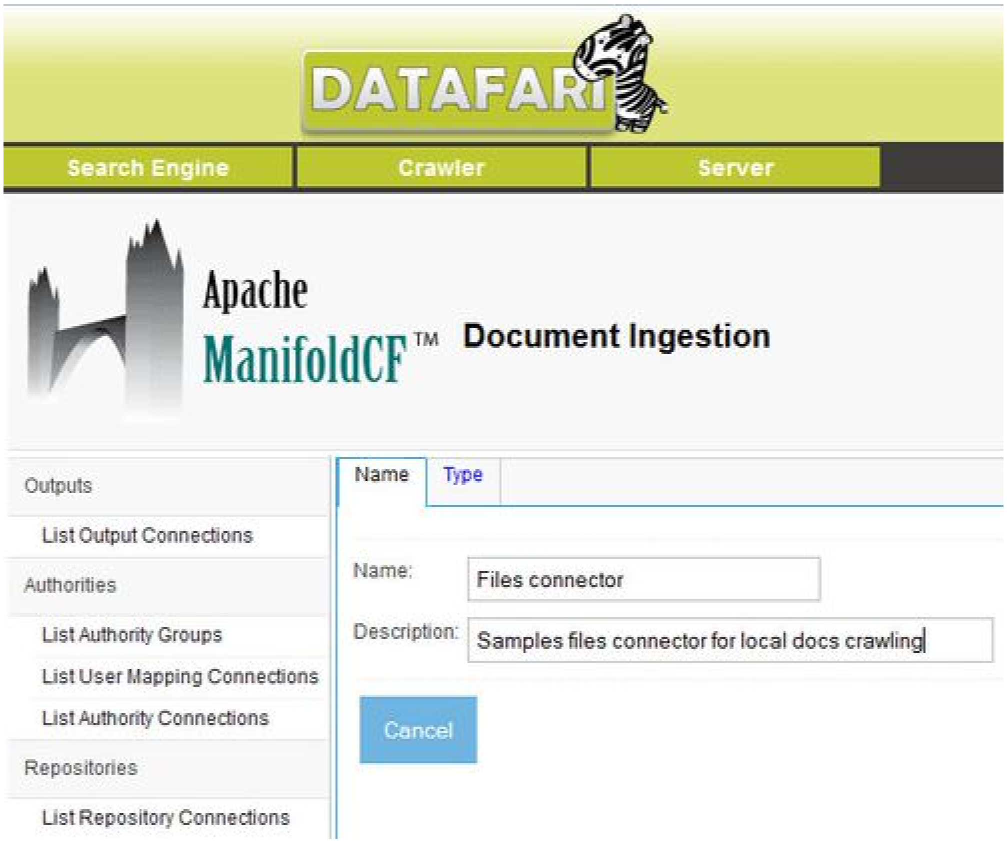
Since we want to crawl local files, we first need to declare a new repository connection (to define the type of connection), and then we will instantiate a job that user this repository connection, pointing a the desired local folder. To declare a new repository connection, click on List Repository Connections on the left menu, then click the Add new connection button on the right panel.



List of Repository Connections

	Name	Description	Con

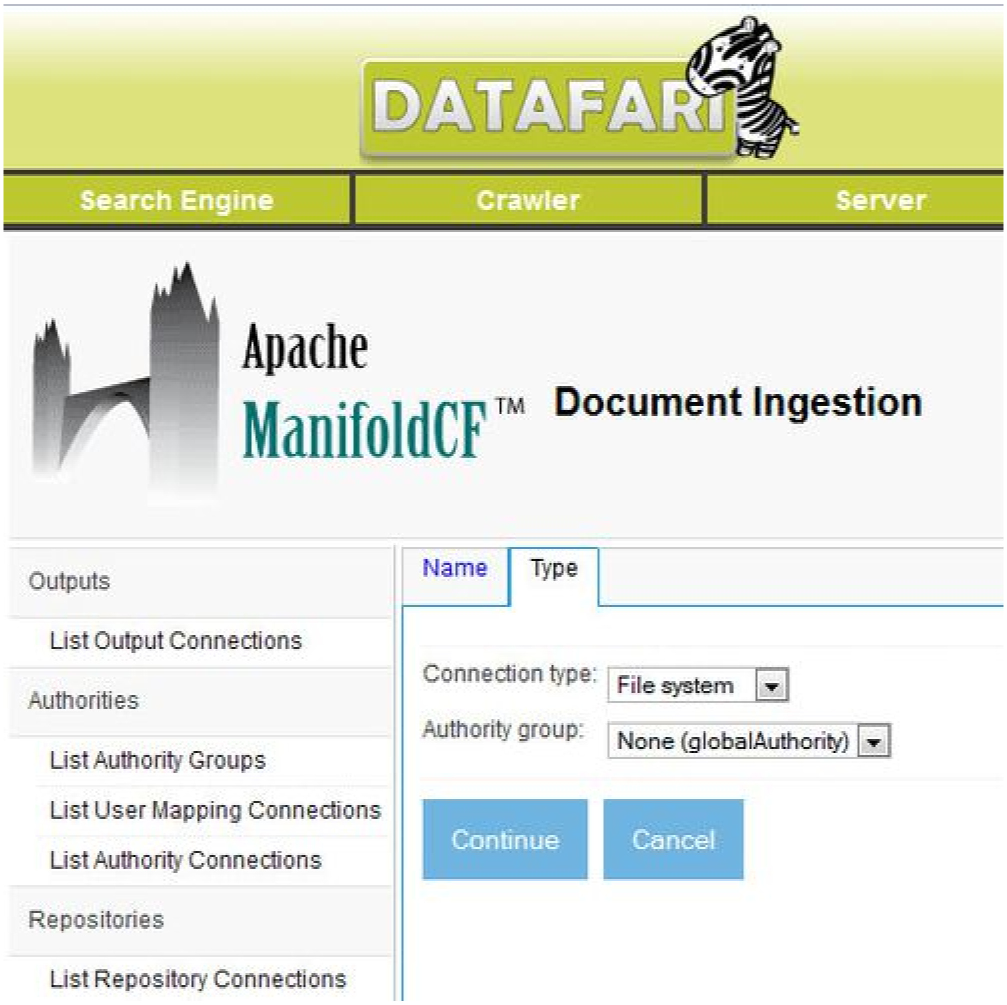
Type in the name and the description, then click on the Type tab.



Name: Files connector

Description: Samples files connector for local docs crawling

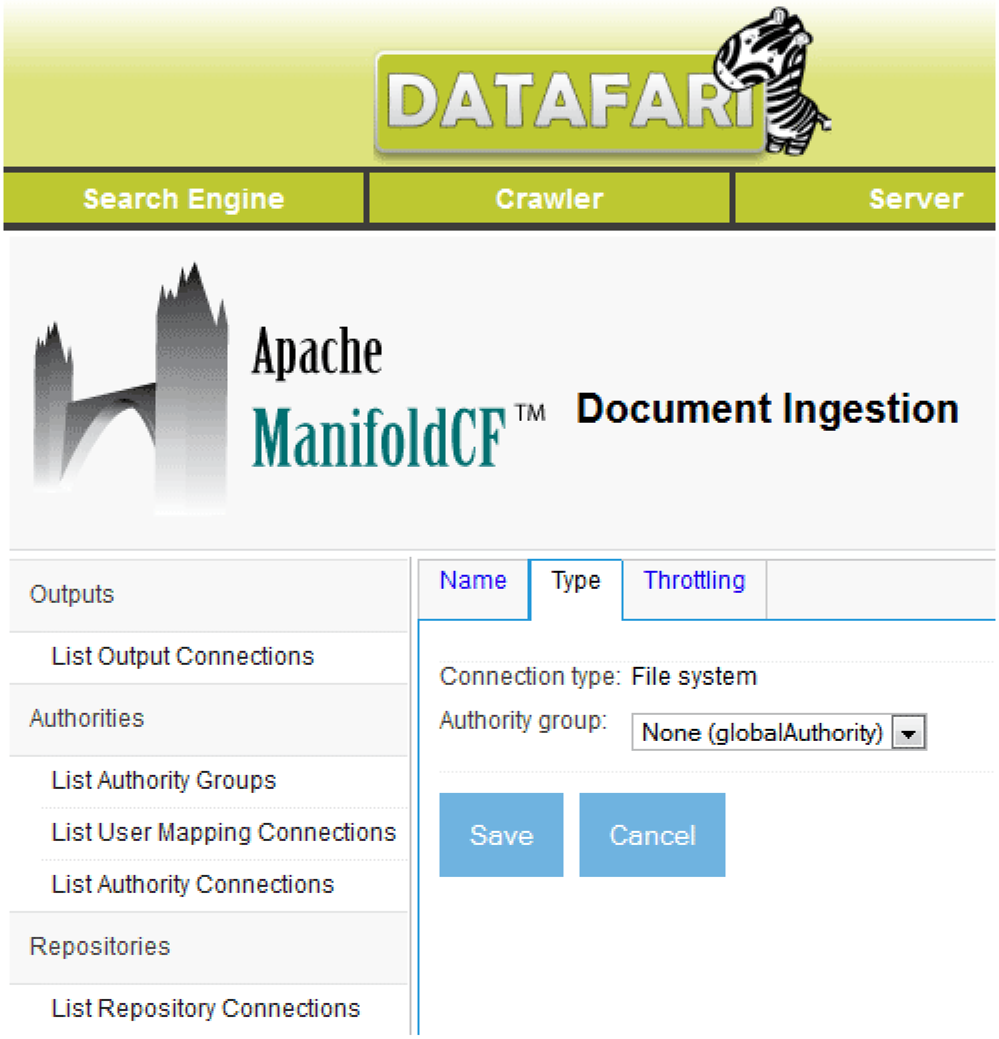
In the type tab, you pick in the dropdown list File System (note that this is a DEMO connector with known bugs, don't use it for production environments), and leave the Authority group to None, since it is quick start exemple.
Again, refer to the full documentation of Apache ManifoldCF to dig deeper into these options.



Connection type: File system

Authority group: None (globalAuthority)

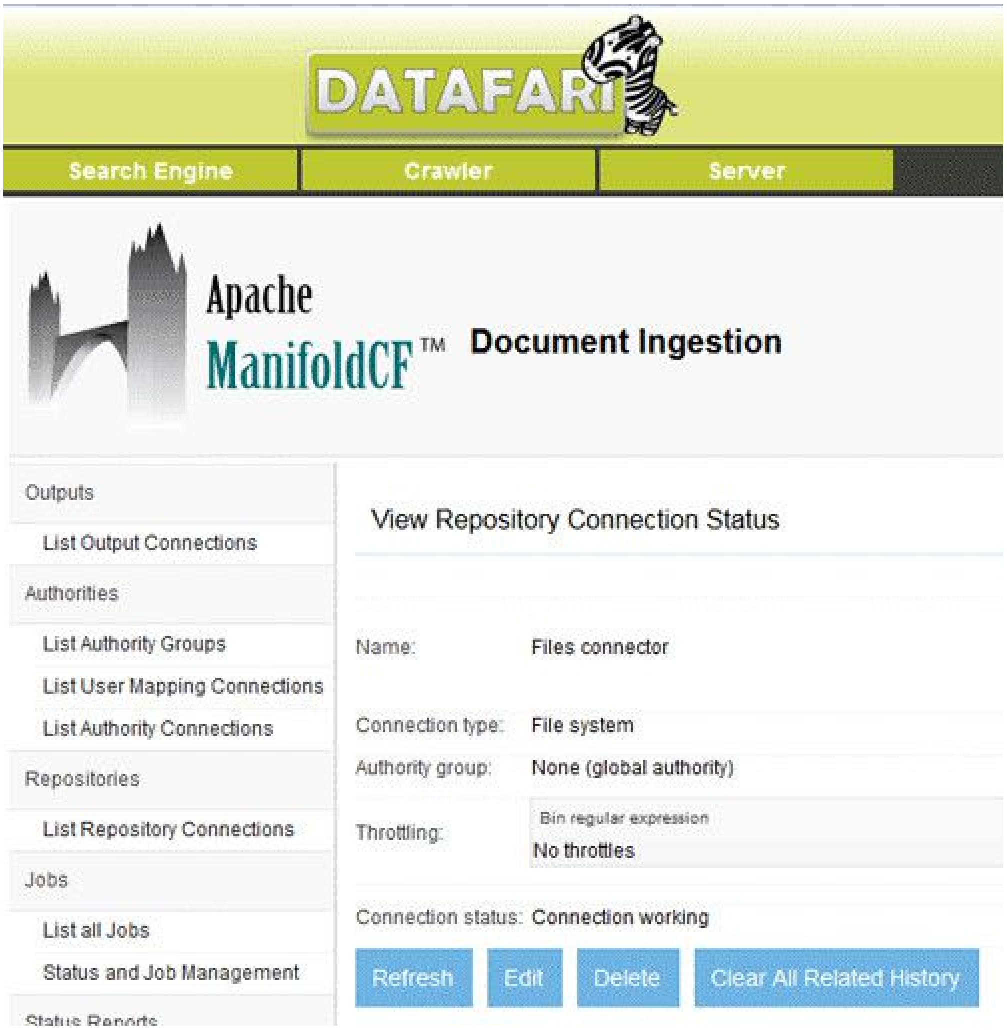
Click on Continue, and a Throttling tab appears, as well as a Save and a Cancel buttons. Click on Save.



Connection type: File system

Authority group: None (globalAuthority)

Clicking on save displays the View Repository Connection Status. You should have something that looks like the illustration below.



View Repository Connection Status

Name: Files connector

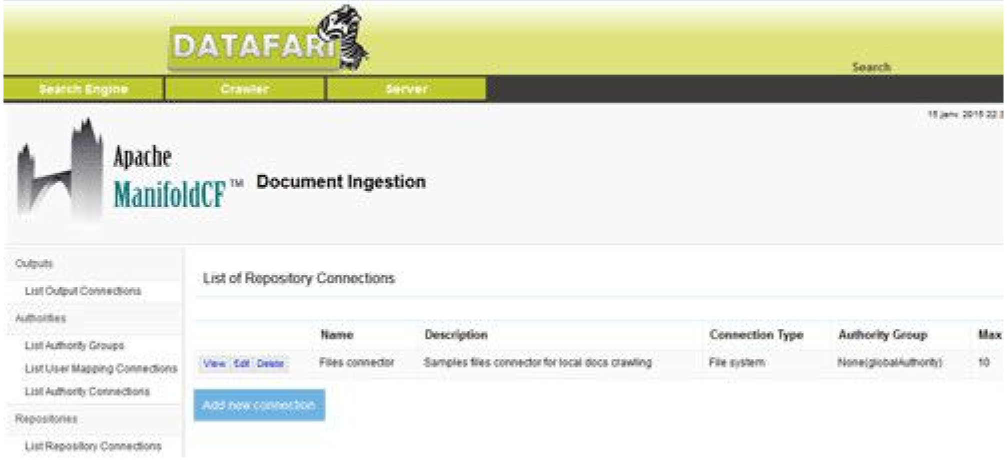
Connection type: File system

Authority group: None (global authority)

Throttling: Bin regular expression
No throttles

Connection status: Connection working

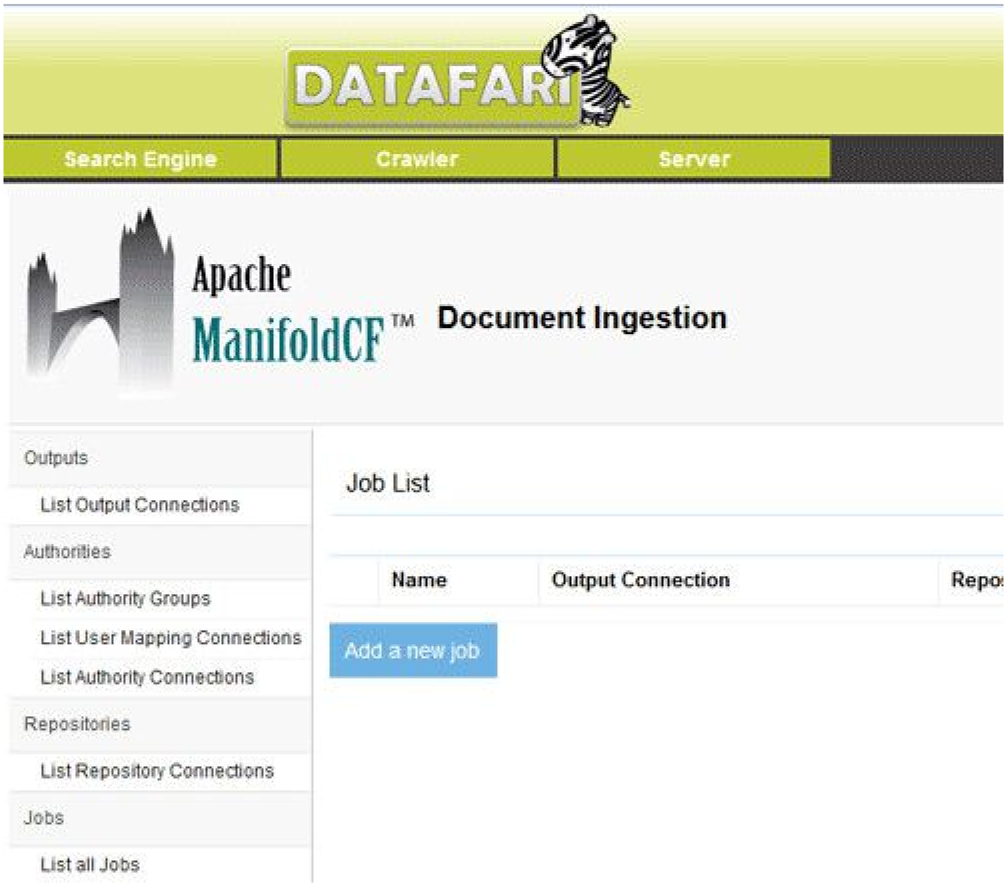
In the menu, click again on List Repository Connections to check that your new connection is now listed. You should have something like this:



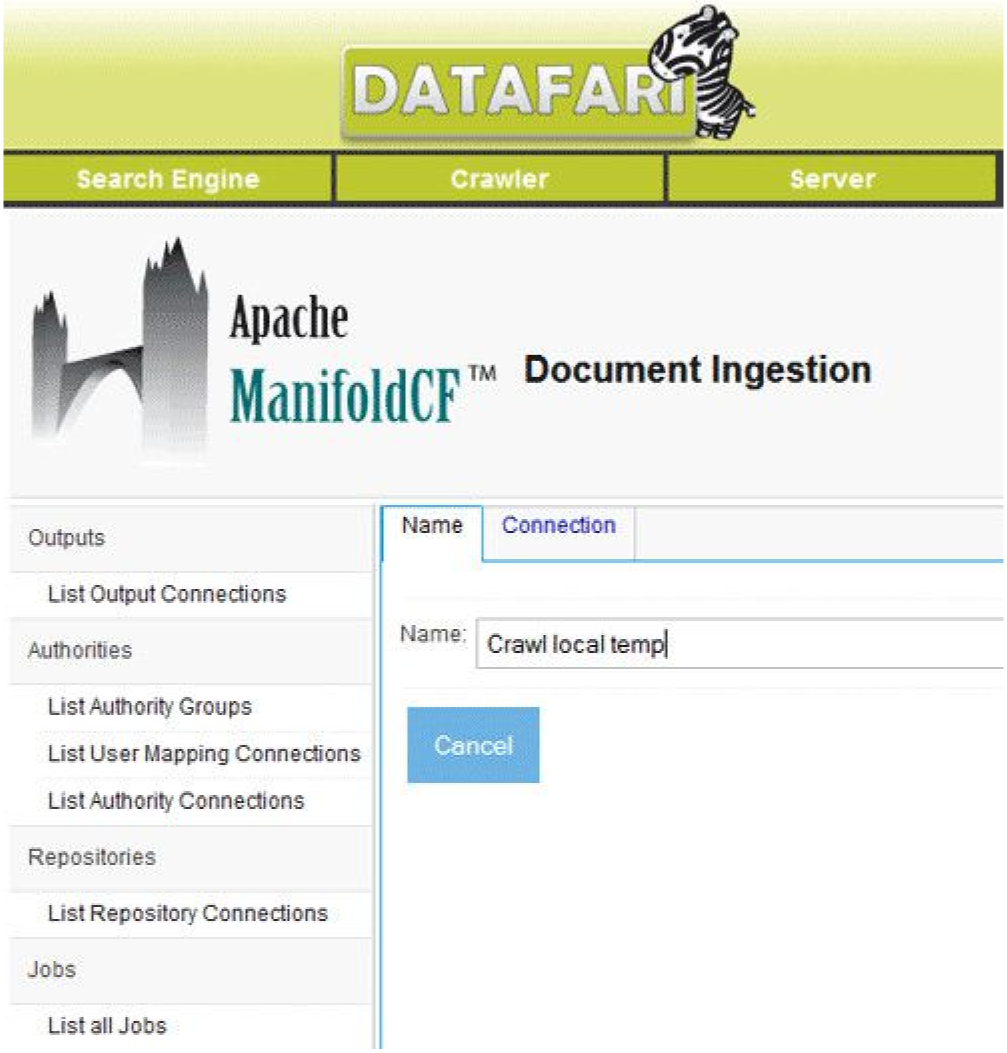
List of Repository Connections

	Name	Description	Connection Type	Authority Group	Max
View Edit Delete	Files connector	Samples files connector for local docs crawling	File system	None(globalAuthority)	10

Now that we have declared a repository connection of type local file system, we can now "instantiate" it through a job. For that, click on the menu on "List all Jobs". The right panel will display an empty job list. Click on the button "Add a new job"

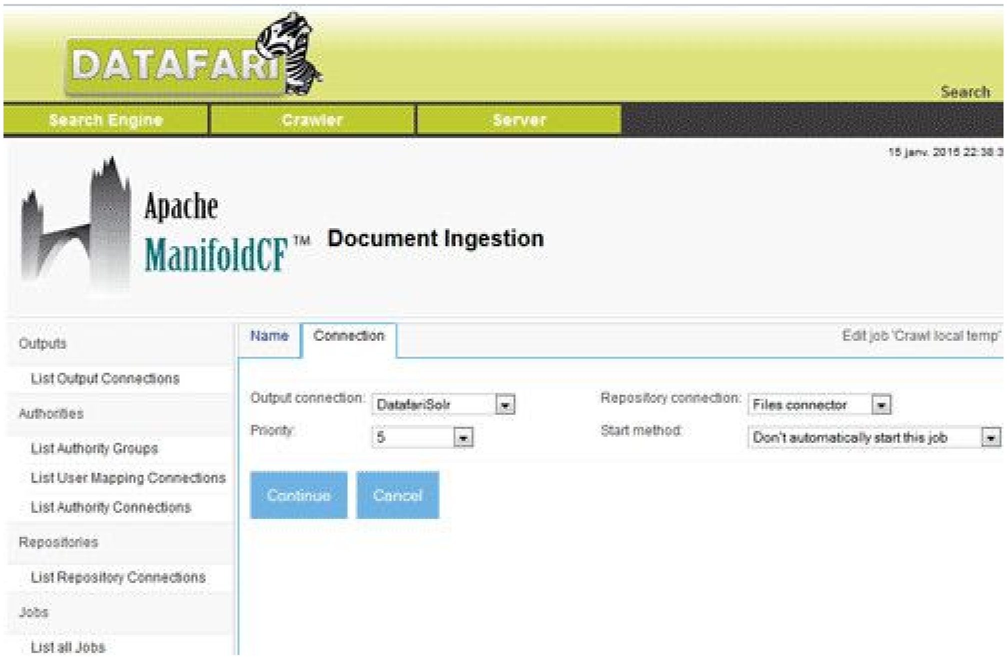


You then need to declare a name. Do it and click on the Connection tab



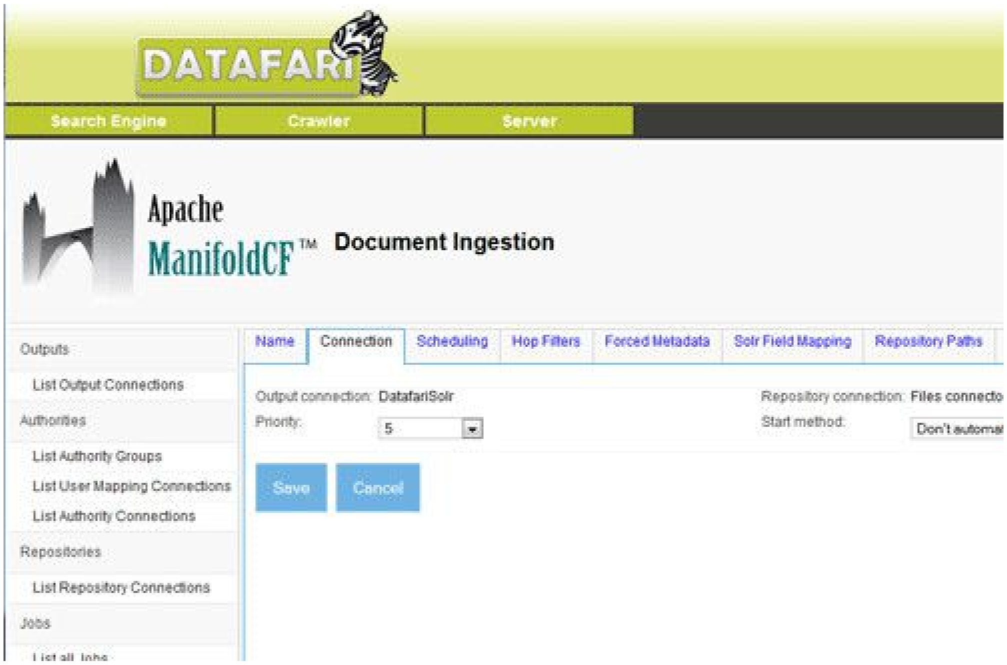
Name: Crawl local temp

In this tab, click on the Output connection dropdown and select the Datafari specific Output connection "DatafariSolr". On the right, click on the Repository connection dropdown and pick the one we just created, in our case it is "Files connector". Then click on Continue.



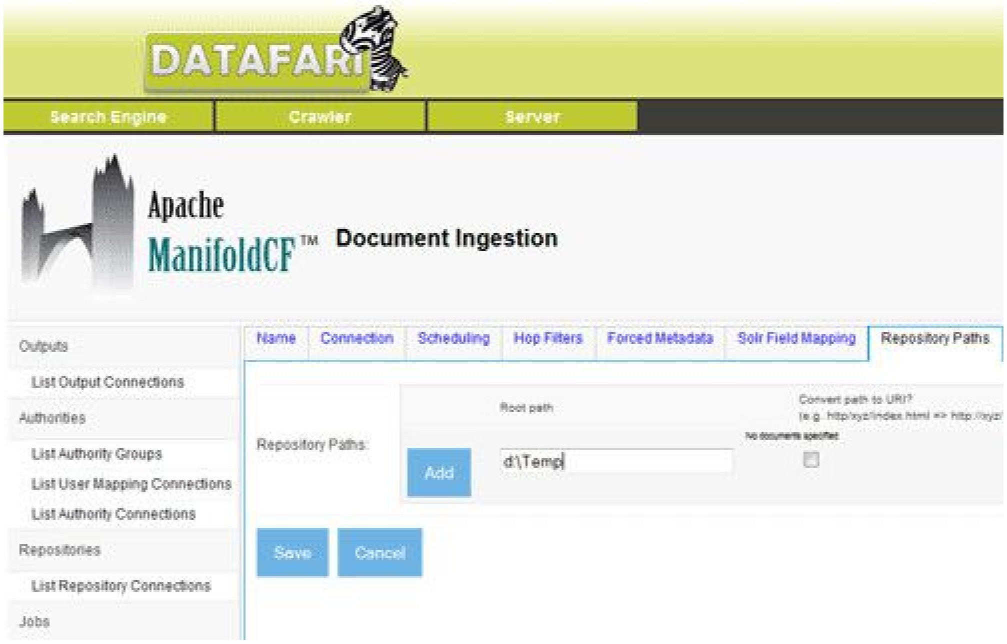
Output connection: DatafariSolr
Priority: 5
Repository connection: Files connector
Start method: Don't automatically start this job

A list of tabs then appears. Check the ManifoldCF official documentation to get an understanding of all of them.



Output connection: DatafariSolr
Repository connection: Files connecto
Priority: 5
Start method: Don't automat

This is where you define the actual location in your local system, that you want the job to crawl and send to Datafari Solr for indexing. For our example, click on the Repository Paths tab. In the illustration below, we enter the local path "d:\Temp" and click on Add (not Save yet!).



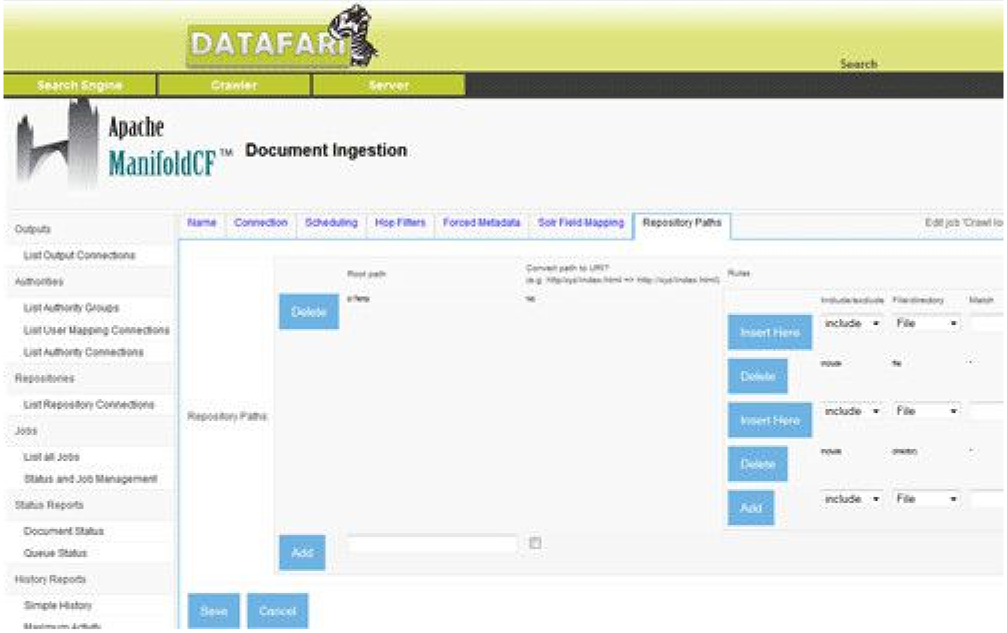
Repository Paths:
No documents specified
Root path: d:\Temp
Convert path to URI? (e.g. http:/xyz/index.html => http://xyz/ □

A list of potential regex appears on the right. Here is some info about the regex, taken from the manifoldcf 2.5 documentation:

> **Using the regex option**
> For each included path, a list of rules is displayed which determines what folders and documents get included with the job. These rules will be evaluated from top to bottom, in order. Whichever rule first matches a given path is the one that will be used for that path.
>
> Each rule describes the path matching criteria. This consists of the file specification (e.g. "*.txt"), whether the path is a file or folder name, and whether a file is considered indexable or not by the output connection. The rule also describes the action to take should the rule be matched: include or exclude. The file specification character "*" is a wildcard which matches zero or more characters, while the character "?" matches exactly one character. All other characters must match exactly.
>
> Remember that your specification must match **all** characters included in the file's path. That includes all path separator characters ("/"). The path you must match always begins with an initial path separator. Thus, if you want to exclude the file "foo.txt" at the root level, your exclude rule must match "/foo.txt".

To add a rule for a starting path, select the desired values of all the pulldowns, type in the desired file criteria, and click the "Add" button. You may also insert a new rule above any existing rule, by using one of the "Insert" buttons.

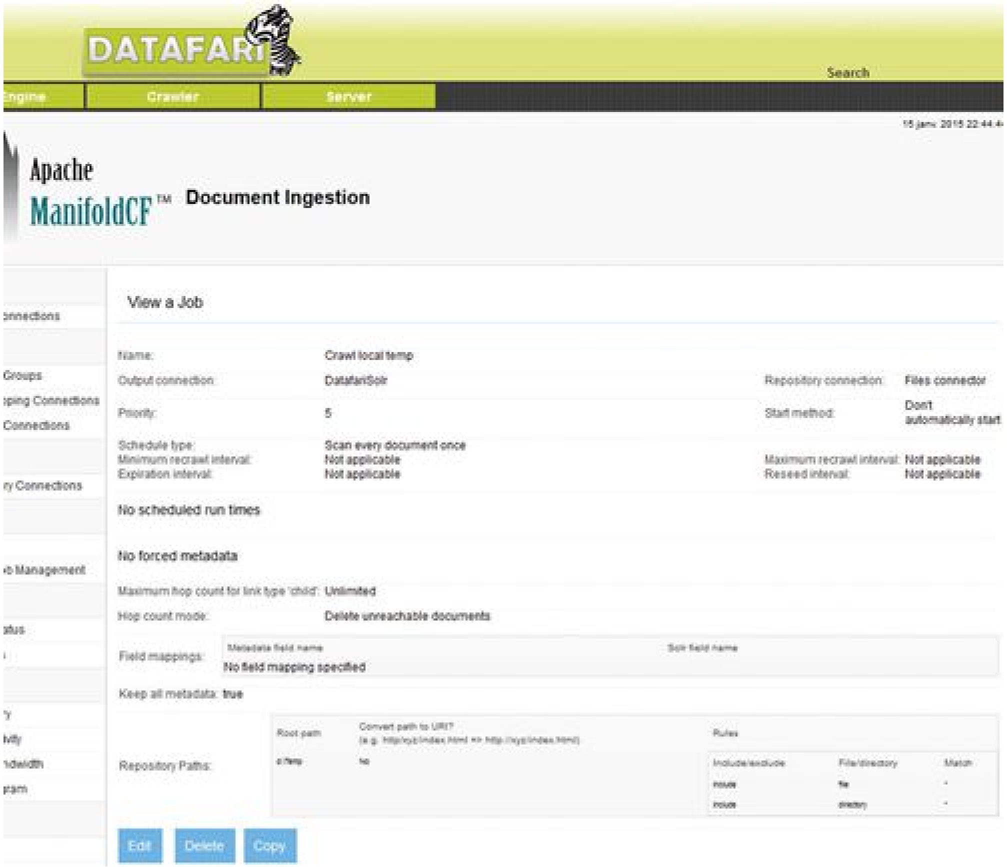
Click now on the Save button.



Repository Paths:

	Root path	Convert path to URI? (e.g. http/xyz/index.html => http://xyz/index.html)	Rules			
				Include/exclude	File/directory	Match
Delete	c:\temp	no	Insert Here	include ▾	File ▾	
			Delete	include	file	*
			Insert Here	include ▾	File ▾	
			Delete	include	directory	*
			Add	include ▾	File ▾	
Add		□				

You get the confirmation that the job is succesfully created when you see the summary of the created job, in the "View a Job" window. This should like the following illustration:



View a Job

Name: Crawl local temp

Output connection: DatafariSolr | Repository connection: Files connector

Priority: 5 | Start method: Don't automatically start

Schedule type: Scan every document once
Minimum recrawl interval: Not applicable | Maximum recrawl interval: Not applicable
Expiration interval: Not applicable | Reseed interval: Not applicable

No scheduled run times

No forced metadata

Maximum hop count for link type 'child': Unlimited

Hop count mode: Delete unreachable documents

Field mappings:

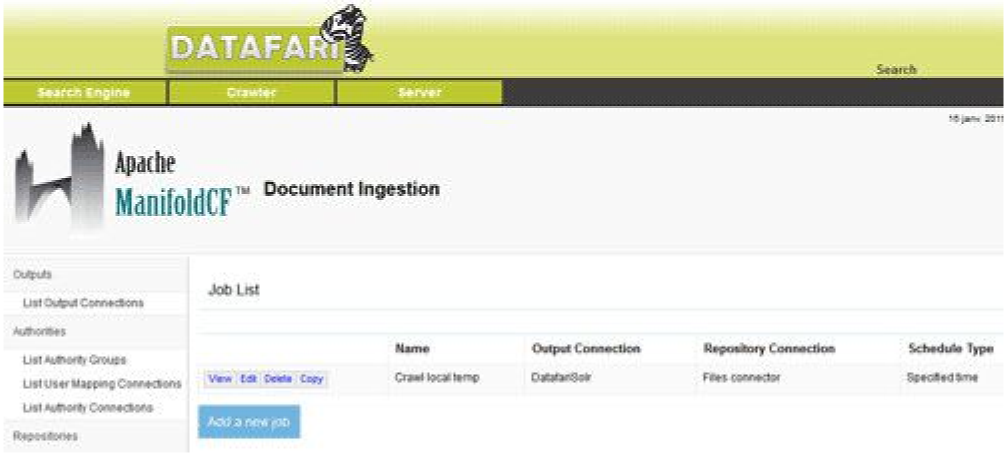
Metadata field name	Solr field name
No field mapping specified	

Keep all metadata: true

Repository Paths:

Root path	Convert path to URI? (e.g. http/xyz/index.html => http://xyz/index.html)	Rules		
		Include/exclude	File/directory	Match
c:\temp	no	include	file	*
		include	directory	*

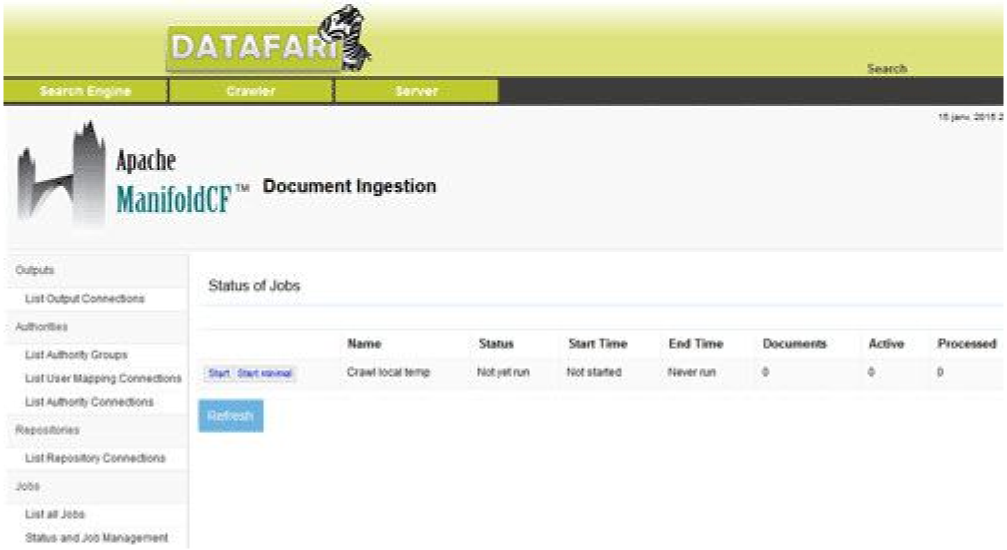
To ensure things really went ok, go on the left menu and select "List all Jobs". You should see on the right panel a job list with only one entry, the job you just created, as in the illustration below.



Job List

	Name	Output Connection	Repository Connection	Schedule Type
View Edit Delete Copy	Crawl local temp	DatafariSolr	Files connector	Specified time

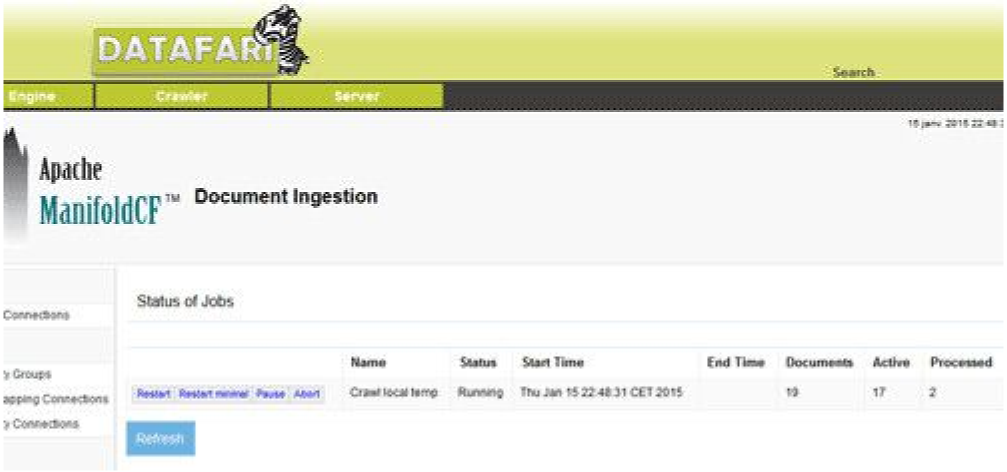
It is now time start the actual crawling of the folder. For that, click in the left on menu on "Status and Job Management". You should get a Jobs status list on the right pane, with only one entry, which is the job you recently created. Click on Start to trigger the crawling, and wait for the magic to happen.



Status of Jobs

	Name	Status	Start Time	End Time	Documents	Active	Processed
Start Start minimal	Crawl local temp	Not yet run	Not started	Never run	0	0	0

If you want, you can click several time on the Refresh button to get a feeling that things are being done. This can be observed by looking the last 3 fields of the list: "Documents", "Active" and "Processed", whose numbers should be evolving as you click on Refresh. Wait for a few documents to be counted in the "Processed" field, and you can then start searching for it in the Search view, which is accessible at any time through the Search link on the upper right of the Datafari header.



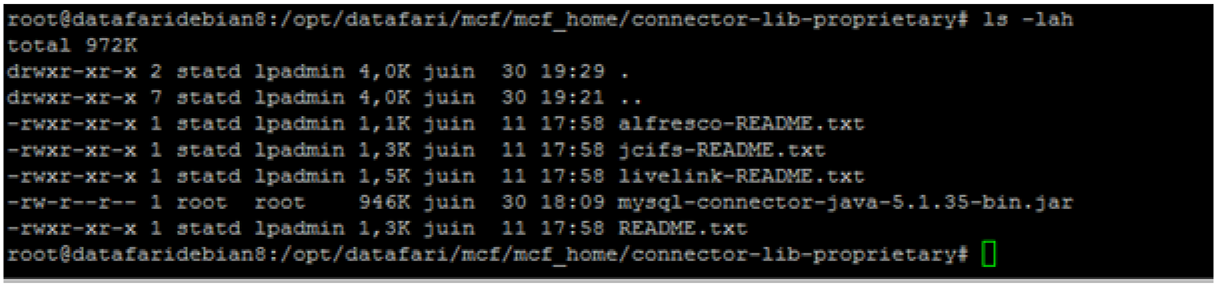
Status of Jobs

	Name	Status	Start Time	End Time	Documents	Active	Processed
Restart Restart minimal Pause Abort	Crawl local temp	Running	Thu Jan 15 22:48:31 CET 2015		19	17	2

Add a JDBC connector (MySQL, Oracle)

In order to crawl databases as MySQL or Oracle databases, please respect these steps (this example is for Debian and for a MySQL database but it is almost the same for Windows version) :

- Download the Java connector for your database
- Add the JAR file into the folder /opt/datafari/mcf_home/connector-lib-proprietary



```
root@datafaridebian8:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# ls -lah
total 972K
drwxr-xr-x 2 statd lpadmin 4,0K juin  30 19:29 .
drwxr-xr-x 7 statd lpadmin 4,0K juin  30 19:21 ..
-rwxr-xr-x 1 statd lpadmin 1,1K juin  11 17:58 alfresco-README.txt
-rwxr-xr-x 1 statd lpadmin 1,3K juin  11 17:58 jcifs-README.txt
-rwxr-xr-x 1 statd lpadmin 1,5K juin  11 17:58 livelink-README.txt
-rw-r--r-- 1 root  root    946K juin  30 18:09 mysql-connector-java-5.1.35-bin.jar
-rwxr-xr-x 1 statd lpadmin 1,3K juin  11 17:58 README.txt
root@datafaridebian8:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# 
```

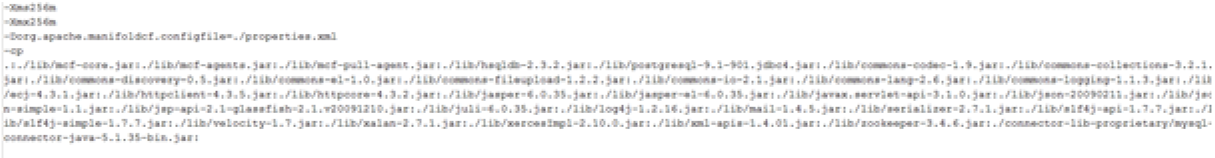
Here : mysql-connector-java-5.1.35-bin.jar

- Edit the file : options.env.unix (in /opt/datafari/mcf/mcf_home) (if you are on Windows, the file is options.env.win)

And add the path to the new lib in the -cp parameter :

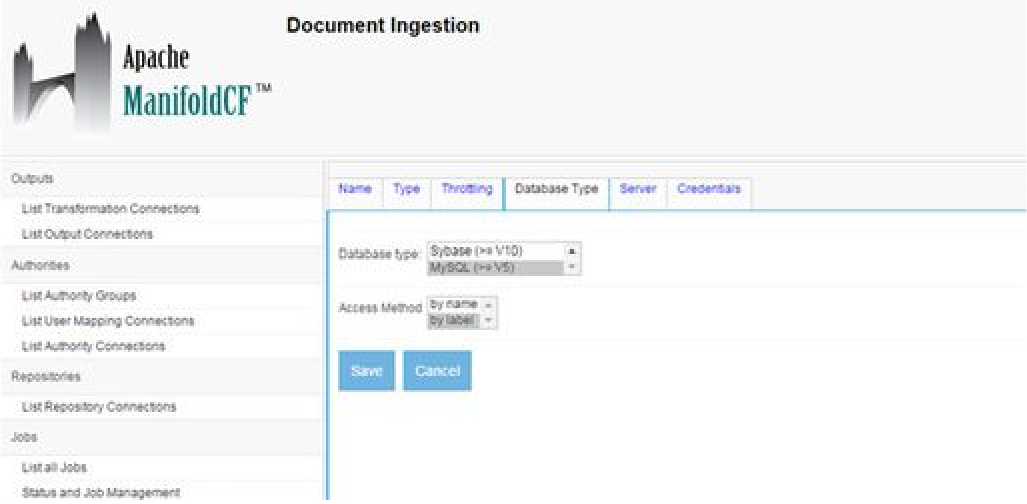
./connector-lib-proprietary/mysql-connector-java-5.1.35-bin.jar:

You will have this :



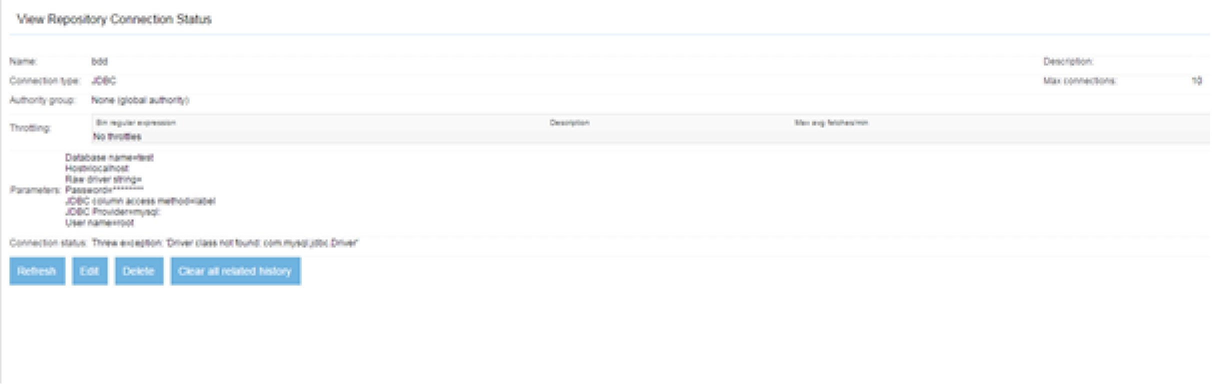
```
-Xms256m
-Xmx256m
-Dorg.apache.manifoldcf.configfile=./properties.xml
-cp
.:./lib/mcf-core.jar:./lib/mcf-agents.jar:./lib/mcf-pull-agent.jar:./lib/hsqldb-2.3.2.jar:./lib/postgresql-9.1-901.jdbc4.jar:./lib/commons-codec-1.9.jar:./lib/commons-collections-3.2.1.
jar:./lib/commons-discovery-0.5.jar:./lib/commons-el-1.0.jar:./lib/commons-fileupload-1.2.2.jar:./lib/commons-io-2.1.jar:./lib/commons-lang-2.6.jar:./lib/commons-logging-1.1.3.jar:./lib
/ecj-4.3.1.jar:./lib/httpclient-4.3.5.jar:./lib/httpcore-4.3.2.jar:./lib/jasper-6.0.35.jar:./lib/jasper-el-6.0.35.jar:./lib/javax.servlet-api-3.1.0.jar:./lib/json-20090211.jar:./lib/js
n-simple-1.1.jar:./lib/jsp-api-2.1-glassfish-2.1.v20091210.jar:./lib/juli-6.0.35.jar:./lib/log4j-1.2.16.jar:./lib/mail-1.4.5.jar:./lib/serializer-2.7.1.jar:./lib/slf4j-api-1.7.7.jar:./l
ib/slf4j-simple-1.7.7.jar:./lib/velocity-1.7.jar:./lib/xalan-2.7.1.jar:./lib/xercesImpl-2.10.0.jar:./lib/xml-apis-1.4.01.jar:./lib/zookeeper-3.4.6.jar:./connector-lib-proprietary/mysql-
connector-java-5.1.35-bin.jar:
```

- Restart Datafari to load the library
- You can now index your Database in ManifoldCF
  - Configure the repository connection :



Database type: Sybase (>= V10) [✓ MySQL (>= V5)]

Access Method: by name [✓ by label]



View Repository Connection Status

Name:	bdd	Description:	
Connection type:	JDBC	Max connections:	10
Authority group:	None (global authority)		

Throttling:

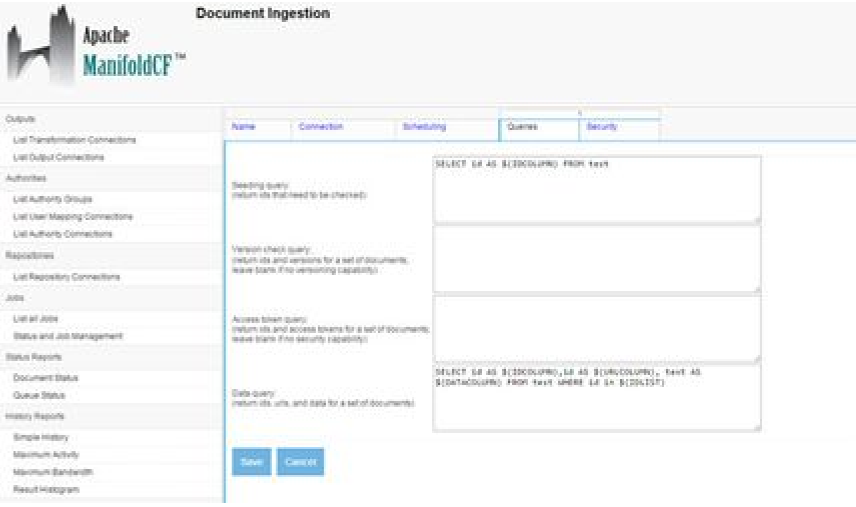
Bin regular expression	Description	Max avg fetches/min
No throttles		

Parameters:
Database name=test
Host=localhost
Raw driver string=
Password=********
JDBC column access method=label
JDBC Provider=mysql:
User name=root

Connection status: Threw exception: 'Driver class not found: com.mysql.jdbc.Driver'

You will see the message : 'Driver class not found' .But it is not the case : the driver is correctly loaded.

- Next configure the job as usual :



Seeding query:
(return ids that need to be checked)

```
SELECT id AS $(IDCOLUMN) FROM test
```

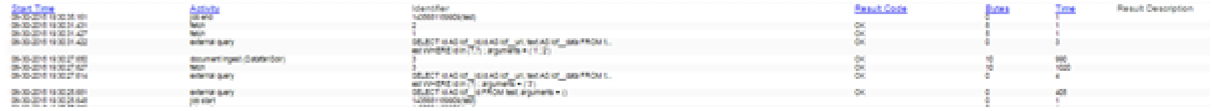
Version check query:
(return ids and versions for a set of documents;
leave blank if no versioning capability)

Access token query:
(return ids and access tokens for a set of documents;
leave blank if no security capability)

Data query:
(return ids, urls, and data for a set of documents)

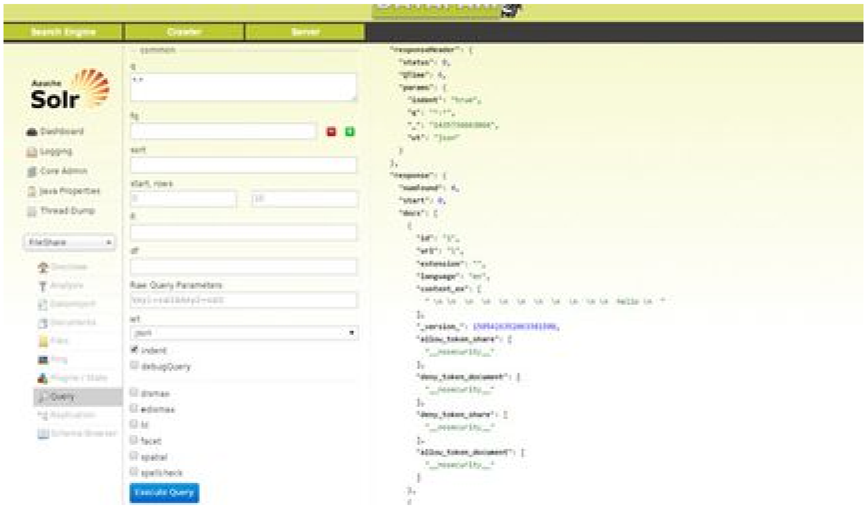
```
SELECT id AS $(IDCOLUMN),id AS $(URLCOLUMN), text AS
$(DATACOLUMN) FROM test WHERE id in $(IDLIST)
```

And you will see in the history that the job is correctly executed.



Start Time	Activity	Identifier	Result Code	Bytes	Time	Result Description
06-30-2015 19:30:35.161	job end	1435651105609(test)		0	1	
06-30-2015 19:30:31.431	fetch	2	OK	8	1	
06-30-2015 19:30:31.427	fetch	1	OK	8	1	
06-30-2015 19:30:31.422	external query	SELECT id AS lcf__id,id AS lcf__url, text AS lcf__data FROM test WHERE id in (?,?) ; arguments = (1,2)	OK	0	3	
06-30-2015 19:30:27.683	document ingest (DataferSolr)	1	OK	10	880	
06-30-2015 19:30:27.627	fetch	1	OK	10	1020	
06-30-2015 19:30:27.614	external query	SELECT id AS lcf__id,id AS lcf__url, text AS lcf__data FROM test WHERE id in (?) ; arguments = (1)	OK	0	4	
06-30-2015 19:30:25.681	external query	SELECT id AS lcf__id FROM test; arguments = ()	OK	0	405	
06-30-2015 19:30:25.648	job start	1435651105609(test)		0	1	

You can check it in Solr admin :



q: *:*
fq:
sort:
start, rows: 0 10
fl:
df:
Raw Query Parameters: key1=val1&key2=val2
wt: json
☑ indent
□ debugQuery
□ dismax
□ edismax
□ hl
□ facet
□ spatial
□ spellcheck

```
"responseHeader": {
  "status": 0,
  "QTime": 4,
  "params": {
    "indent": "true",
    "q": "*:*",
    "_": "1435735660904",
    "wt": "json"
  }
},
"response": {
  "numFound": 4,
  "start": 0,
  "docs": [
    {
      "id": "1",
      "url": "1",
      "extension": "",
      "language": "en",
      "content_en": [
        " \n \n  \n  \n  \n  \n  \n  \n  \n  \n \n hello \n "
      ],
      "_version_": 1505426352803381248,
      "allow_token_share": [
        "__nosecurity__"
      ],
      "deny_token_document": [
        "__nosecurity__"
      ],
      "deny_token_share": [
        "__nosecurity__"
      ],
      "allow_token_document": [
        "__nosecurity__"
      ]
    },
    {
```

Add the JCIFS Connector to Datafari

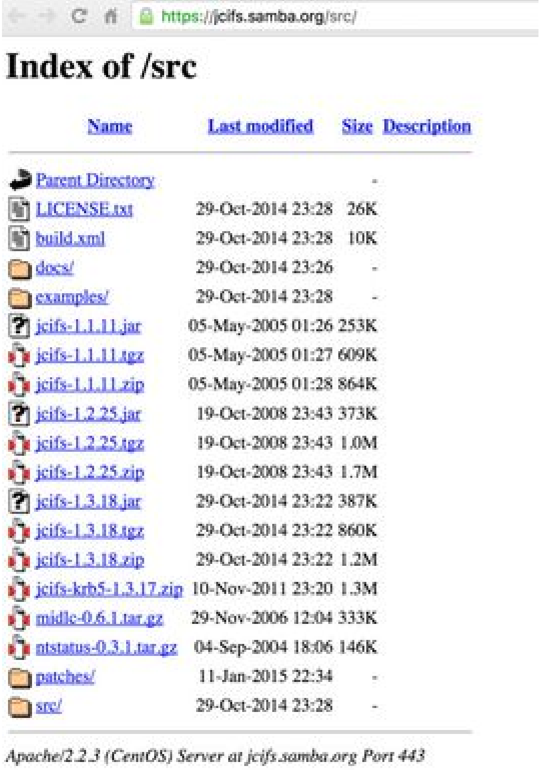
Due to licences issues, we do not have the right to package the JCIFS library with Datafari. So you have to download it on your own and to add it to Datafari.

If you want to add the JCIFS connector to the list of existing connectors, here it is the procedure in order to do it. We distinct 2 cases :

- you just downloaded Datafari and you NEVER launched it
- you have an existing Datafari running

The beginning is the same for the two cases. If you are in the second case, you just need to follow few more steps that we will detail (please stop Datafari : bash stop-datafari.sh before beginning).

1. Download the jcifs-1.3.xx.jar from http://jcifs.samba.org/src/



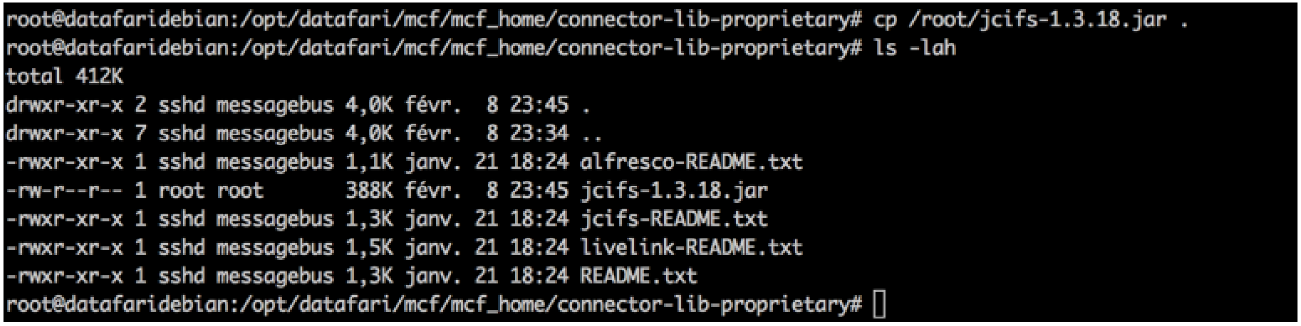
https://jcifs.samba.org/src/

Index of /src

Name	Last modified	Size	Description
Parent Directory		-	
LICENSE.txt	29-Oct-2014 23:28	26K	
build.xml	29-Oct-2014 23:28	10K	
docs/	29-Oct-2014 23:26	-	
examples/	29-Oct-2014 23:28	-	
jcifs-1.1.11.jar	05-May-2005 01:26	253K	
jcifs-1.1.11.tgz	05-May-2005 01:27	609K	
jcifs-1.1.11.zip	05-May-2005 01:28	864K	
jcifs-1.2.25.jar	19-Oct-2008 23:43	373K	
jcifs-1.2.25.tgz	19-Oct-2008 23:43	1.0M	
jcifs-1.2.25.zip	19-Oct-2008 23:43	1.7M	
jcifs-1.3.18.jar	29-Oct-2014 23:22	387K	
jcifs-1.3.18.tgz	29-Oct-2014 23:22	860K	
jcifs-1.3.18.zip	29-Oct-2014 23:22	1.2M	
jcifs-krb5-1.3.17.zip	10-Nov-2011 23:20	1.3M	
midlc-0.6.1.tar.gz	29-Nov-2006 12:04	333K	
ntstatus-0.3.1.tar.gz	04-Sep-2004 18:06	146K	
patches/	11-Jan-2015 22:34	-	
src/	29-Oct-2014 23:28	-	

Apache/2.2.3 (CentOS) Server at jcifs.samba.org Port 443

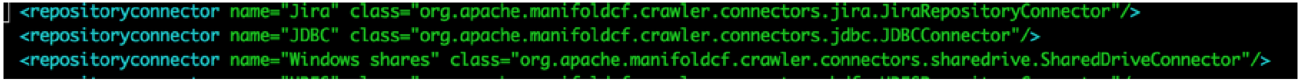
2. Copy the JAR to DATAFARI_SOURCE_DIR\mcf\mcf_home\connector-lib-proprietary
   So if you are in Debian for example, copy to /opt/datafari/mcf/mcf_home/connector-lib-proprietary



```
root@datafaridebian:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# cp /root/jcifs-1.3.18.jar .
root@datafaridebian:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# ls -lah
total 412K
drwxr-xr-x 2 sshd messagebus 4,0K févr.  8 23:45 .
drwxr-xr-x 7 sshd messagebus 4,0K févr.  8 23:34 ..
-rwxr-xr-x 1 sshd messagebus 1,1K janv. 21 18:24 alfresco-README.txt
-rw-r--r-- 1 root root       388K févr.  8 23:45 jcifs-1.3.18.jar
-rwxr-xr-x 1 sshd messagebus 1,3K janv. 21 18:24 jcifs-README.txt
-rwxr-xr-x 1 sshd messagebus 1,5K janv. 21 18:24 livelink-README.txt
-rwxr-xr-x 1 sshd messagebus 1,3K janv. 21 18:24 README.txt
root@datafaridebian:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# 
```

3. Rename it into jcifs.jar
   mv jcifs-1.3.18.jar jcifs.jar
4. Edit the file /opt/datafari/mcf/mcf_home/connectors.xml and uncomment the line :

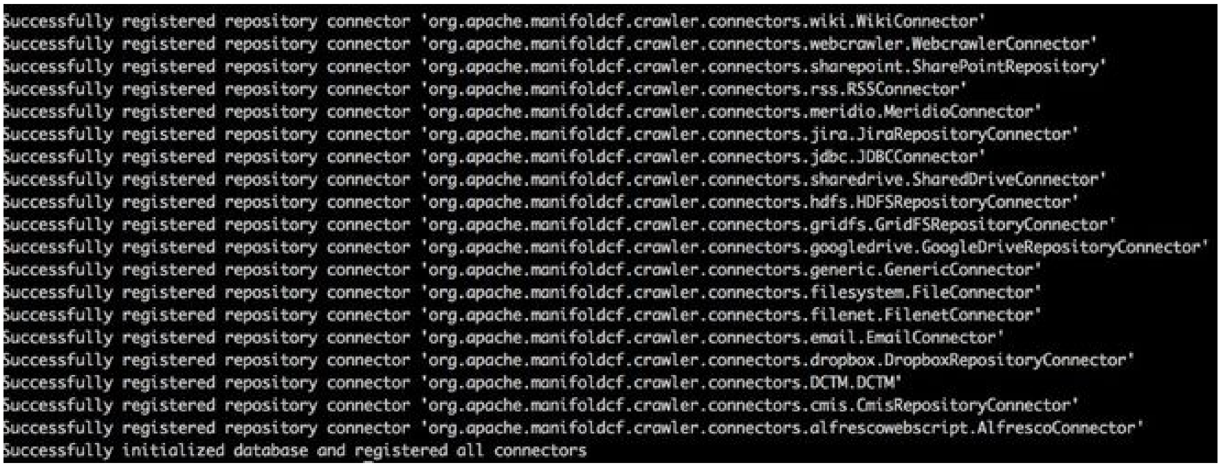
<!--repositoryconnector name="Windows shares" class="org.apache.manifoldcf.crawler.connectors.sharedrive.SharedDriveConnector"/



```
<repositoryconnector name="Jira" class="org.apache.manifoldcf.crawler.connectors.jira.JiraRepositoryConnector"/>
<repositoryconnector name="JDBC" class="org.apache.manifoldcf.crawler.connectors.jdbc.JDBCConnector"/>
<repositoryconnector name="Windows shares" class="org.apache.manifoldcf.crawler.connectors.sharedrive.SharedDriveConnector"/>
```

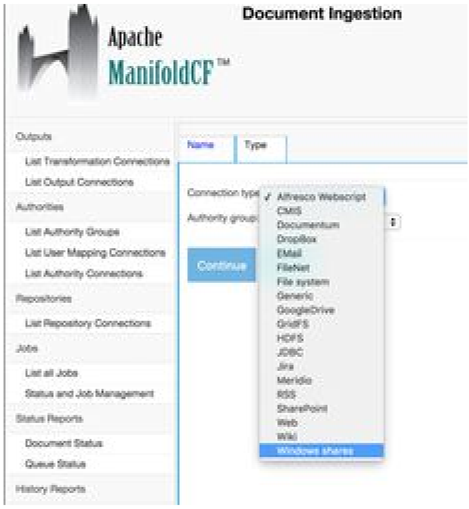
If you just downloaded Datafari and never launched it, you are done !

5. Additional step if you have an existing Datafari :
    a. You need to initialize the connectors in MCF. To do it the simplest is to add this script in /opt/datafari/bin : initialize_connectors.sh
    b. Then authorize the execution :
       chmod u+x initialize_connectors.sh
    c. Finally launch the script :
       bash initialize_connectors.sh
       You will see that the connector is now in the list of MCF connectors :



```
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.wiki.WikiConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.webcrawler.WebcrawlerConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.sharepoint.SharePointRepository'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.rss.RSSConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.meridio.MeridioConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.jira.JiraRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.jdbc.JDBCConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.sharedrive.SharedDriveConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.hdfs.HDFSRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.gridfs.GridFSRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.googledrive.GoogleDriveRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.generic.GenericConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.filesystem.FileConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.filenet.FilenetConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.email.EmailConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.dropbox.DropboxRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.DCTM.DCTM'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.cmis.CmisRepositoryConnector'
Successfully registered repository connector 'org.apache.manifoldcf.crawler.connectors.alfrescowebscript.AlfrescoConnector'
Successfully initialized database and registered all connectors
```

Finally you can launch Datafari. In the repository menu of MCF, the new connector is now present :



Advanced Crawling website

Note : You can use the standard configuration on Datafari for crawling a website (pages and files).

If you want to crawl only webpages (without files), you can customize the part of the page to be indexed. This configuration is done for you.

Starting from Datafari 3.2 version, we add the possibility to crawl easily a website into Datafari.

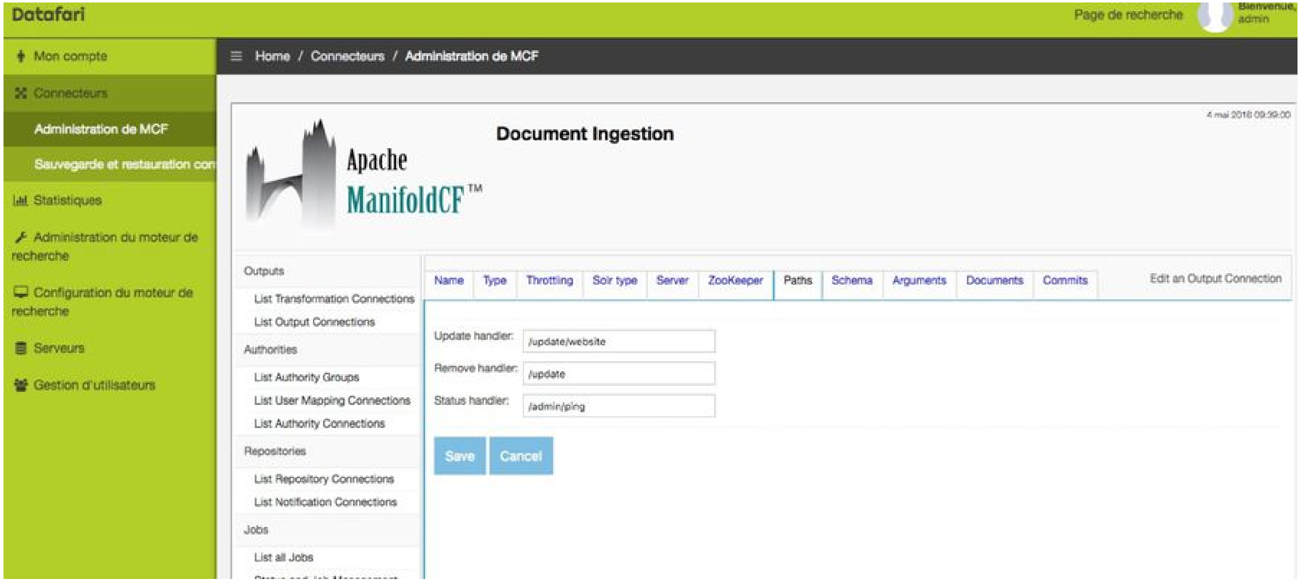
Indeed, if you use the normal crawling with the web connector from MCF, the raw stream of the page is into your Solr document : it means that it is the same content that if you display the HTML source code of the page, which is not very nice for the users.

In order to easily improve that, we add the JSOUP library to the Datafari project. JSOUP is basically a HTML parser. You can indicate with a CSS selector which part of the content to be gathered. Thanks to that, you can control the parts of the webpages that you want to index. The tags are also removed to keep only the content.

Let's follow this tutorial to use it in Datafari in few steps :

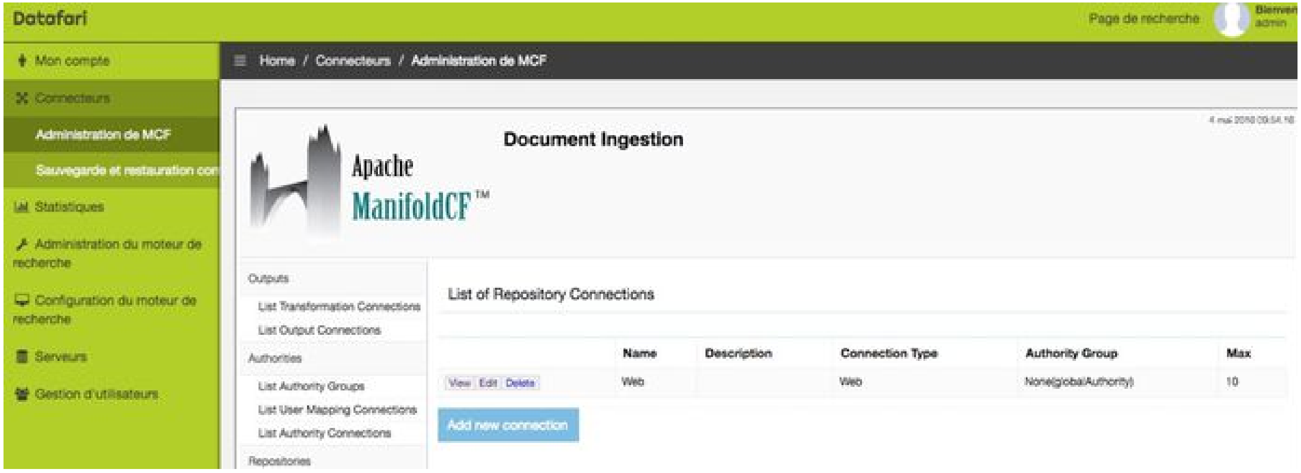
1) Go to MCF admin page

- Add an output connector :
  Click on List Output Connectors.
  Then click on the button Add a ne output connector.
  The configuration is exactly the same that for the standard output connector that already exists named DatafariSolr so you can copy/paste all the configuration except the Paths tab configuration and obviously the name of the output connector : in this example Solr.
  Change the update handler field to **/update/website** (instead of /update/extract).
  We will see at the end of the tutorial in what consists the configuration of that handler in Solr.



Update handler: /update/website
Remove handler: /update
Status handler: /admin/ping

- Add a repository connector
  Now that we have the configuration for the Solr handler we have to add the Web repository connector and the associated job. So click on List Repository Connectors. Then click on Add new connection.
  Name : what you want, in this example Web.
  Connection Type : Web
  Authority group : None
  For the other values, I invite you to read the MCF documentation about it. You can leave the default configuration, just add an email address to identify the crawler in the website that you will crawl.

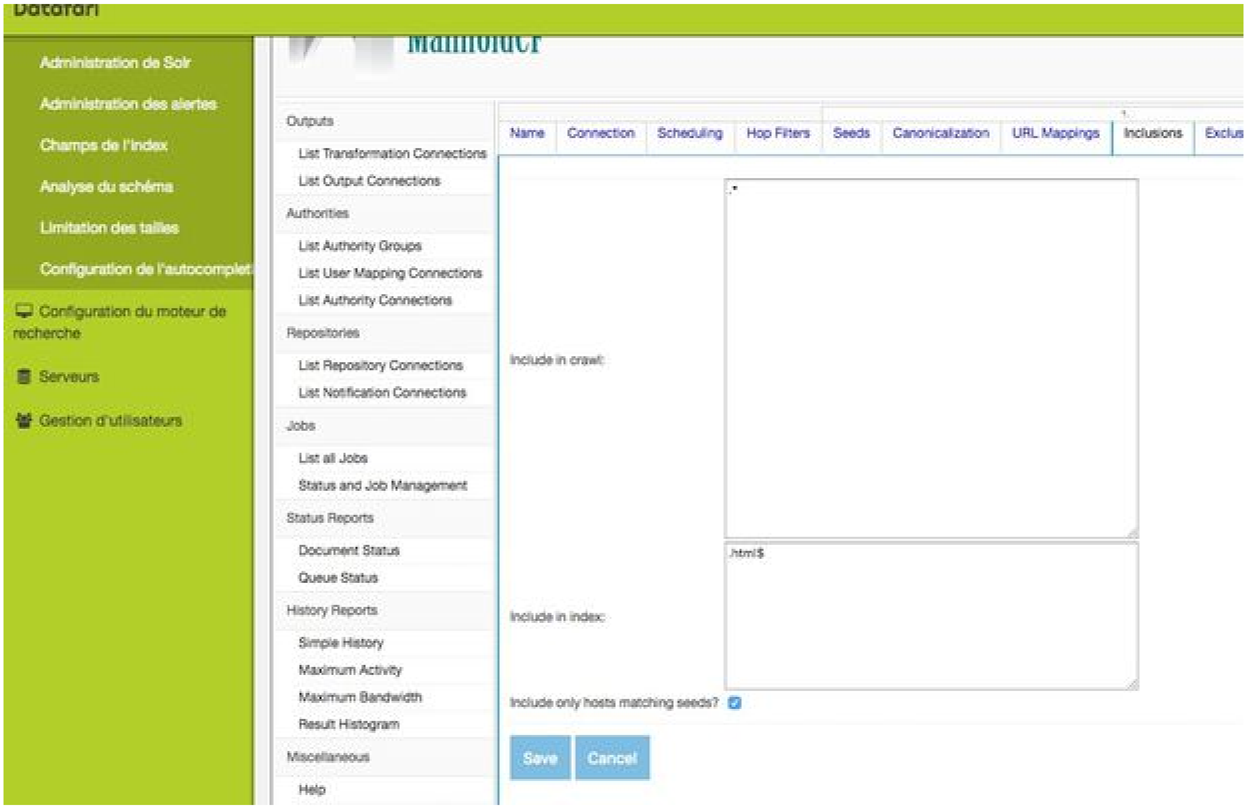


List of Repository Connections

	Name	Description	Connection Type	Authority Group	Max
View Edit Delete	Web		Web	None(globalAuthority)	10

- Add a job connector
  We can now add the job linked to the repository connector and the output connector that we just added.
  Let's see the configuration :
  Name : choose whatever you want. In this example : DatafariWebSite

Connection : choose the repository and the output connection that we just created previously
Seeds : the website that you want to index. Here we enterd http://www.datafari.com
Inclusions : we only want to parse HTML pages so we enter ".*" for the textfield Include in Crawl and ".html$" for the textfield Include in index.



Include in crawl: .*

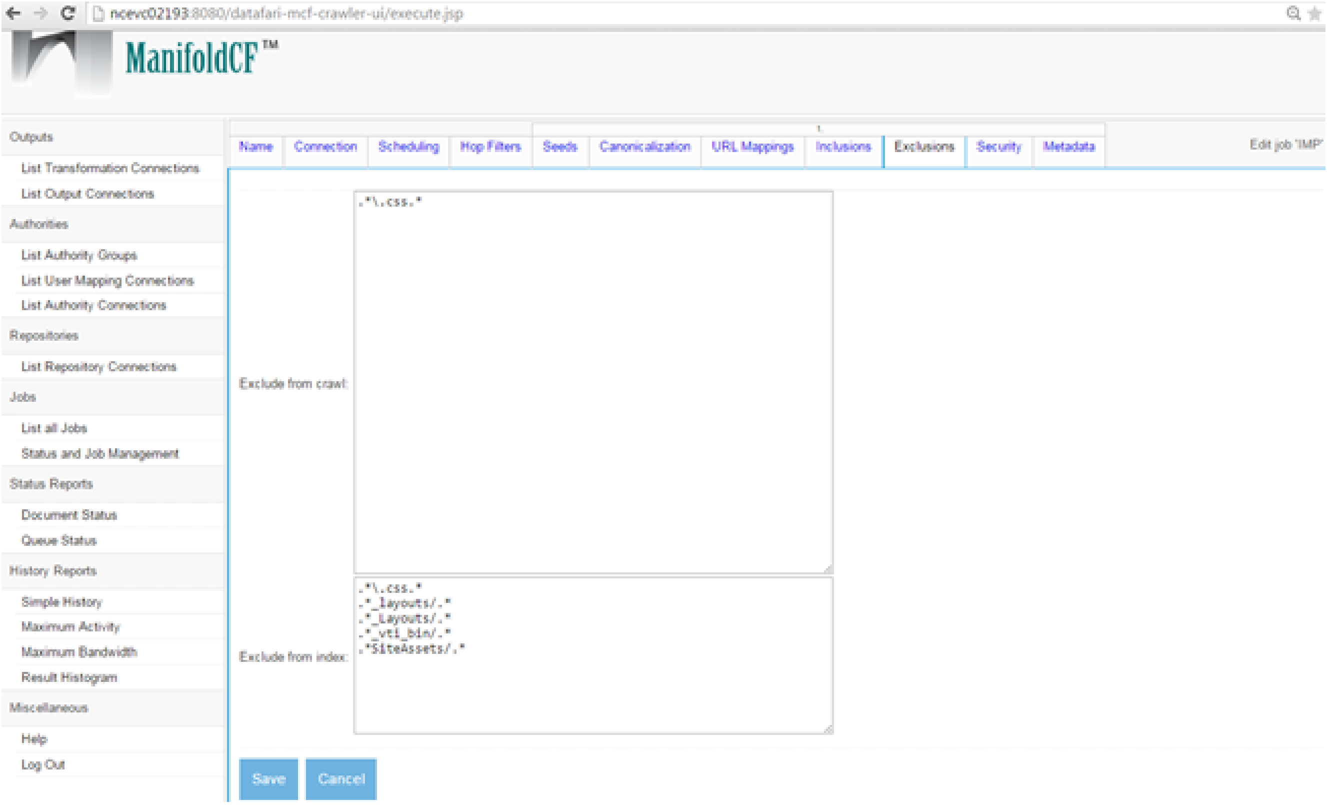
Include in index: .html$

Include only hosts matching seeds? ☑

For the others tab you can let the default parameters.

Here are some further examples for the regex to include/exclude pages, in case you need inspiration 🙂

In the screenshot below, we specify that we do not want URL with CSS extension and we exclude from index all the URL which contain in their path the *layout* directory or *vti bin* or *SiteAssets*. The difference between exclude from crawl and exclude from index is that for example if ManifoldCF finds an URL with the *layouts* folder it will be crawled, then MCF will search into it if it contains other URL to be fetched but this URL will not be included in the Solr index.



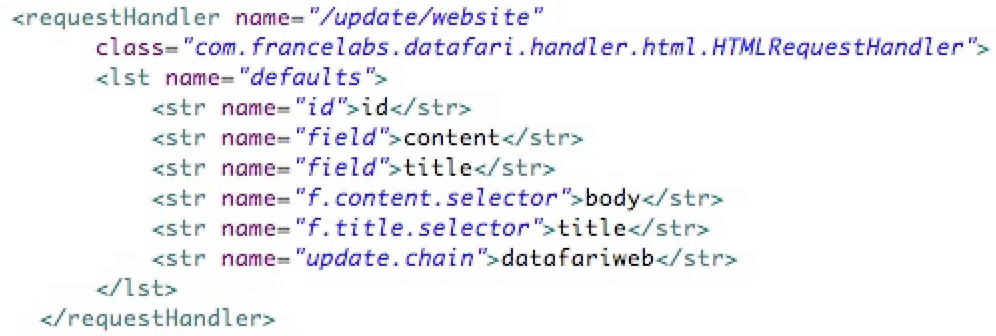
Exclude from crawl:

```
.*\.css.*
```

Exclude from index:

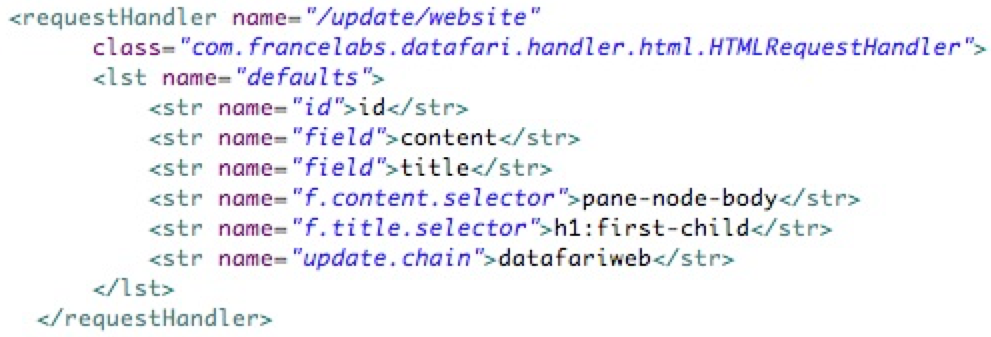
```
.*\.css.*
.*_layouts/.*
.*_Layouts/.*
.*_vti_bin/.*
.*SiteAssets/.*
```

2) Optional : customize the parts of the webpage to be gathered : go to Solr admin page
In the default configuration that we add into Solr, we tell to JSOUP to extract the part in the body tag to the field content and the part that is included in the tag title to the field title.
You can change that in solr_home/FileShare/conf/solrconfig.xml, in the /update/website handler :



```
<requestHandler name="/update/website"
      class="com.francelabs.datafari.handler.html.HTMLRequestHandler">
      <lst name="defaults">
          <str name="id">id</str>
          <str name="field">content</str>
          <str name="field">title</str>
          <str name="f.content.selector">body</str>
          <str name="f.title.selector">title</str>
          <str name="update.chain">datafariweb</str>
      </lst>
  </requestHandler>
```

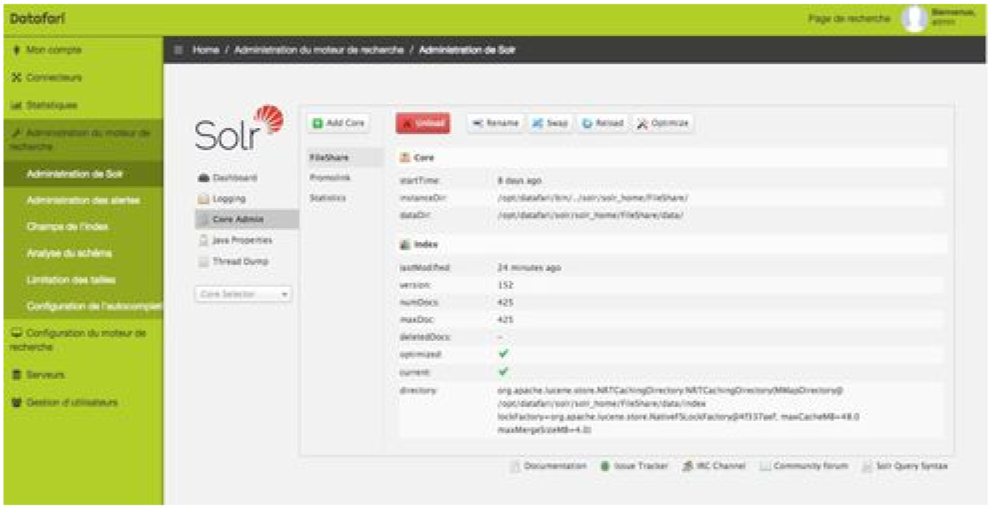
So edit the lines f.content.selector and f.title.selector. Let's say that for the title I want the content of the tag h1:first-child and for content the content of the tag pane-node-body the configuration will be :



```
<requestHandler name="/update/website"
      class="com.francelabs.datafari.handler.html.HTMLRequestHandler">
      <lst name="defaults">
          <str name="id">id</str>
          <str name="field">content</str>
          <str name="field">title</str>
          <str name="f.content.selector">pane-node-body</str>
          <str name="f.title.selector">h1:first-child</str>
          <str name="update.chain">datafariweb</str>
      </lst>
  </requestHandler>
```

After that, you have to relaunch Datafari or to reload the Solr core FileShare :

Click on Core admin and then on the button Reload.



Core

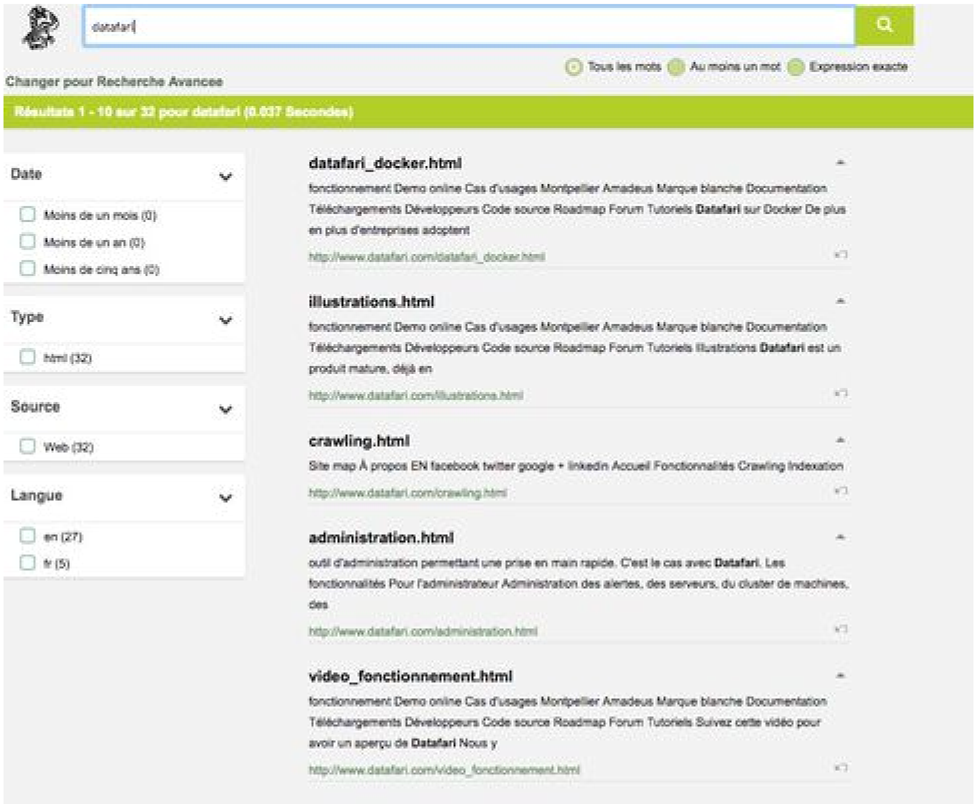
startTime:	8 days ago
instanceDir:	/opt/datafari/bin/../solr/solr_home/FileShare/
dataDir:	/opt/datafari/solr/solr_home/FileShare/data/

Index

lastModified:	24 minutes ago
version:	152
numDocs:	425
maxDoc:	425
deletedDocs:	-
optimized:	✓
current:	✓

directory:	org.apache.lucene.store.NRTCachingDirectory:NRTCachingDirectory(MMapDirectory@
	/opt/datafari/solr/solr_home/FileShare/data/index
	lockFactory=org.apache.lucene.store.NativeFSLockFactory@4f337aef; maxCacheMB=48.0
	maxMergeSizeMB=4.0)

3) Go to MCF and launch the job
Finally you can go back to the MCF admin page and then click on Status dans Job management and then on the button start front of your new job.
You can after that go to to the search page and see your new Solr documents :



Résultats 1 - 10 sur 32 pour datafari (0.037 Secondes)

Date
- ☐ Moins de un mois (0)
- ☐ Moins de un an (0)
- ☐ Moins de cinq ans (0)

Type
- ☐ html (32)

Source
- ☐ Web (32)

Langue
- ☐ en (27)
- ☐ fr (5)

datafari_docker.html
fonctionnement Demo online Cas d'usages Montpellier Amadeus Marque blanche Documentation Téléchargements Développeurs Code source Roadmap Forum Tutoriels **Datafari** sur Docker De plus en plus d'entreprises adoptent
http://www.datafari.com/datafari_docker.html

illustrations.html
fonctionnement Demo online Cas d'usages Montpellier Amadeus Marque blanche Documentation Téléchargements Développeurs Code source Roadmap Forum Tutoriels Illustrations **Datafari** est un produit mature, déjà en
http://www.datafari.com/illustrations.html

crawling.html
Site map À propos EN facebook twitter google + linkedin Accueil Fonctionnalités Crawling Indexation
http://www.datafari.com/crawling.html

administration.html
outil d'administration permettant une prise en main rapide. C'est le cas avec **Datafari**. Les fonctionnalités Pour l'administrateur Administration des alertes, des serveurs, du cluster de machines, des
http://www.datafari.com/administration.html

video_fonctionnement.html
fonctionnement Demo online Cas d'usages Montpellier Amadeus Marque blanche Documentation Téléchargements Développeurs Code source Roadmap Forum Tutoriels Suivez cette vidéo pour avoir un aperçu de **Datafari** Nous y
http://www.datafari.com/video_fonctionnement.html

Be aware that this handler only works for webpages, if you want to index documents like PDF documents, Microsoft Office documents, etc... you have to add an additional job with an other Web repository connector and to choose the standard Output connector. With this configuration Tika will be used to extract the content of the documents and add the associated metadata. In the job configuration, you have to exclude the webpages that are indexed by your "JSOUP job".

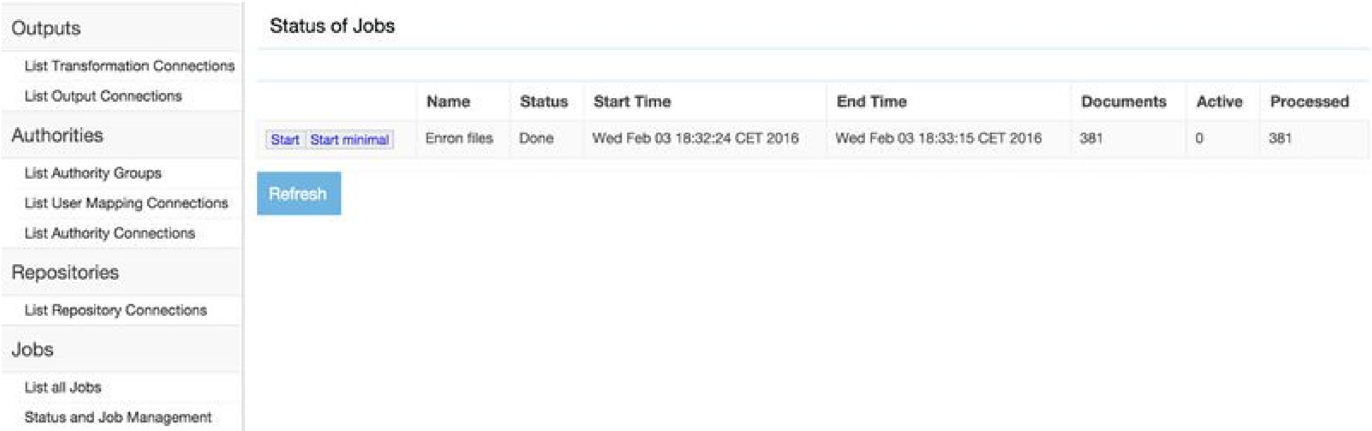
Reset a MCF job

In this case, we assume that you indexed some data with a MCF job and you want to delete the indexed data before restarting the MCF job.

So in order to do it, the steps are :

- Make sure that your job is in "Done" mode :

Go to Jobs -> Status and Job management


Outputs
List Transformation Connections
List Output Connections
Authorities
List Authority Groups
List User Mapping Connections
List Authority Connections
Repositories
List Repository Connections
Jobs
List all Jobs
Status and Job Management
Refresh

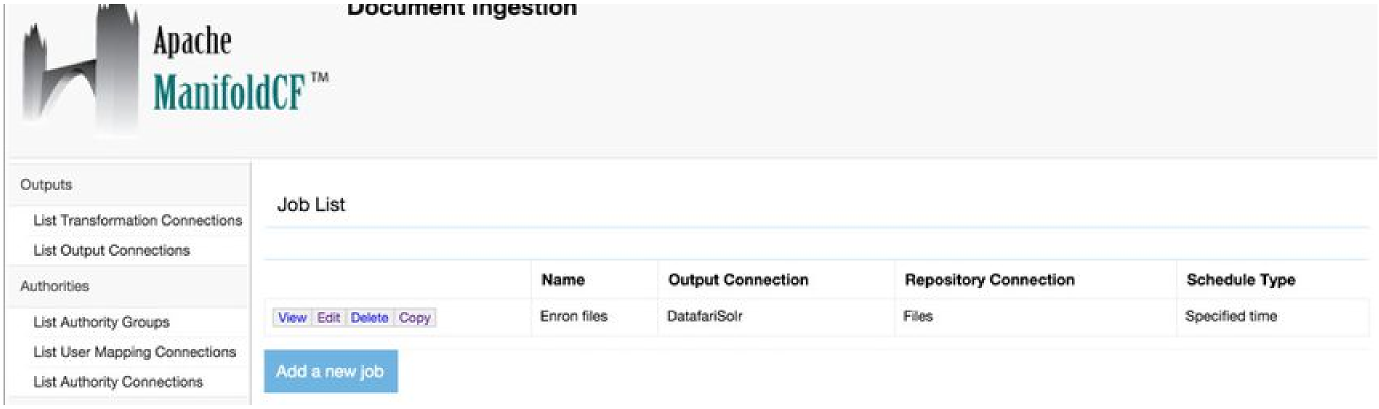
Status of Jobs

	Name	Status	Start Time	End Time	Documents	Active	Processed
Start | Start minimal	Enron files	Done	Wed Feb 03 18:32:24 CET 2016	Wed Feb 03 18:33:15 CET 2016	381	0	381

- Copy your existing job :

Go to Jobs -> List all Jobs

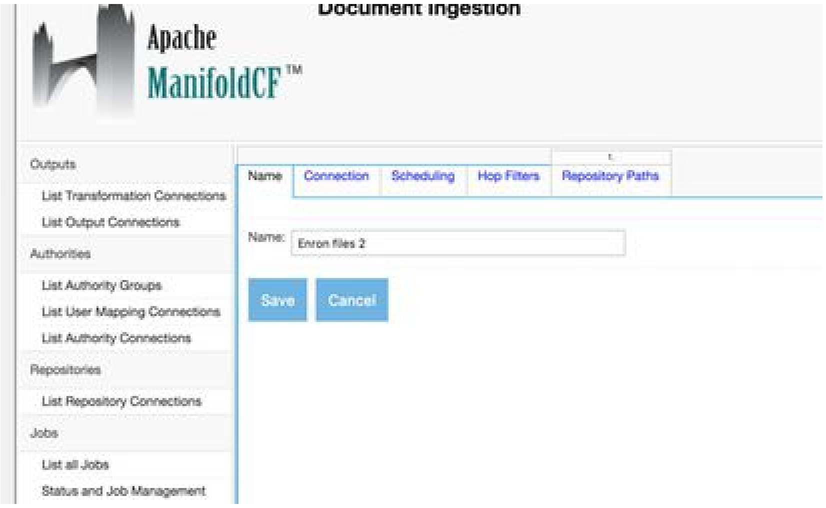
Then click on Copy next to the job name



Job List

	Name	Output Connection	Repository Connection	Schedule Type
View | Edit | Delete | Copy	Enron files	DatafariSolr	Files	Specified time

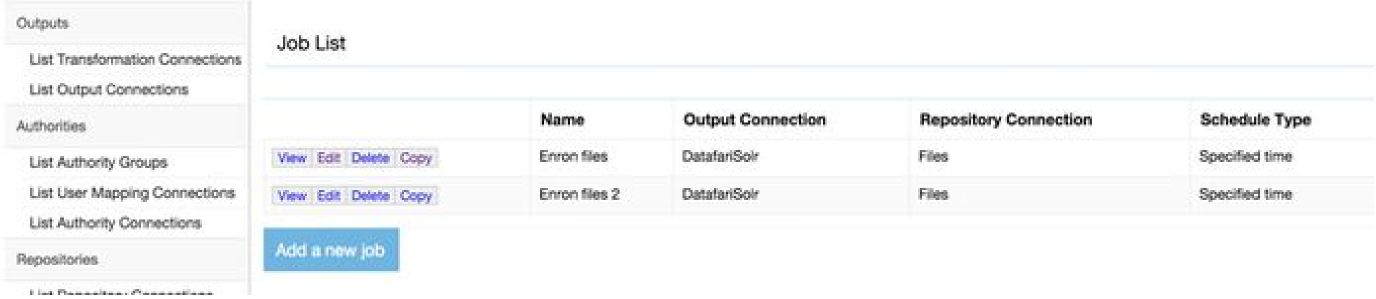
In the next screen, change the job name, here we add "2" at the end of the name then click on the Save button.



Name: Enron files 2

- Go back to the List jobs menu

So click on Jobs -> List all jobs.



Job List

	Name	Output Connection	Repository Connection	Schedule Type
View Edit Delete Copy	Enron files	DatafariSolr	Files	Specified time
View Edit Delete Copy	Enron files 2	DatafariSolr	Files	Specified time

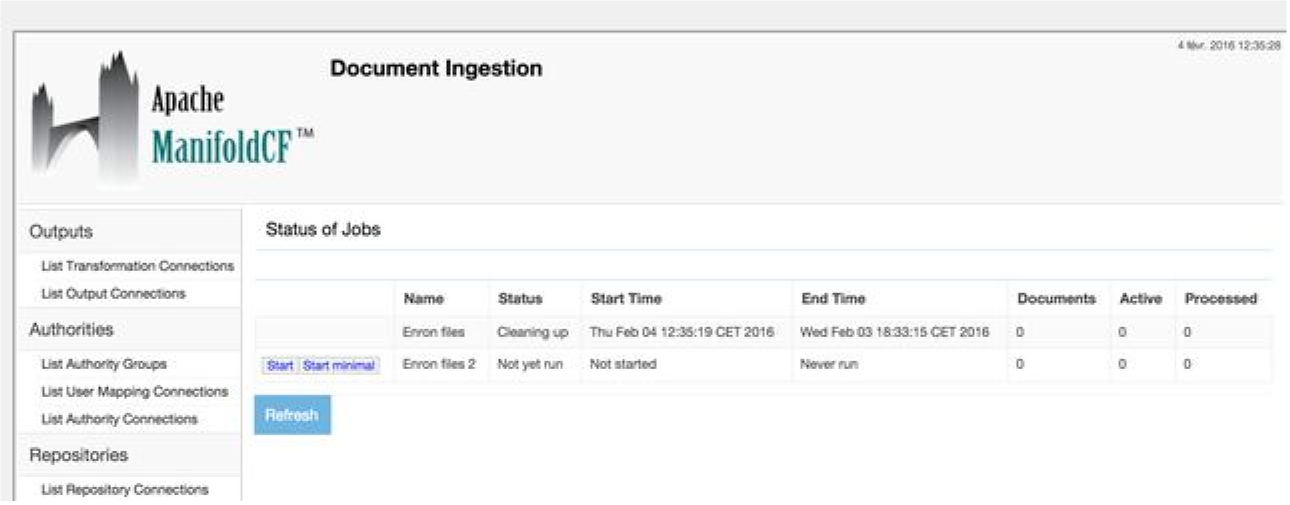
Now we have the 2 jobs present on the List all jobs menu. We now can delete the original job.

- Delete the original job

Click on the Delete button front of the "old" job, so here Enron files. Confirm your choice on the popup.

- Go to the Status and job management

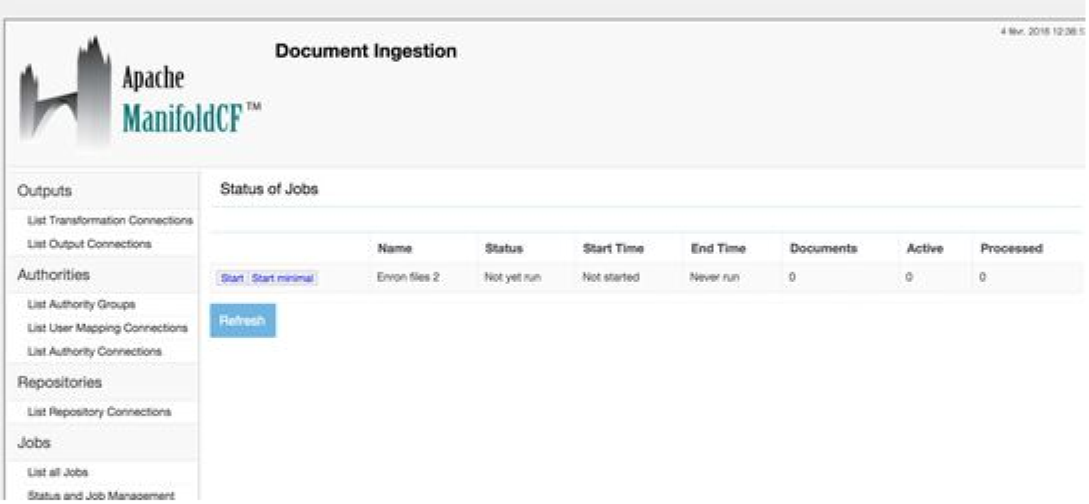
Click on Jobs -> Status and Job management. You will see that the Enron files job status is cleaning up. It means that MCF gives the order to Solr to delete all the documents crawled per this job.



Status of Jobs

	Name	Status	Start Time	End Time	Documents	Active	Processed
	Enron files	Cleaning up	Thu Feb 04 12:35:19 CET 2016	Wed Feb 03 18:33:15 CET 2016	0	0	0
Start Start minimal	Enron files 2	Not yet run	Not started	Never run	0	0	0

Wait a little and click periodically on the resfresh button until the cleaning process ends. At the end, only the new job is present.


Apache ManifoldCF™
Document Ingestion
Outputs
List Transformation Connections
List Output Connections
Authorities
List Authority Groups
List User Mapping Connections
List Authority Connections
Repositories
List Repository Connections
Jobs
List all Jobs
Status and Job Management
Refresh

Status of Jobs

	Name	Status	Start Time	End Time	Documents	Active	Processed
Start Start minimal	Enron files 2	Not yet run	Not started	Never run	0	0	0

- Start the new job

Finally you can launch your new job. Just click on the start button next to it and the data will be crawled and indexed into Solr.

Deduplication management

BETA Version: Instable as of 2.2.1

Administration:

To enable or disable this functionality, we have not done the user interface for now (as of Datafari 1.2). You need to open the datafari.properties file, which is available in the folder \[Datafari_install_Folder]\bin .

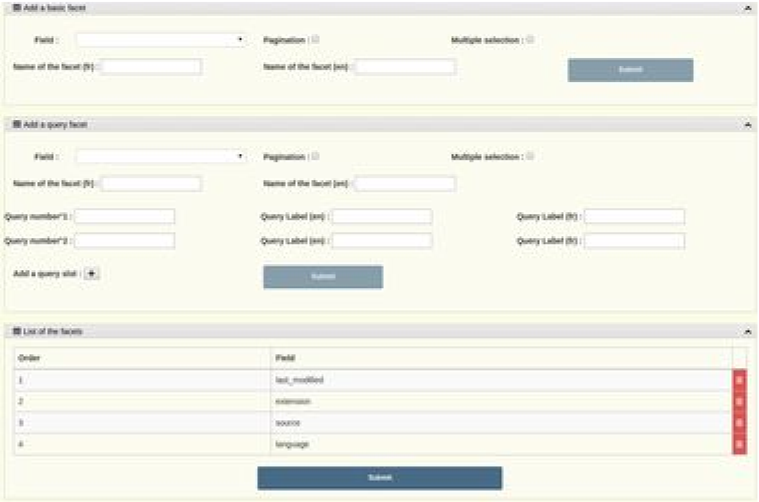
In this file, set the parameter to DEDUPLICATION_IS_ENABLED=true or false depending on your needs. By default, it is set to true as it is resource consuming. Note that you need to restart Datafari in order for the change to be taken into account.

If you need more technical information, please refer to Deduplication : Technical note

Facets management

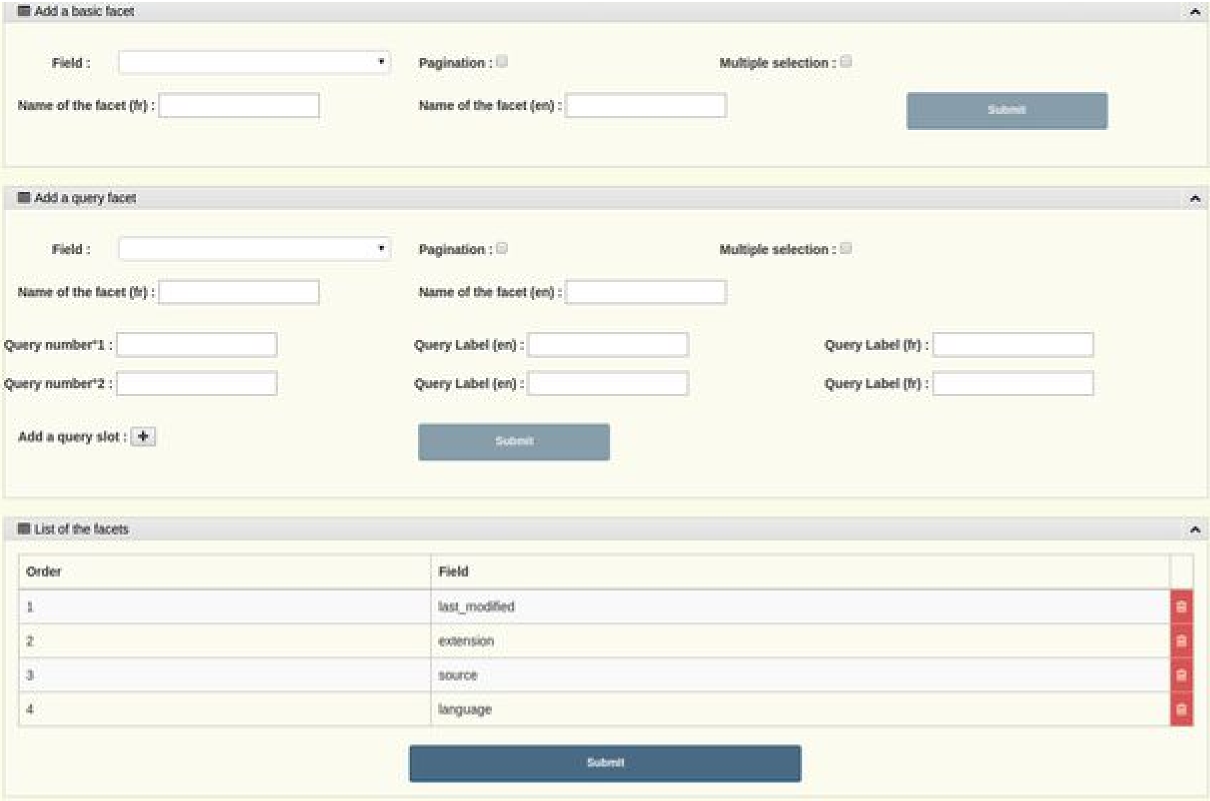
BETA version, Instable as of version 2.2.1

From the admin interface, you can create, delete, or modify the order of appearance of the facets displayed in the searchview.



Order	Field
1	last_modified
2	extension
3	source
4	language

For that, you need to have the search expert or search administrator role to access the admin interface.



Order	Field
1	last_modified
2	extension
3	source
4	language

There are two distinct boxes to add a facet:

- **"Add a basic facet" box**: The first one is to add simple facets. The only mandatory form field is the Solr field. You can optionally:
    - set the pagination to true,
    - allow the user to select multiple elements of this facet,
    - customize the name of this facet.
- **"Add a query facet" box**: The second one is for the query facet. It adds custom queries, that you will have to fill, at least one, otherwise it makes no sense, customize the name of the request.
    - If you don't customize the name, they will be called by the name of the field which is not explicit.
    - If you click on the “*add a query slot*” button, it will add another field for a query and for its query label. In this context, query label is the label that will be printed next to the checkbox in the search UI. Note that if the query slot is empty, Datafari will not take into account the field dedicated to this query label. Also, if no query is declared, Datafari will ignore the request. We do not do validity checks on the queries, so it is your responsibility to ensure they are correct. If they are invalid, it is highly probable that the global search will not work anymore. But you can still delete them by clicking on the trash at the end of the line of the bottom table.

> Note that you can neither have two facets on a field, nor can you have a basic facet and a query facet on a single field. Thus the fields that already faceted are removed from the selection

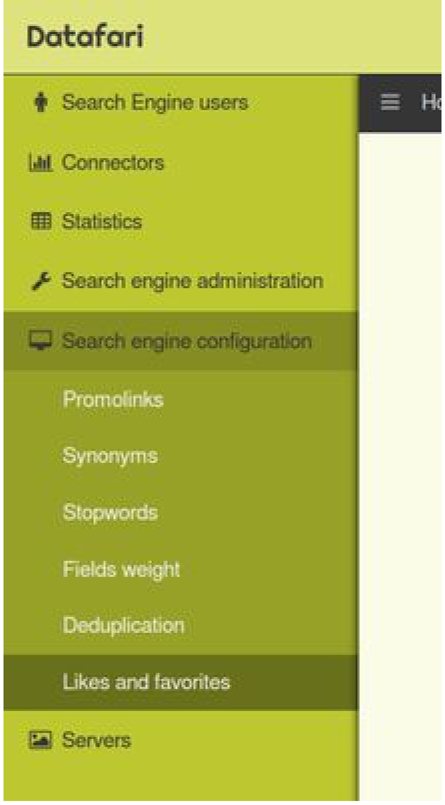
The **"List of the facets" box** allows to do the following on existing facets:

- Deletion, by clicking the trash icon on the right
- Change the printing order of the facet. To change it you can drag and drop lines of the table and once the order fits your needs, click the submit button.

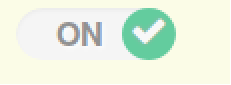
Favorites management

Especially useful for librarians or researchers, the Favorites functionality allows users to save documents that appear in the result.

This functionality is disabled by default. To enable it, to login to the admin section. Browse to “Search Engine Configuration” and then click on “Likes and favorites” as shown in the image at the left.



Now click on the switcher to activate or deactivate the functionality. If the switcher displays the “OFF” statement, it means that the functionality is deactivated.



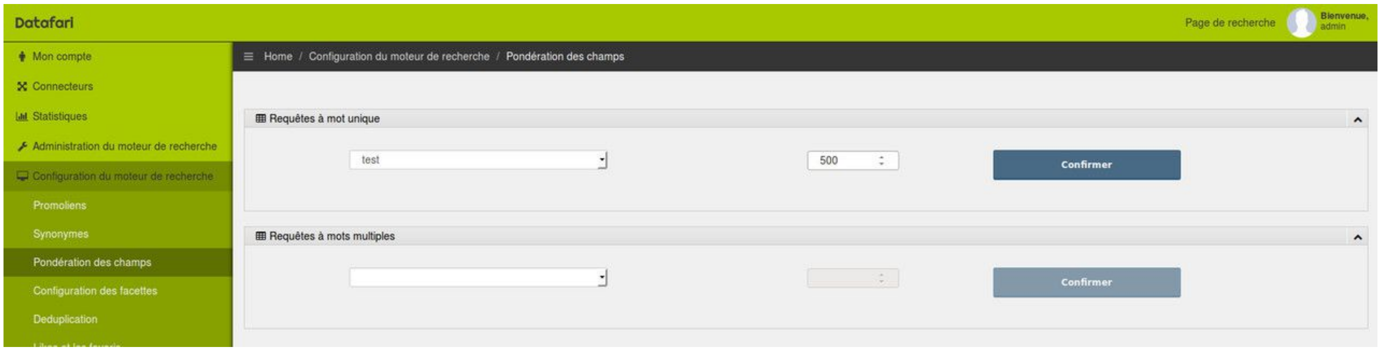
If you click on it, it will be activated. The switcher should display the “ON” statement.

Fieldweights

Every modification you do with this UI will be saved in a custom searchHandler which is located in the {Datafari_home}/solr/solr_home/FileShare/conf/customs_solrconfig/custom_search_handler.incl, and the original searchHandler from the standard solrconfig.xml file will be (if not already) commented. Thanks to this behavior you can "rollback" your modifications and re-apply the default settings by uncommenting the original searchHandler in {Datafari_home}/solr/solr_home/FileShare/conf/solrconfig.xml and removing the custom one in {Datafari_home}/solr/solr_home/FileShare/conf/customs_solrconfig/custom_search_handler.incl

This admin UI allows you to modify the weight of any field, in order to optimize the persistence of Datafari regarding your documents.

There are two kinds of weights: those for unique word search and those for multiple words search. So, depending on the kind of weight you want to modify, select your desired field in the corresponding list. The current weight of the field will be displayed in the text area and you will be free to set a new value.


Datafari
Page de recherche
Bienvenue,
admin
Mon compte
Connecteurs
Statistiques
Administration du moteur de recherche
Configuration du moteur de recherche
Promoliens
Synonymes
Pondération des champs
Configuration des facettes
Deduplication
Home / Configuration du moteur de recherche / Pondération des champs
Confirmer
Confirmer

Requêtes à mot unique

test

500

Requêtes à mots multiples

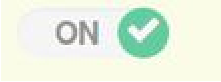
Once you are satisfied with the new value, click on the "Confirm" button.

> Notice that in order to take your modifications into account, you will need to reload the Solr Core conf through the Solr admin UI or through a restart.

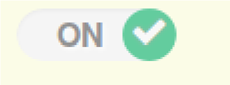
Likes management

Inspired from social networks, the Likes functionality allows users to like a search result. This may be used as part of the relevancy algorithm. In terms of administration, the search expert can decide to activate or deactivate the functionality.

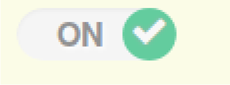
This functionality is disabled by default. To enable it, you have to log into the admin section. Browse to “Search Engine Configuration” and then click on “Likes and favorites” as shown in the image at the left.



Now click on the switcher to activate or deactivate the functionality. If the switcher displays the “OFF” statement, it means that the functionality is deactivated.



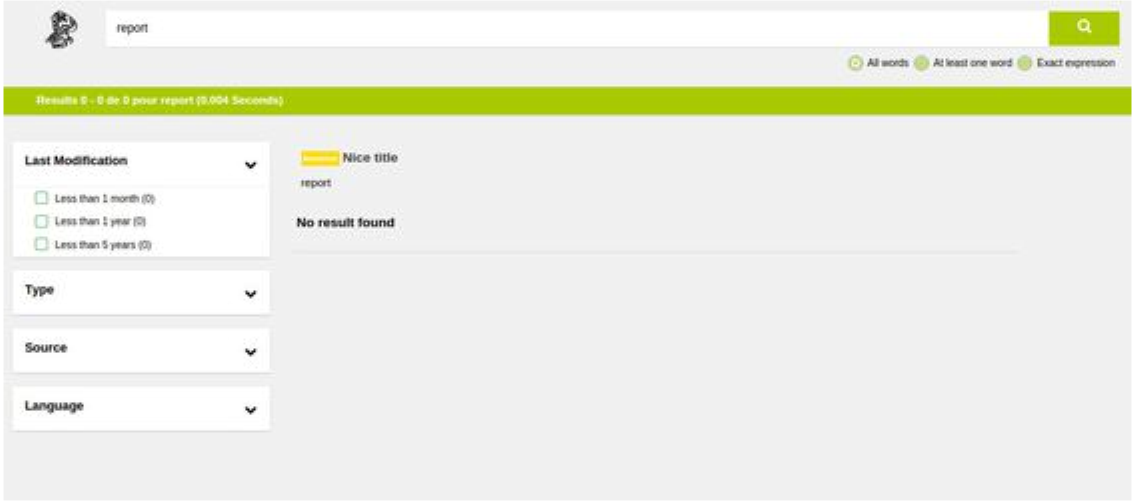
If you click on it, it will be activated. The switcher should display the “ON” statement.



Promolinks management

Starting with Datafari v1.2, promolinks can be conceptually assimilated to google adwords: independently from the indexed content, it allows the search expert to display text in a dedicated place on the screen (usually on top or on the side). This display is triggered by the presence of specific terms in the search query.

For instance, the illustration below shows a promolink entitled "Nice title" which has the content “report” and is displayed because the keyword "report" was used in the search query::



report

○ All words ○ At least one word ○ Exact expression

Results 0 - 0 de 0 pour report (0.004 Seconds)

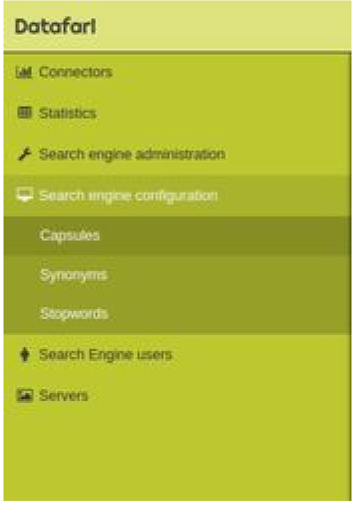
Nice title

report

No result found

In order to create Promolinks, you need to have the search expert role.

You have access to a dedicated web interface in the admin UI. First, go to the administration menu, click on Search Engine Configuration and select Capsule in the dropdown list.



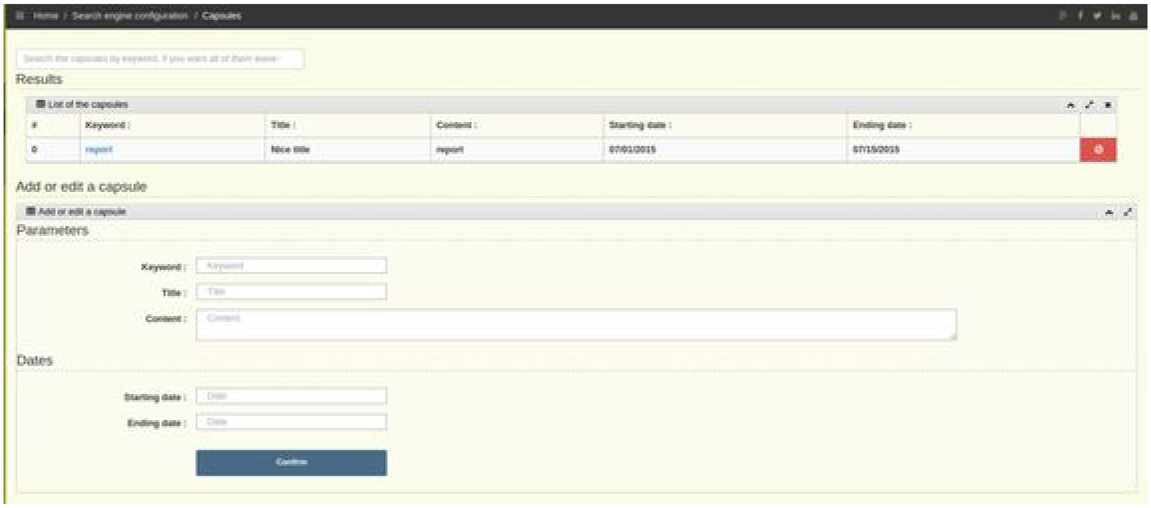
You then have access to the promolink admin UI, which shows the list of all existing promolinks. You can do the following operations on the existing promolinks:

- Delete by clicking the red button,
- Edit by clicking the keyword,

Using the search bar on top, you can filter out the results by typing in something and pressing enter.

To add a promolink, use the form at the bottom of the page, and fill in the fields to add a promolink.

Note that the dates are optional, that you may fill only one out of those fields, and that if you choose to fill both, the starting date must be prior to then ending date. If dates have been filled during the add, Datafari will check if, when a search is made on the matching keyword, the current date is posterior to the starting date and prior to the ending date


Home / Search engine configuration / Capsules
Search the capsules by keyword, if you want all of them leave
Results
List of the capsules

#	Keyword :	Title :	Content :	Starting date :	Ending date :	
0	report	Nice title	report	07/01/2015	07/15/2015	

Query Elevator

The version 2.3 of Datafari comes with the new "Query Elevator" feature. It allows users with the "SearchExpert" role to elevate documents for specific queries, in order to make them appear in the top of the results list.

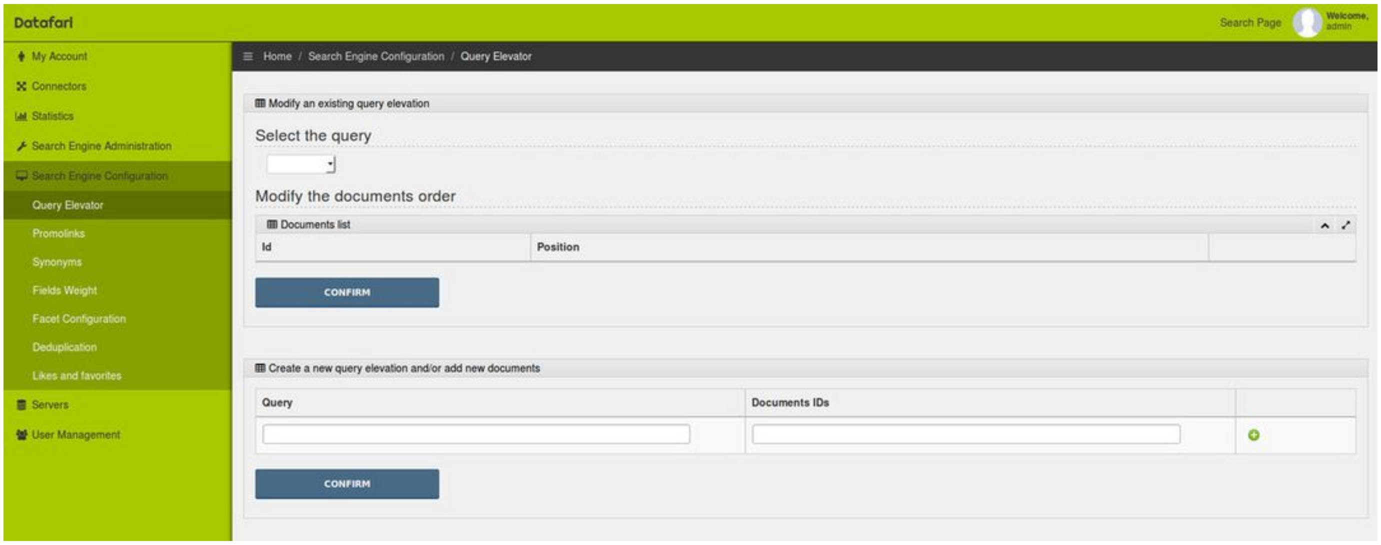
Two possibilities are provided to elevate documents:

- Directly from the search view
- Thanks to the admin UI

Admin UI feature

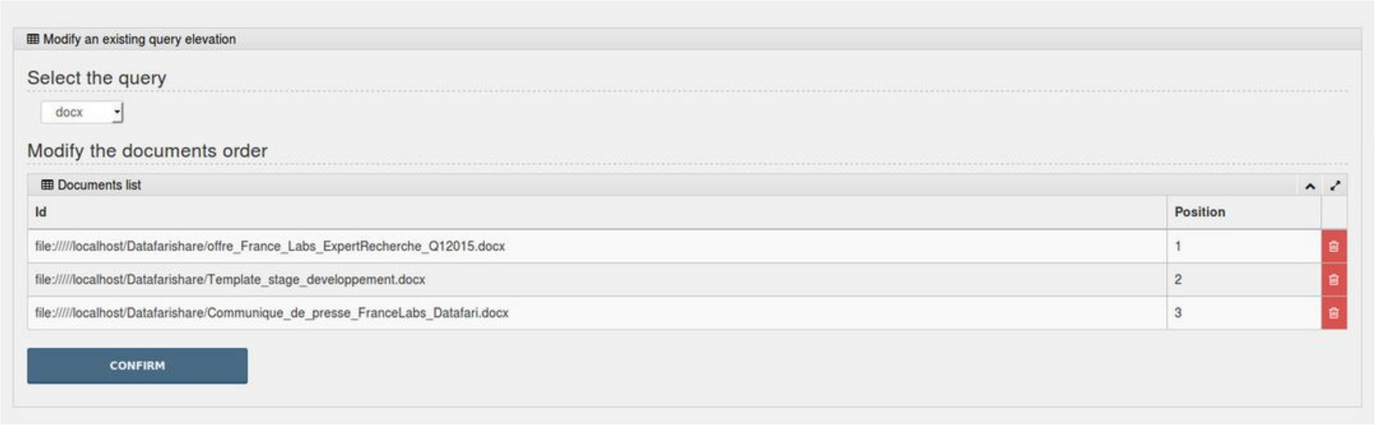
> In order to access the admin UI described bellow, you will need to be authenticated in Datafari as a user with the "SearchExpert" or the "SearchAdministrator" privilege. For more details refer to the Types of users.

From the 2.3 version of Datafari, a new feature is available for the "SearchExpert" and "SearchAdministrator" users in the admin UI, the "Query Elevator" :



This UI allows you to either modify/remove existing elevate boosts, or create new ones.

Let's start with the modify/remove part :



Select the query

docx

Modify the documents order

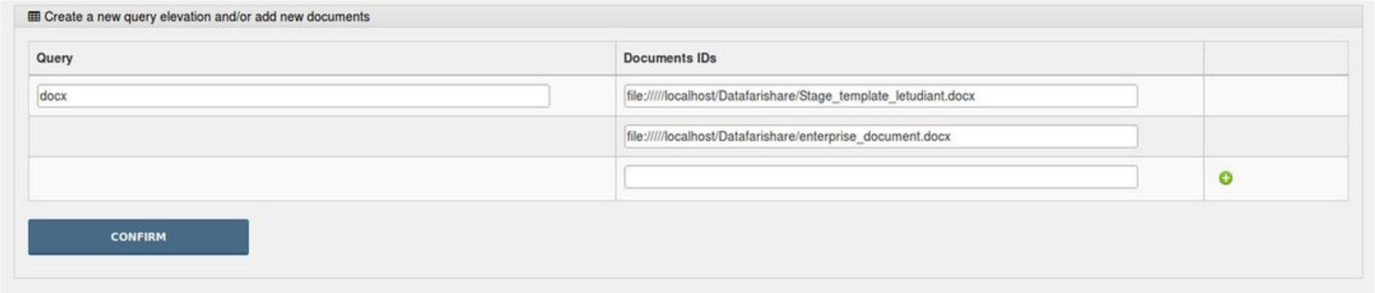
Id	Position
file://///localhost/Datafarishare/offre_France_Labs_ExpertRecherche_Q12015.docx	1
file://///localhost/Datafarishare/Template_stage_developpement.docx	2
file://///localhost/Datafarishare/Communique_de_presse_FranceLabs_Datafari.docx	3

As you can see, it is composed by two sections :

- The first one is a "Select" where you will be able to select a query which has elevated documents. The list only contains queries that are present in the elevate.xml file located in [Datafari_Home]/solr/solr_home/FileShare/conf.
- The second is a table where will appear the documents which have an elevate boost for the query you have selected.
  The documents are ordered by priority of appearance in the search results, which is also indicated by their "Position" number.
  Here you can simply drag and drop the documents to change their position in the search results. You can also remove some of them if you want by clicking on their associated trash icon.

Once you are satisfied with the order of the documents for the selected query, you can click on the "confirm" button. Datafari will update the elevate.xml file to match your configuration and will reload the Solr core so your modifications will take effect at the end of the process. Once the Solr core is reloaded, a confirmation message will appear next to the "confirm" button.

Let's focus now on the create part:



Query	Documents IDs
docx	file://///localhost/Datafarishare/Stage_template_letudiant.docx
	file://///localhost/Datafarishare/enterprise_document.docx

This part allow you to create or add elevate boosts for a query. Notice that if you add new elevate boosts to a query which already has ones, the new ones will be added at the top of the list in the same order as you have entered them.

Simply type a query in the corresponding text field and then enter the id of the documents you want to elevate for this query. You can add several documents by clicking on the green cross which will add a new document ID line.

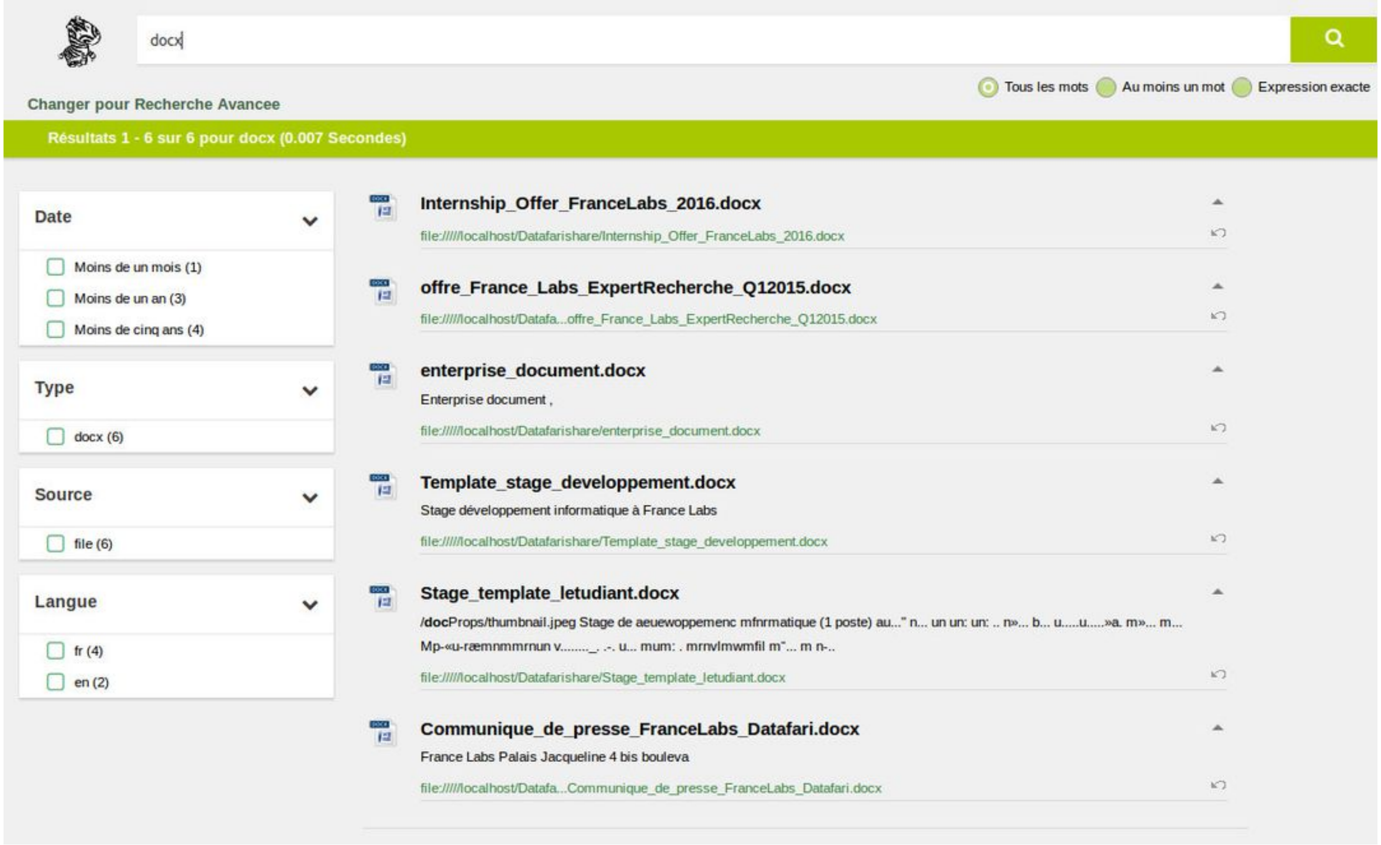
Once your are done with your configuration, click on the "confirm" button. Datafari will update the elevate.xml file to match your configuration and will reload the Solr core so your modifications will take effect at the end of the process. Once the Solr core is reloaded, a confirmation message will appear next to the "confirm" button.

Search View Feature

NB : The feature is temporarily unavailable for Datafari 3.0. It will be integrated again in Datafari 3.2.

> To see the feature described bellow in the search view of Datafari, you must be connected to Datafari with a user having either the "SearchExpert" role, or the "SearchAdministrator" role. Refer to the Types of users for more details.

Once you are connected as a "SearchExpert" or a "SearchAdministrator", if you perform a search in Datafari, you will notice two news icons at the right of each result:


docx
● Tous les mots ○ Au moins un mot ○ Expression exacte
Changer pour Recherche Avancee

Résultats 1 - 6 sur 6 pour docx (0.007 Secondes)

Date
□ Moins de un mois (1)
□ Moins de un an (3)
□ Moins de cinq ans (4)

Type
□ docx (6)

Source
□ file (6)

Langue
□ fr (4)
□ en (2)

Internship_Offer_FranceLabs_2016.docx
file://///localhost/Datafarishare/Internship_Offer_FranceLabs_2016.docx

offre_France_Labs_ExpertRecherche_Q12015.docx
file://///localhost/Datafa...offre_France_Labs_ExpertRecherche_Q12015.docx

enterprise_document.docx
Enterprise document ,
file://///localhost/Datafarishare/enterprise_document.docx

Template_stage_developpement.docx
Stage développement informatique à France Labs
file://///localhost/Datafarishare/Template_stage_developpement.docx

Stage_template_letudiant.docx
/**doc**Props/thumbnail.jpeg Stage de aeuewoppemenc mfnrmatique (1 poste) au..." n... un un: un: .. n»... b... u.....u.....»a. m»... m... Mp-«u-ræmnmmrnun v........_. .-. u... mum: . mrnvlmwmfil m"... m n-..
file://///localhost/Datafarishare/Stage_template_letudiant.docx

Communique_de_presse_FranceLabs_Datafari.docx
France Labs Palais Jacqueline 4 bis bouleva
file://///localhost/Datafa...Communique_de_presse_FranceLabs_Datafari.docx

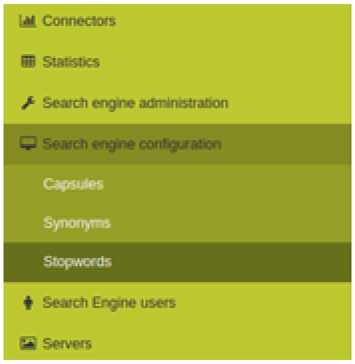
These icons are used to elevate or remove an elevate boost of the associated document for the current query:

- ▲ : Is used to elevate the associated document. It will add a boost to the document for the current query, that will make it appear as the first result of the list.
  When you click on this button, Datafari modifies the elevate.xml file located in [Datafari_Home]/solr/solr_home/FileShare/conf to add or move the document as the first of the list and then restart the Solr core. This is the reason why you will have a delay before the search view is reloaded and the document appears as the first of the results.
  If you elevate several documents with this method, notice that it will always be the last elevated doc which will be on top of the results list. It works like a stack where the last added element is added on top of the stack so it has the priority on the previous inserted one which also has the priority on the previous etc etc.
  This feature is intended to quickly and easily elevate only one or very few documents. As on each click the Solr core is reloaded and that it may be annoying to find the doc you want to elevate or to wait after each click to elevate several docs, you will prefer to manage the elevate configuration through the Admin UI feature.
- ⟲ : Is used to remove the elevate boost of the associated document for the current query. If the document has an entry for the current query in the elevate.xml file located in [Datafari_Home]/solr/solr_home/FileShare/conf, it will be removed from the list of elevated documents and the Solr core reloaded.
  As for the elevate feature, this search view incorporated feature is intended to quickly and easily remove elevate boost on only one or very few documents. For better control and ergonomic interface, you will prefer the Admin UI feature.

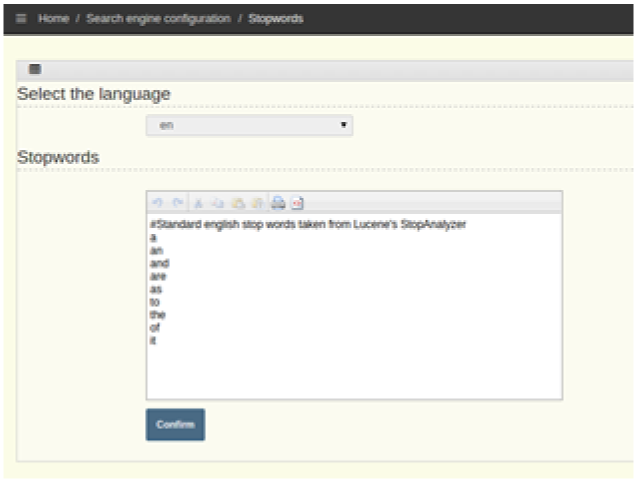
Stopwords management

In order to create Stopwords, you need to be connected with the search expert role.

Go to the administration interface. Then, go to the administration left menu, click on Search Engine Configuration and select Stopwords in the dropdown list.



Stopwords are language specific. You need to select the language for which you want to manage the stopwords. Once this is done, you get a text window allowing you to edit the stopwords list. Note that only one search expert at a given time can edit this file. Any other simultaneous tentative will end up with an error message on the screen.



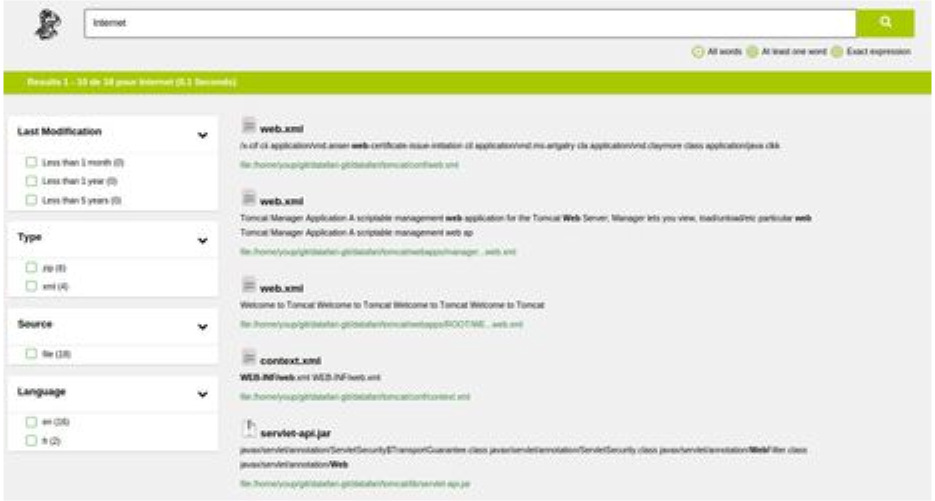
```
#Standard english stop words taken from Lucene's StopAnalyzer
a
an
and
are
as
to
the
of
it
```

When editing the text file, please respect the format used by the Solr 5.1 for its stopwords file :

```
#Standard english stop words taken from Lucene's StopAnalyzer
a
an
and
are
as
at
be
but
by
for
if
in
into
is
it
```

Synonyms

Synonyms are a way to create a link between words. If a word A is declared as a synonym of word B, then any search for word B will end up looking also for word A. This allows for instance, to link the words *internet* and *web*, which have an identical meaning in a given context, but have a totally dissimilar spelling. Web is a good exemple of the usefulness of giving “directions” to your synonyms. You may want searches for internet to look for documents with web in it as well, but you may not want searches for web to look for documents with internet in it, in case you are looking for spider webs. Here you can see that the term “Internet” is linked with the word “web”



Results 1 - 10 de 18 pour Internet (0.1 Seconds)

web.xml
/x-cif cli application/vnd.anser **web**-certificate-issue-initiation cil application/vnd.ms-artgalry cla application/vnd.claymore class application/java clkk
file:/home/youp/git/datafari-git/datafari/tomcat/conf/web.xml

web.xml
Tomcat Manager Application A scriptable management **web** application for the Tomcat **Web** Server; Manager lets you view, load/unload/etc particular **web**
Tomcat Manager Application A scriptable management web ap
file:/home/youp/git/datafari-git/datafari/tomcat/webapps/manager...web.xml

web.xml
Welcome to Tomcat Welcome to Tomcat Welcome to Tomcat Welcome to Tomcat
file:/home/youp/git/datafari-git/datafari/tomcat/webapps/ROOT/WE...web.xml

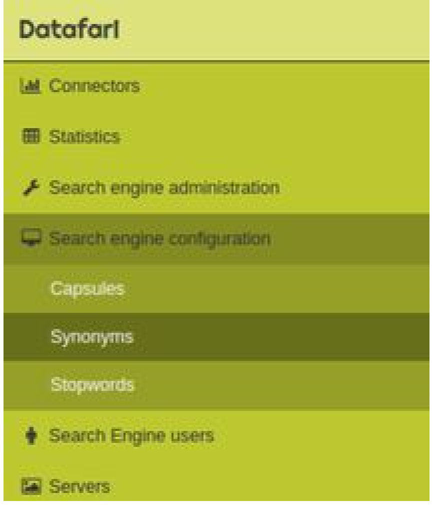
context.xml
WEB-INF/web.xml WEB-INF/web.xml
file:/home/youp/git/datafari-git/datafari/tomcat/conf/context.xml

servlet-api.jar
javax/servlet/annotation/ServletSecurity$TransportGuarantee.class javax/servlet/annotation/ServletSecurity.class javax/servlet/annotation/**Web**Filter.class
javax/servlet/annotation/**Web**
file:/home/youp/git/datafari-git/datafari/tomcat/lib/servlet-api.jar

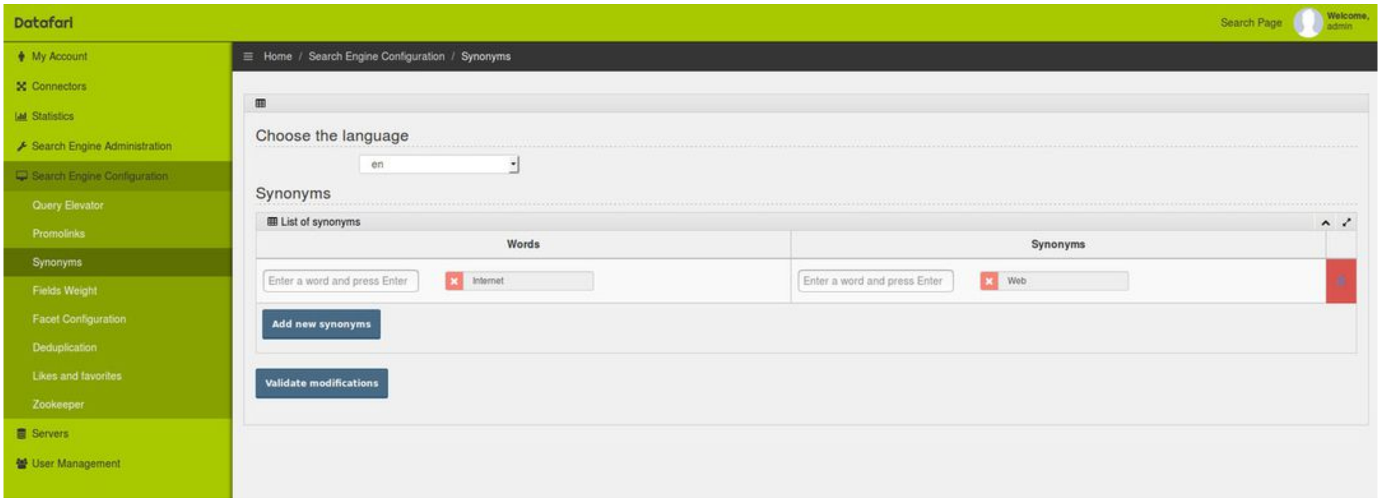
Search Expert: managing synonyms

In order to create Synonyms, you need to be connected with the search expert role.

Once in the administration interface, go to the administration menu, click on Search Engine Configuration and select Synonyms in the dropdown list.



Synonyms are language specific. You need to select the language for which you want to manage the synonyms. Once this is done, you get a nice interface allowing you to edit the synonyms list. Note that only one search expert at a given time can edit this file. Any other simultaneous tentative will end up with an error message on the screen.

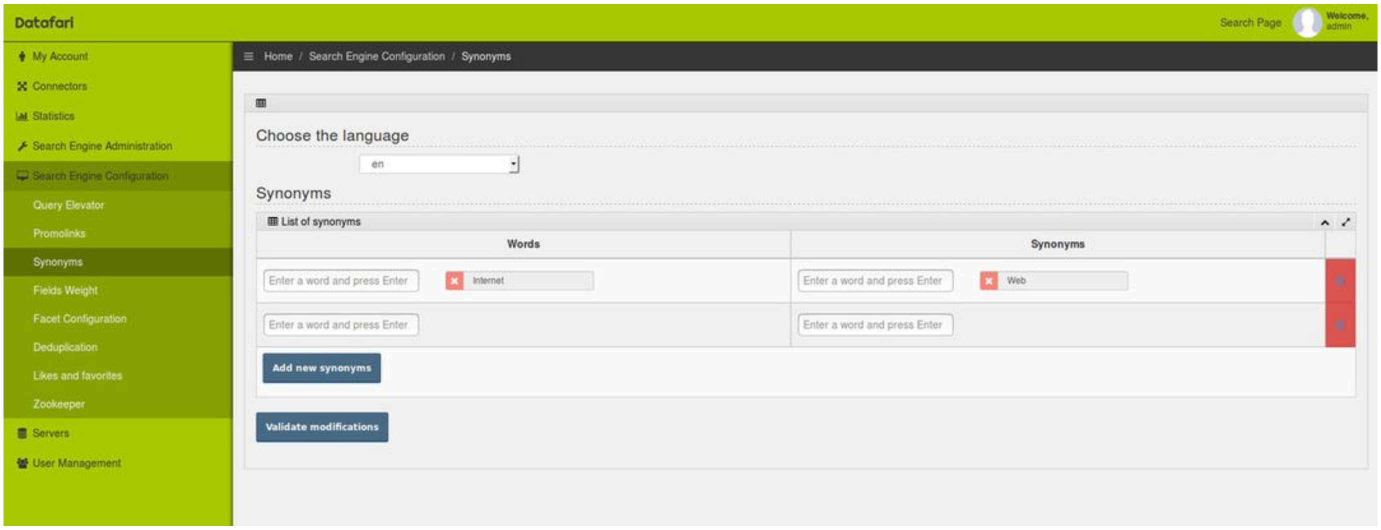


Choose the language: en

Words	Synonyms
Enter a word and press Enter ☒ Internet	Enter a word and press Enter ☒ Web

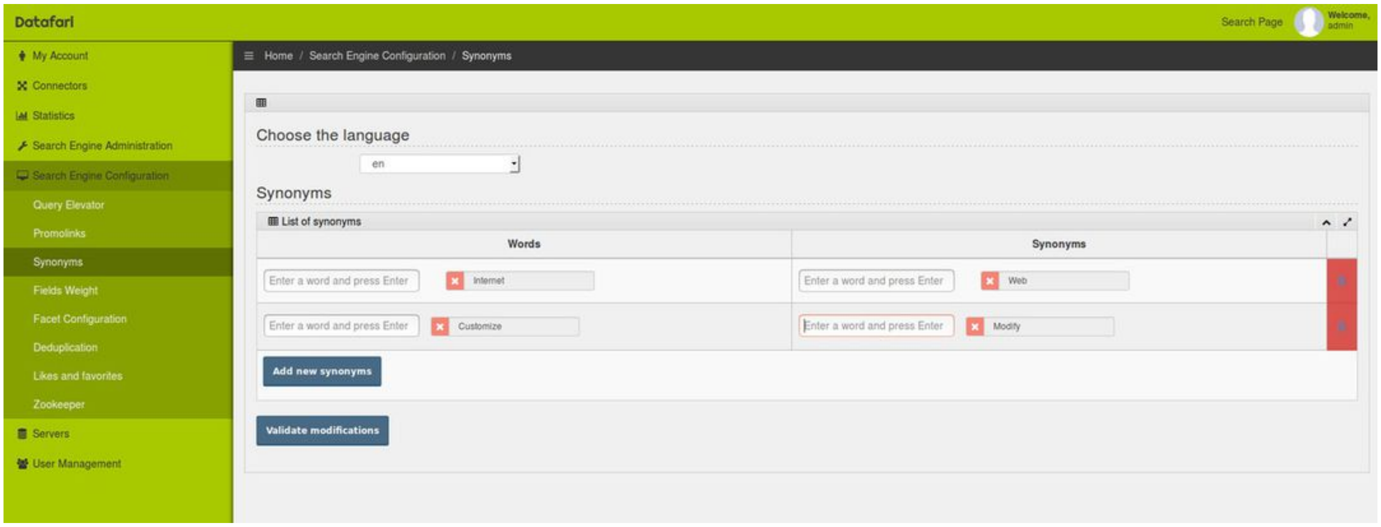
Here you can delete/add words and add/delete their corresponding synonyms or even delete a synonym entry by clicking on the 'trash' button at the end of a line.

To add new synonym entry, simply click on the 'Add new synonyms' button, a new line will be created and you will be able to edit this new entry:



Choose the language: en

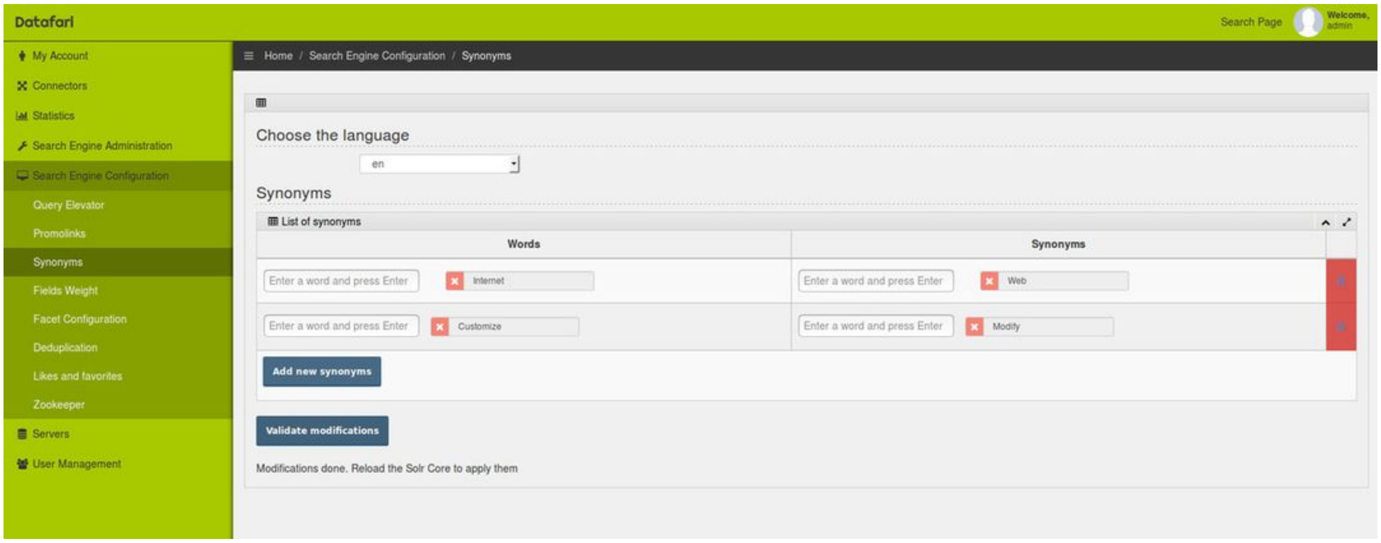
Words	Synonyms
Enter a word and press Enter ☒ Internet	Enter a word and press Enter ☒ Web
Enter a word and press Enter	Enter a word and press Enter



Choose the language: en

Words	Synonyms
Enter a word and press Enter ☒ Internet	Enter a word and press Enter ☒ Web
Enter a word and press Enter ☒ Customize	Enter a word and press Enter ☒ Modify

Once you are ok with your modifications, click on the 'Validate modifications'. If everything is ok you will be noticed by a message to reload the Solr core in order to apply the new configuration:



Choose the language

en

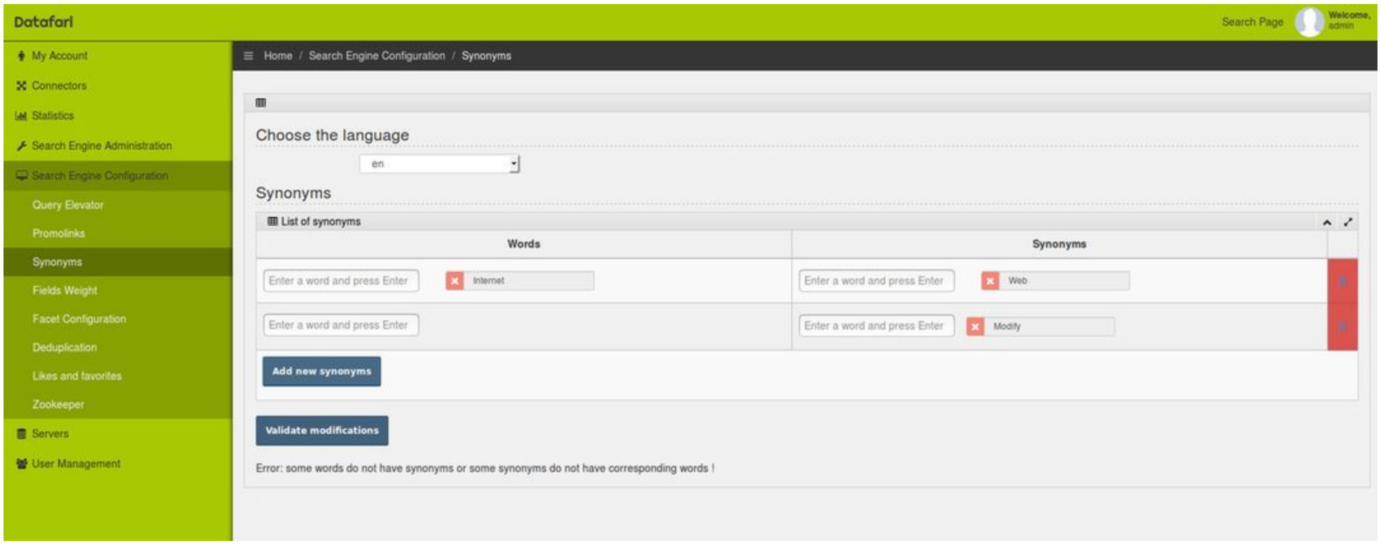
Synonyms

List of synonyms

Words	Synonyms
Enter a word and press Enter – Internet	Enter a word and press Enter – Web
Enter a word and press Enter – Customize	Enter a word and press Enter – Modify

Modifications done. Reload the Solr Core to apply them

Otherwise you will get an error:



Choose the language

en

Synonyms

List of synonyms

Words	Synonyms
Enter a word and press Enter – Internet	Enter a word and press Enter – Web
Enter a word and press Enter	Enter a word and press Enter – Modify

Error: some words do not have synonyms or some synonyms do not have corresponding words !

Search users

In this section, we present all the functionalities that can be used by both anonymous and connected search users.

1. Anonymous search user :

This is the most basic type of user. This user can only use the search functionality, and gets only two possible displays: the search bar (including autocomplete), and the results list (including facets). He has no access to functionalities such as alerts, statistics, saving favorites, or liking results. Also, since he is anonymous, he can only search through public data. Anything that has been labelled as restricted in terms of visibility (e.g. through ACLs for files) will not be searchable. If searching through these secured documents is required, the search user will need to switch to the "connected search user" role.

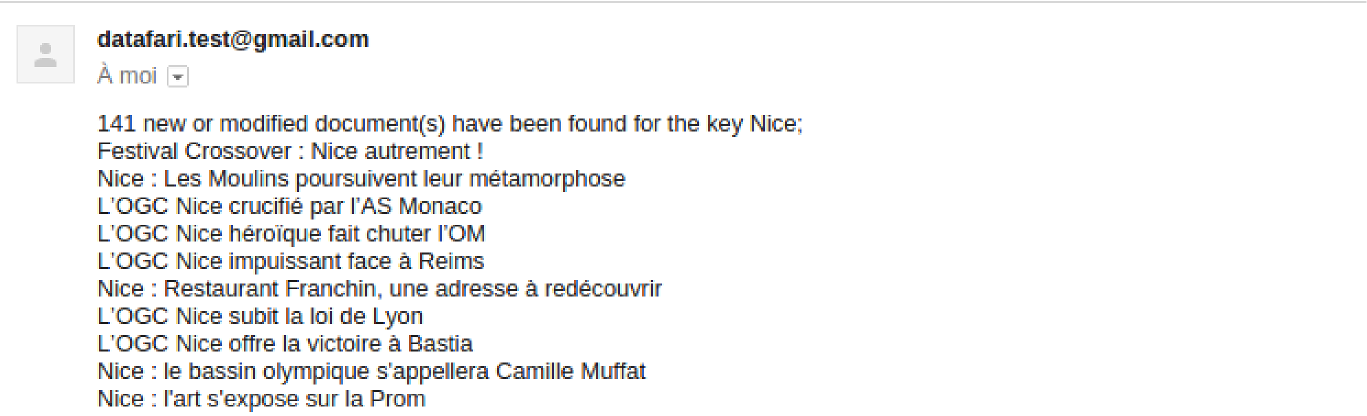
2. Connected search user :

This is a role similar to the anonymous search user role, except that it proposes more search functionalities. Obviously, the user can search using the search bar (including autocomplete) and the results list (including facets). Contrarily to the anonymous user, he has access to functionalities such as alerts and saving favorites. To reach these functionalities, the searcher needs to go in his administration panel, where he will see the required tabs. Also, he can search through secured documents. This means that when searching, Datafari does a security check on the documents and the user, and displays in the search result all the documents that the connected search user is allowed to see. Check the pages detailing the alerts functionnality and likes and favorites functionality to learn how to use it.

Alerts

> **Alerts on plain text (eg .txt) files**
> ⚠ For the moment, the alerts cannot be used on plain text files, as Tika cannot extract any metadata like the modification date from them.

Rather than actively searching for documents using queries, alerts allow you to receive information about documents via email. Once you define a query and a frequency, you regularly receive information about new and modified documents that refer to your query. For instance, the screenshot below illustrates an email alert on the keyword *Nice*.



datafari.test@gmail.com
À moi

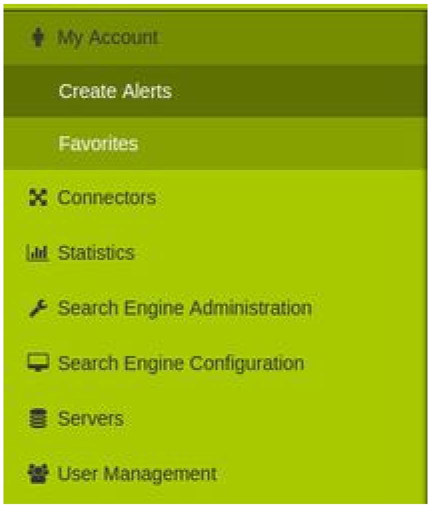
141 new or modified document(s) have been found for the key Nice;
Festival Crossover : Nice autrement !
Nice : Les Moulins poursuivent leur métamorphose
L'OGC Nice crucifié par l'AS Monaco
L'OGC Nice héroïque fait chuter l'OM
L'OGC Nice impuissant face à Reims
Nice : Restaurant Franchin, une adresse à redécouvrir
L'OGC Nice subit la loi de Lyon
L'OGC Nice offre la victoire à Bastia
Nice : le bassin olympique s'appellera Camille Muffat
Nice : l'art s'expose sur la Prom

Each document is composed of three pieces of information : the document title, where to find it, and when it has been created or modified.

Connected Search user: managing your alerts

In order to create an alert, you need to be connected with a connected search user role.

We expose a web interface. First, go to the administration menu, click on My Account and select Create Alerts in the dropdown list.



You then have access to the UI that allows you to create Alerts by filling the fields, and choosing a frequency.

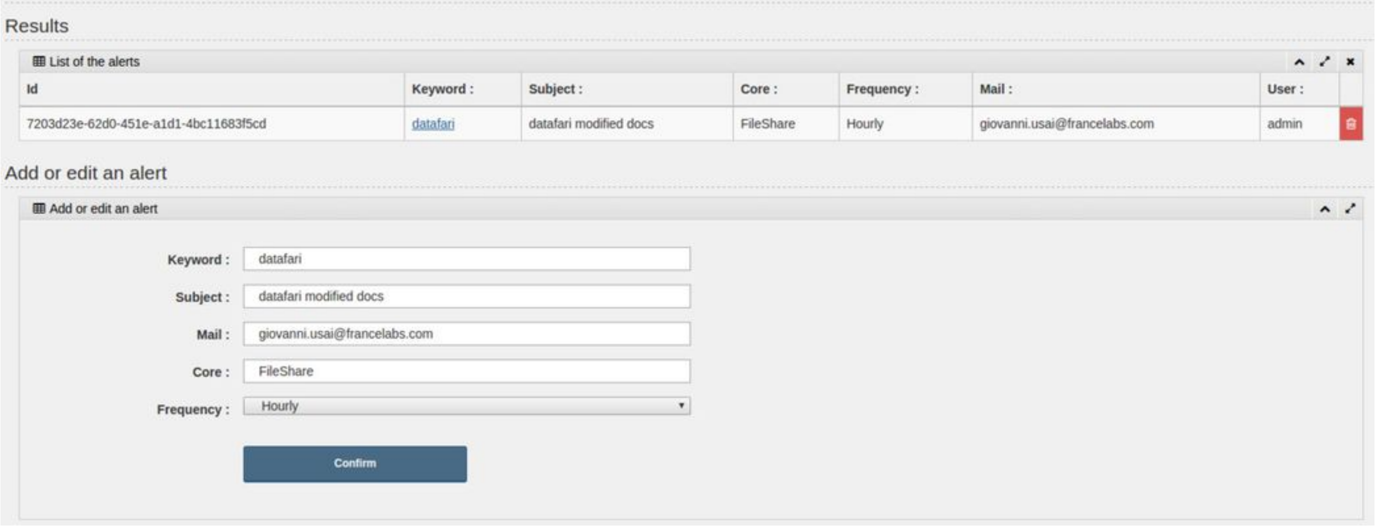
The form asks for the following information:

- ***Keyword*** is the word that will be used by the Solr request,
- ***Subject*** is the subject of the email,
- **Mail** is the recipient's email address,
- ***Core*** is the Solr core in which the request will be made,
- ***Frequency*** is the frequency for triggering your search query and sending the corresponding emails.

If you want to know which alerts you have already created, use the search bar on top of the page, and type in what you want to search for (it searches in the queries keywords). If you want to see all your searches, leave the search bar empty and click on the search icon.

The identified results are then displayed in the results table. You can edit an existing alert by clicking on the keyword, or delete one by clicking on the red button.

Note that the alert results are related to the currently connected user. Only the search admin role and the search expert role can see all the existing alerts for all the users.


Results

List of the alerts

Id	Keyword :	Subject :	Core :	Frequency :	Mail :	User :
7203d23e-62d0-451e-a1d1-4bc11683f5cd	datafari	datafari modified docs	FileShare	Hourly	giovanni.usai@francelabs.com	admin

Add or edit an alert

Add or edit an alert

Keyword : datafari
Subject : datafari modified docs
Mail : giovanni.usai@francelabs.com
Core : FileShare
Frequency : Hourly

Autocomplete

Autocomplete displays a dropdown window below the main search bar. It suggests a list pf search queries based on what the user is currently typing, more precisely by suggesting words that start with what the user is currently typing. To pick a suggestion, the user clicks on it with the left mouse button.

Deduplication

Datafari can allow a user to see wich documents are duplicated in the result of the search.

The deduplication functionnality uses the MD5 Algorithm for hashing the documents so that solr could recognize which documents are duplicated.

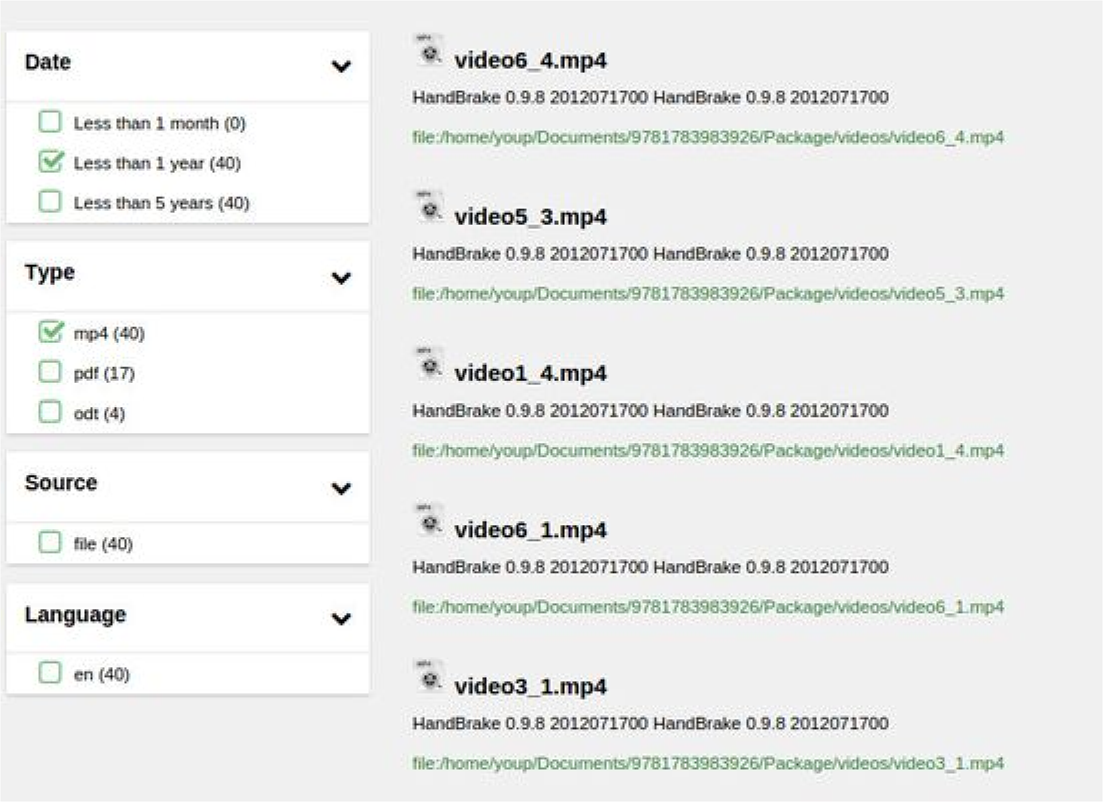
When activated, users have a special “duplication” facet that appears on the bottom left of the results page. Each item in this facets represents a set of duplicated documents, with a name and the number of duplications in parenthesis.

When clicking on a facet item, the results display will show all the duplicated documents related to the clicked facet item. This functionality can be useful to find out how many duplicated documents are present in the corpus.

Facets

Facets allow you to only see the results matching a certain criteria.

It can be the type of a document or its modification date, or its content language. Facets are available in the search and you can click on the criteria you want to activate or desactivate this functionnality.



Date

□ Less than 1 month (0)
☑ Less than 1 year (40)
□ Less than 5 years (40)

Type

☑ mp4 (40)
□ pdf (17)
□ odt (4)

Source

□ file (40)

Language

□ en (40)

video6_4.mp4
HandBrake 0.9.8 2012071700 HandBrake 0.9.8 2012071700
file:/home/youp/Documents/9781783983926/Package/videos/video6_4.mp4

video5_3.mp4
HandBrake 0.9.8 2012071700 HandBrake 0.9.8 2012071700
file:/home/youp/Documents/9781783983926/Package/videos/video5_3.mp4

video1_4.mp4
HandBrake 0.9.8 2012071700 HandBrake 0.9.8 2012071700
file:/home/youp/Documents/9781783983926/Package/videos/video1_4.mp4

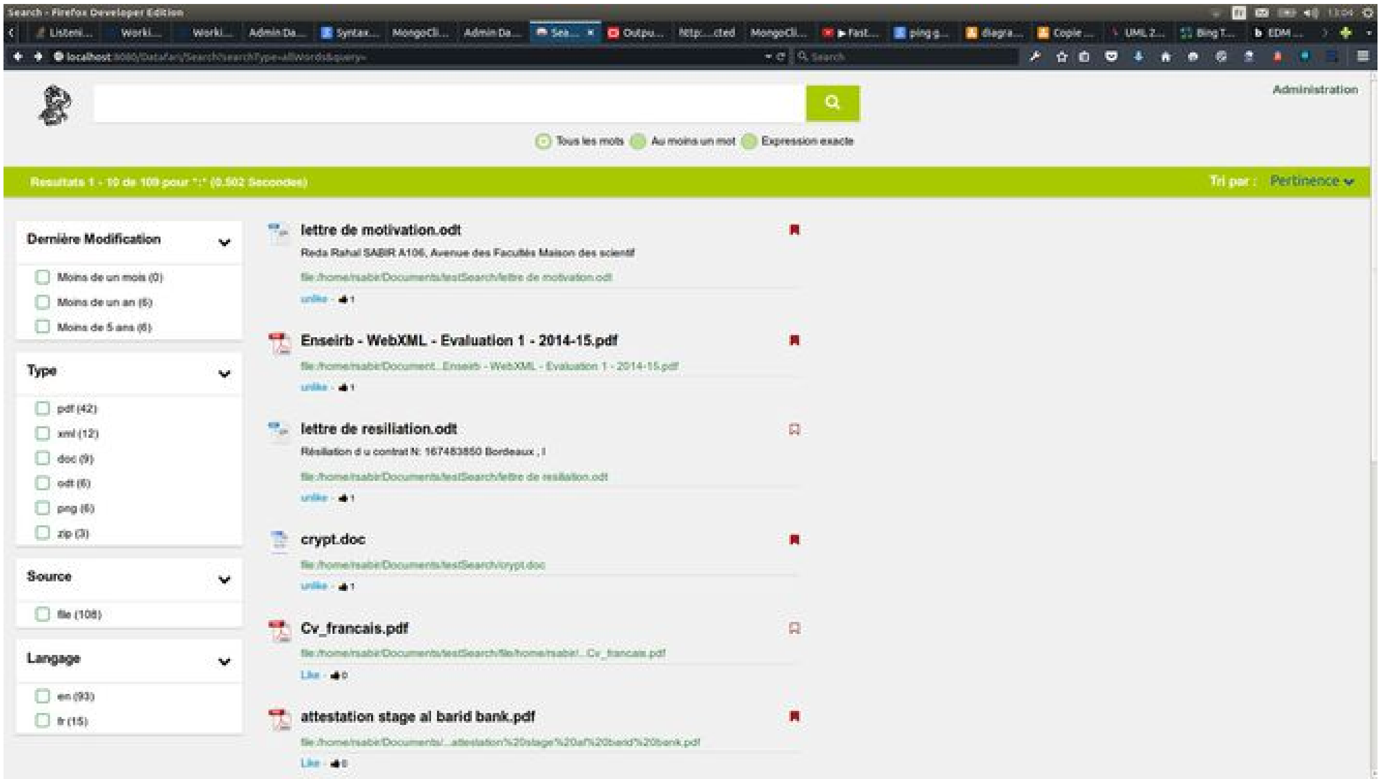
video6_1.mp4
HandBrake 0.9.8 2012071700 HandBrake 0.9.8 2012071700
file:/home/youp/Documents/9781783983926/Package/videos/video6_1.mp4

video3_1.mp4
HandBrake 0.9.8 2012071700 HandBrake 0.9.8 2012071700
file:/home/youp/Documents/9781783983926/Package/videos/video3_1.mp4

Favorites

The favorites functionnality allows connected user to save search results that they appreciate or want to look at later on.

When activated, the search interface will be as follows:


Search - Firefox Developer Edition
localhost:8080/Datafari/Search/searchType=allWords&query=
Administration

○ Tous les mots ○ Au moins un mot ○ Expression exacte

Resultats 1 - 10 de 109 pour ":" (0.502 Secondes)

Tri par : Pertinence

Dernière Modification

- □ Moins de un mois (0)
- □ Moins de un an (6)
- □ Moins de 5 ans (6)

Type

- □ pdf (42)
- □ xml (12)
- □ doc (9)
- □ odt (6)
- □ png (6)
- □ zip (3)

Source

- □ file (108)

Langage

- □ en (93)
- □ fr (15)

lettre de motivation.odt
Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif
file:/home/rsabir/Documents/testSearch/lettre de motivation.odt
unlike - 1

Enseirb - WebXML - Evaluation 1 - 2014-15.pdf
file:/home/rsabir/Document...Enseirb - WebXML - Evaluation 1 - 2014-15.pdf
unlike - 1

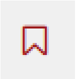
lettre de resiliation.odt
Résiliation d u contrat N: 167483850 Bordeaux , l
file:/home/rsabir/Documents/testSearch/lettre de resiliation.odt
unlike - 1

crypt.doc
file:/home/rsabir/Documents/testSearch/crypt.doc
unlike - 1

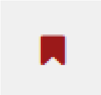
Cv_francais.pdf
file:/home/rsabir/Documents/testSearch/file/home/rsabir/...Cv_francais.pdf
Like - 0

attestation stage al barid bank.pdf
file:/home/rsabir/Documents/...attestation%20stage%20al%20barid%20bank.pdf
Like - 0

The only difference from a standard search result interface, is that it now includes the “favorite” icon next to each result title.This icon is a little red flag.



When the flag is empty, it means the corresponding document is not part of the user's favorites.



When the flag is fully red, it means it is part of the user's favorites. Clicking on the flag will add or delete the document into the favorites.

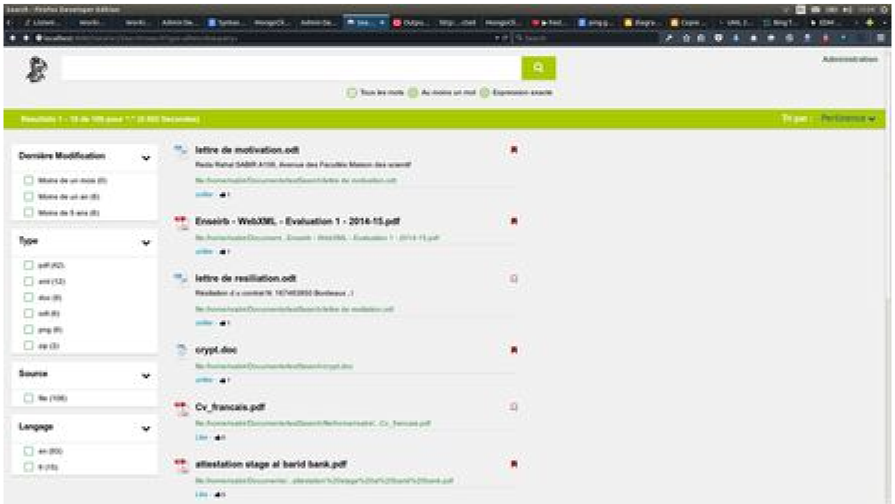
In the screenshot above, *crypt.doc* is saved as a favorite (fully red flag), *Cv_français.pdf* is not (empty red flag).

Highlighting

When doing a search, Datafari highlights parts of the results list. The highlight is set on terms of the user queries that can be found in the results list, be it in the title or in the snippet. This allows for a quick understanding of the document context for the query terms stand.

Likes

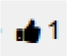
Inspired from social networks, **Likes** allows users to like selected search result entries. When activated, the search interface will be as follows:


Dernière Modification
Type
Source
Langage
lettre de motivation.odt
Enseirb - WebXML - Evaluation 1 - 2014-15.pdf
lettre de resiliation.odt
crypt.doc
Cv_francais.pdf
attestation stage al barid bank.pdf

The only difference from a standard search result interface, is that it now includes the “like” text below each result entry, as well as the number of likes, identified with a “thumbs-up” icon.

The display of the “like” button means that the connected user has not liked the document yet.

The display of the “unlike” button means that the connected user has already liked the document. By cliking on the (un)like button, you will switch the state of the document to liked or unliked.



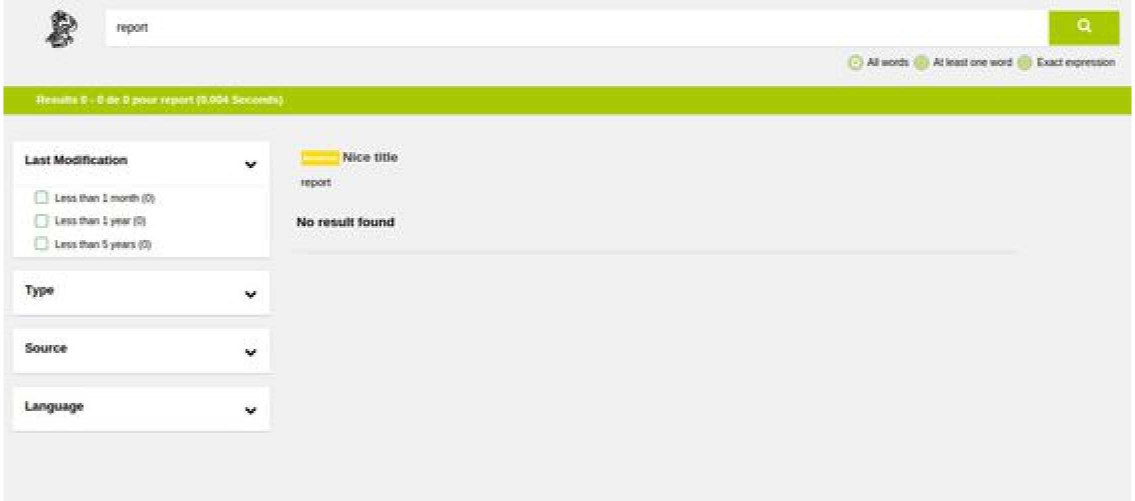
Next to the (un)like button, the thumbs-up symbol indicates the number of cumulated likes for the document.

In the screenshot above, *crypt.doc* is liked and only one person likes this document.

Promolinks

Promolinks can be conceptually assimilated to google adwords: independently from the indexed content, it allows the search expert to display text in a dedicated place on the screen (usually on top or on the side). This display is triggered by the presence of specific terms in the search query.

For instance, the illustration below shows a promolink entitled "Nice title" which has the content “report” and is displayed because the keyword "report" was used in the search query:



Results 0 - 0 de 0 pour report (0.004 Seconds)

□ Less than 1 month (0)
□ Less than 1 year (0)
□ Less than 5 years (0)

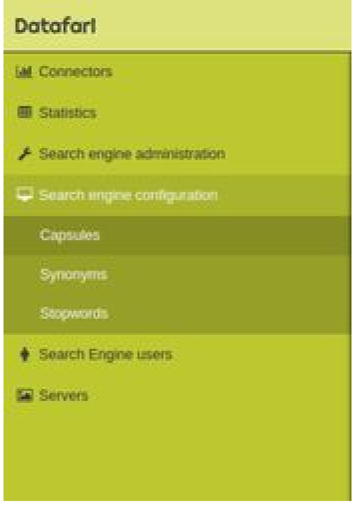
Nice title
report

No result found

Search expert: managing promolinks

In order to create Promolinks, you need to have the search expert role.

You have access to a dedicated web interface in the admin UI. First, go to the administration menu, click on Search Engine Configuration and select Capsule in the dropdown list.



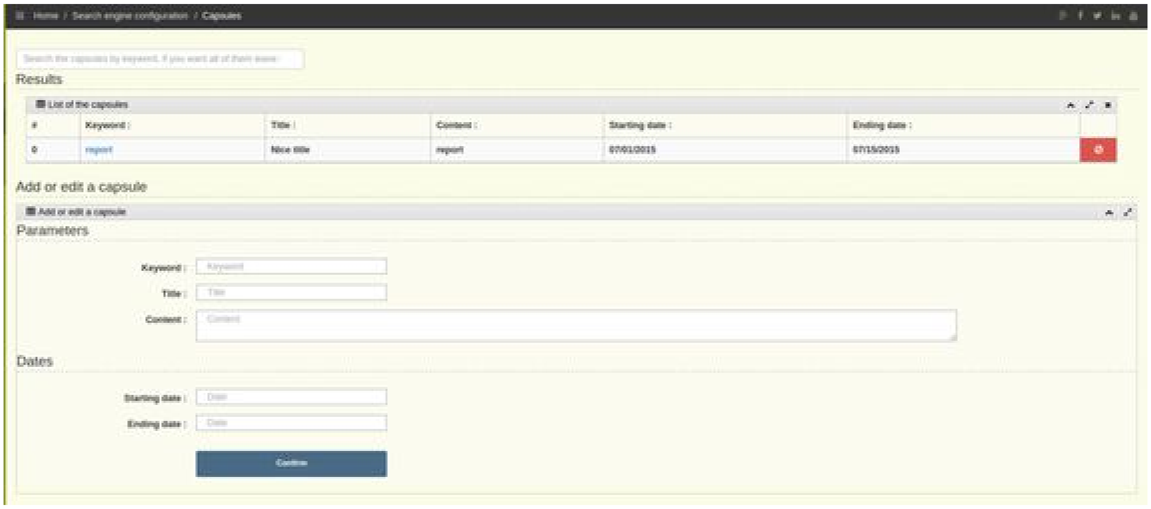
You then have access to the promolink admin UI, which shows the list of all existing promolinks. You can do the following operations on the existing promolinks:

- Delete by clicking the red button,
- Edit by clicking the keyword

Using the search bar on top, you can filter out the results by typing in something and pressing enter.

To add a promolink, use the form at the bottom of the page, and fill in the fields to add a promolink.

Note that the dates are optional, that you may fill only one out of those fields, and that if you choose to fill both, the starting date must be prior to then ending date. If dates have been filled during the add, Datafari will check if, when a search is made on the matching keyword, the current date is posterior to the starting date and prior to the ending date.



Results

List of the capsules

#	Keyword :	Title :	Content :	Starting date :	Ending date :	
0	report	Nice title	report	07/01/2015	07/15/2015	

Add or edit a capsule

Add or edit a capsule

Parameters

Keyword : Keyword

Title : Title

Content : Content

Dates

Starting date : Date

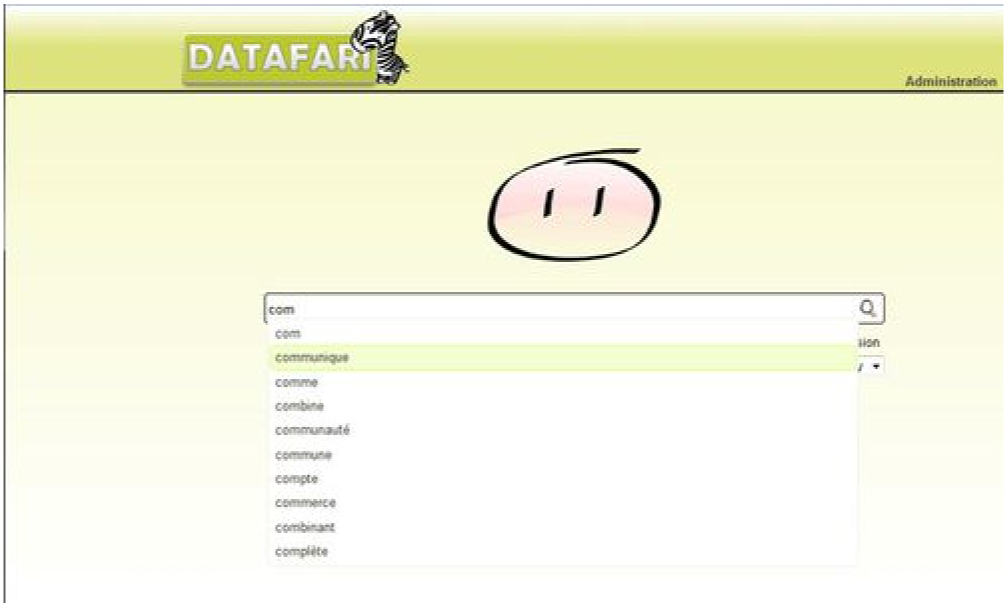
Ending date : Date

In the creation phase, if you type in a keyword that is already used as a keyword for another capsule, the existing capsule will be displayed and you can choose to replace it or not. This also means that you cannot have two capsules using the same keyword.

Searching

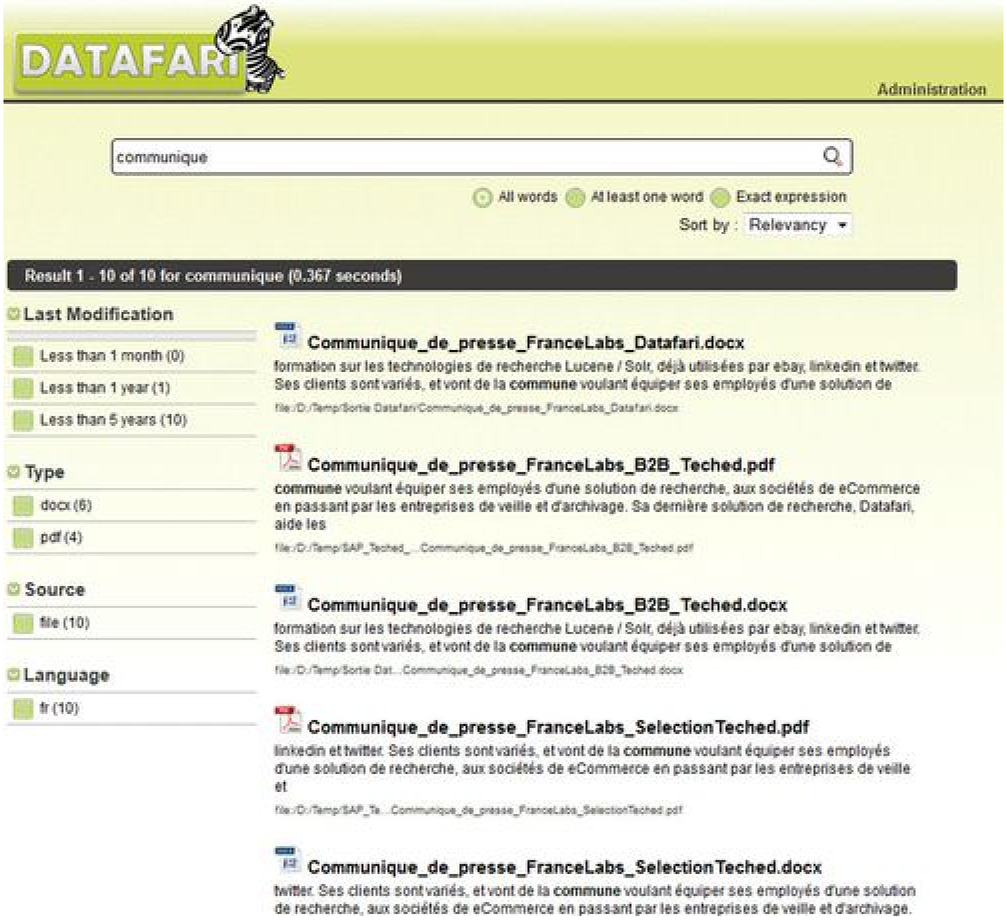
The search phase is the most "intuitive" one, in a sense that it leverages user interface paradigms that are already widespread thanks to web search engines such as Google or Bing, and also thanks for ecommerce search engines such as eBay or Amazon. We explain briefly here the different capabilities that Datafari exposes, thanks to the graphical framework Ajaxfrancelabs, and of course thanks to the Apache Solr engine working behind the scene.

The first page you see when you use the search functionnality, is a simple search bar. Start typing text in it, and the autocomplete automatically starts proposing you terms that are present in the search index, ranked by "relevance" (the relevance here being mainly correlated to the terms frequency in the corpus). You can either continue typing, or at any point in time you can click on one of the proposed term. The autocomplete by default is a term by term autocomplete. If you select a term, and type a whitespace and start typing a second term, the autocomplete will work on the second term.



Once you've clicked the search icon, the display takes you to the search result page. This page is composed as follows: on the upper part, you get the standard search bar. On the left, you get the default facets to filter and navigate through the results. And on the right, you get the list of results ranked by relevance, related to the query terms. These panels use defaults views, and you can refer to the Ajaxfrancelabs framework documentation if you want to modify the display. The deeper the modifications, the more probable it is that you will need to modify also the configuration of the Solr of Datafari, in which case you should refer to the reference Apache Solr 4.8 documentation.

- Facets panel: by default, the following facets have been configured:
    - Last modifications: allows to filter based on time windows compared to the current date. Less than 1 month old documents, Less than 1 year old documents, Less than 5 years old documents.
    - Type: allows to filter based on document types (pdf, doc, docx, xls, html ...)
    - Source: allows to filter based on the source repository configured in Apache ManifoldCF
    - Language: based on Solr capability to autodetect languages, allows to filter based on the documents language.
- Results list: by default, each result in this list is composed as follows:
    - Graphic icon symbolising the document type
    - Document file name in bold font / Title from the HTML header title in case of web page
    - Text snippet, 3 lines maximum, surrounding the query terms found in the document. These query terms are highlighted in bold fonts.
    - Document path in the source repository (or URL for web pages)



communique

Sort by : Relevancy

Result 1 - 10 of 10 for communique (0.367 seconds)

Last Modification

- Less than 1 month (0)
- Less than 1 year (1)
- Less than 5 years (10)

Type

- docx (6)
- pdf (4)

Source

- file (10)

Language

- fr (10)

Communique_de_presse_FranceLabs_Datafari.docx
formation sur les technologies de recherche Lucene / Solr, déjà utilisées par ebay, linkedin et twitter. Ses clients sont variés, et vont de la **commune** voulant équiper ses employés d'une solution de
file:/D:/Temp/Sortie Datafari/Communique_de_presse_FranceLabs_Datafari.docx

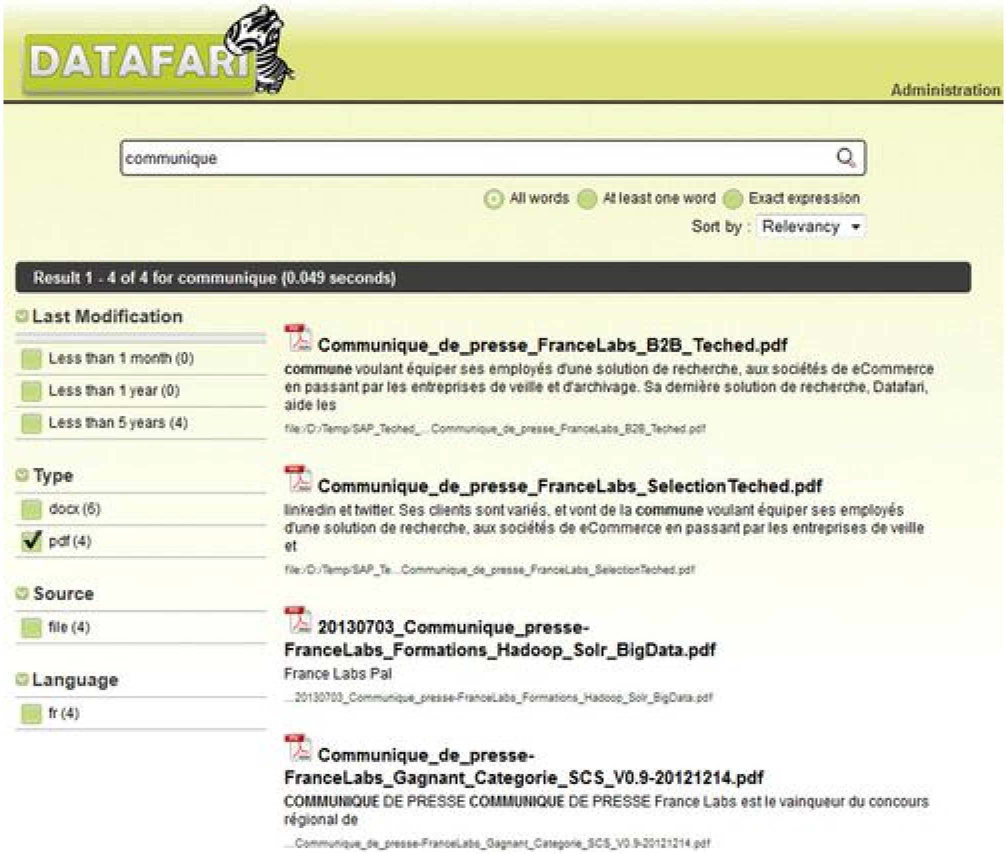
Communique_de_presse_FranceLabs_B2B_Teched.pdf
commune voulant équiper ses employés d'une solution de recherche, aux sociétés de eCommerce en passant par les entreprises de veille et d'archivage. Sa dernière solution de recherche, Datafari, aide les
file:/D:/Temp/SAP_Teched_...Communique_de_presse_FranceLabs_B2B_Teched.pdf

Communique_de_presse_FranceLabs_B2B_Teched.docx
formation sur les technologies de recherche Lucene / Solr, déjà utilisées par ebay, linkedin et twitter. Ses clients sont variés, et vont de la **commune** voulant équiper ses employés d'une solution de
file:/D:/Temp/Sortie Dat...Communique_de_presse_FranceLabs_B2B_Teched.docx

Communique_de_presse_FranceLabs_SelectionTeched.pdf
linkedin et twitter. Ses clients sont variés, et vont de la **commune** voulant équiper ses employés d'une solution de recherche, aux sociétés de eCommerce en passant par les entreprises de veille et
file:/D:/Temp/SAP_Te...Communique_de_presse_FranceLabs_SelectionTeched.pdf

Communique_de_presse_FranceLabs_SelectionTeched.docx
twitter. Ses clients sont variés, et vont de la **commune** voulant équiper ses employés d'une solution de recherche, aux sociétés de eCommerce en passant par les entreprises de veille et d'archivage.

Facets have a standard way of working: if you click on a facet value, it will filter out all the results that don't satisfy the facet value condition. In the illustration below, selecting the pdf value in the Type facet only displays pdf documents in the result panel.



DATAFARI

Administration

communique

● All words ○ At least one word ○ Exact expression

Sort by : Relevancy

Result 1 - 4 of 4 for communique (0.049 seconds)

Last Modification

- ☐ Less than 1 month (0)
- ☐ Less than 1 year (0)
- ☐ Less than 5 years (4)

Type

- ☐ docx (6)
- ☑ pdf (4)

Source

- ☐ file (4)

Language

- ☐ fr (4)

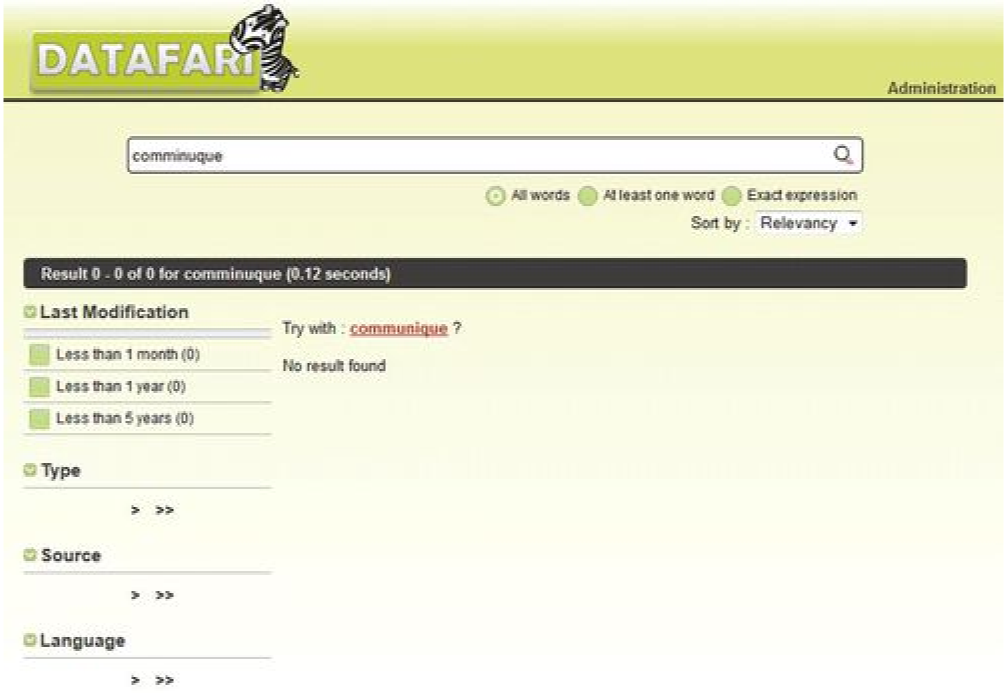
Communique_de_presse_FranceLabs_B2B_Teched.pdf
commune voulant équiper ses employés d'une solution de recherche, aux sociétés de eCommerce en passant par les entreprises de veille et d'archivage. Sa dernière solution de recherche, Datafari, aide les
file:/D:/Temp/SAP_Teched_...Communique_de_presse_FranceLabs_B2B_Teched.pdf

Communique_de_presse_FranceLabs_SelectionTeched.pdf
linkedin et twitter. Ses clients sont variés, et vont de la **commune** voulant équiper ses employés d'une solution de recherche, aux sociétés de eCommerce en passant par les entreprises de veille et
file:/D:/Temp/SAP_Te...Communique_de_presse_FranceLabs_SelectionTeched.pdf

20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf
France Labs Pal
...20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf

Communique_de_presse-FranceLabs_Gagnant_Categorie_SCS_V0.9-20121214.pdf
COMMUNIQUE DE PRESSE **COMMUNIQUE** DE PRESSE France Labs est le vainqueur du concours régional de
...Communique_de_presse-FranceLabs_Gagnant_Categorie_SCS_V0.9-20121214.pdf

Datafari also proposes the spellchecker functionnality of Apache Solr. In the illustration below, we enter the query term "comminuque" instead of "communique". The spellchecker automatically proposes a correct word. By default, Datafari does not do a search on this suggestion, but it can do it with slight modifications.



DATAFARI

Administration

comminuque

● All words ○ At least one word ○ Exact expression

Sort by : Relevancy

Result 0 - 0 of 0 for comminuque (0.12 seconds)

Last Modification

- ☐ Less than 1 month (0)
- ☐ Less than 1 year (0)
- ☐ Less than 5 years (0)

Type

> > &gt;&gt;

Source

> > &gt;&gt;

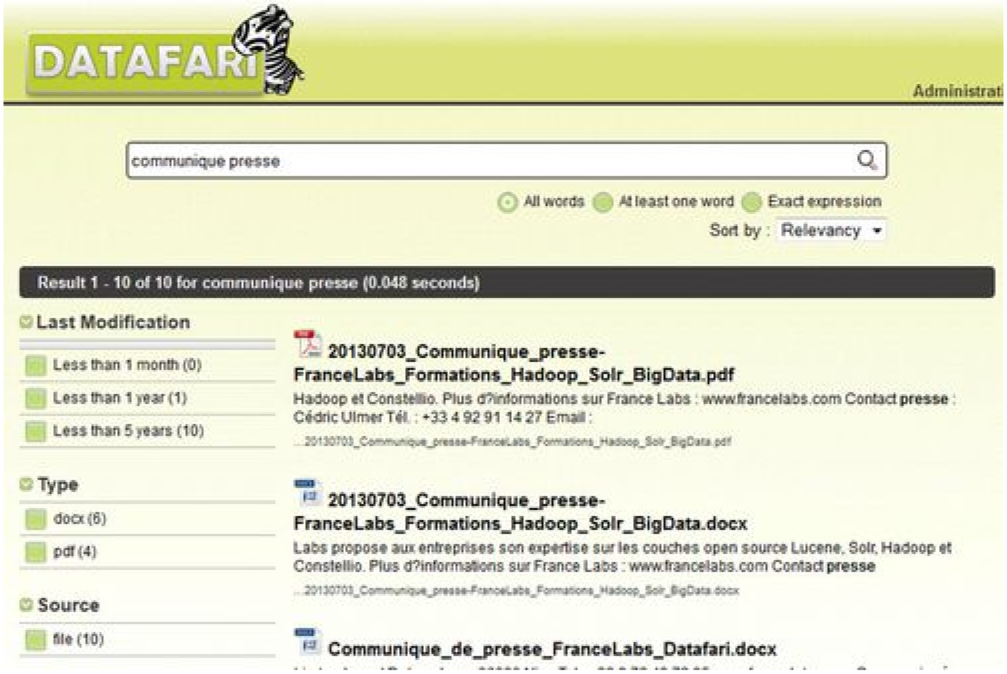
Language

> > &gt;&gt;

Try with : communique ?

No result found

In the search bar, you can type in several query terms, and you can put operators to fine tune your search, such as AND and OR. Check the Apache Solr reference documentation for the full list of operators.



communique presse

◉ All words ○ At least one word ○ Exact expression

Sort by : Relevancy ▾

Result 1 - 10 of 10 for communique presse (0.048 seconds)

Last Modification
- Less than 1 month (0)
- Less than 1 year (1)
- Less than 5 years (10)

Type
- docx (6)
- pdf (4)

Source
- file (10)

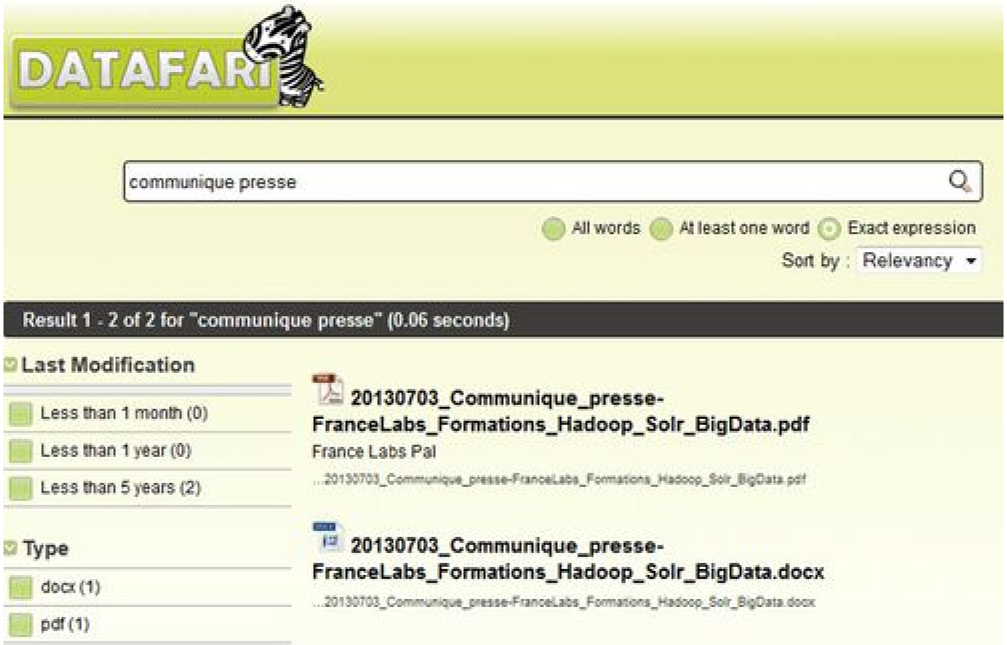
20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf
Hadoop et Constellio. Plus d?informations sur France Labs : www.francelabs.com Contact **presse** : Cédric Ulmer Tél. : +33 4 92 91 14 27 Email :
...20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf

20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.docx
Labs propose aux entreprises son expertise sur les couches open source Lucene, Solr, Hadoop et Constellio. Plus d?informations sur France Labs : www.francelabs.com Contact **presse**
...20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.docx

Communique_de_presse_FranceLabs_Datafari.docx

Right below the search bar, three buttons allow to modify the search behaviour:

- All words: forces the search engine to return only the documents that contain ALL the terms present in the query term
- At least one word: that's the standard behavior. The search engine will favor documents that contain all the terms, but it not all terms are present, it will display the documents that contain a subset of the query terms
- Exact expression: forces the search engine to return only the documents that contain the exact expression in the query term. In the illustration below, only documents containing the exact expression "communique presse" will be return. So a document containing "presse communique" won't be returned.



communique presse

○ All words ○ At least one word ◉ Exact expression

Sort by : Relevancy ▾

Result 1 - 2 of 2 for "communique presse" (0.06 seconds)

Last Modification
- Less than 1 month (0)
- Less than 1 year (0)
- Less than 5 years (2)

Type
- docx (1)
- pdf (1)

20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf
France Labs Pal
...20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.pdf

20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.docx
...20130703_Communique_presse-FranceLabs_Formations_Hadoop_Solr_BigData.docx

Stopwords

Stopwords allows you to ignore identified words at the indexing phase.

If a word listed in the stopword file is found, then Datafari will ignore it when doing the indexing of the corresponding document, except if they are only words listed as stopwords in the query.

Alerts (from Datafari 3.1)

Since the version 3.1.x of Datafari, the Alerts feature has evolved. You can now create your alert directly from the search view of Datafari. Here are the steps to follow in order to create an alert:

- Authenticate yourself in Datafari
- Perform the search on which you want to create an alert
- Click on the "Create alert" button
- Enter you e-mail address in the corresponding field and select a send frequency
- Validate

[Picture]

You can now consult the alerts you have created in the "Parameters" view in the "Alerts" tab.

[Picture]

Here you can click on the "modify" button to change the send frequency of the corresponding alert or delete it.

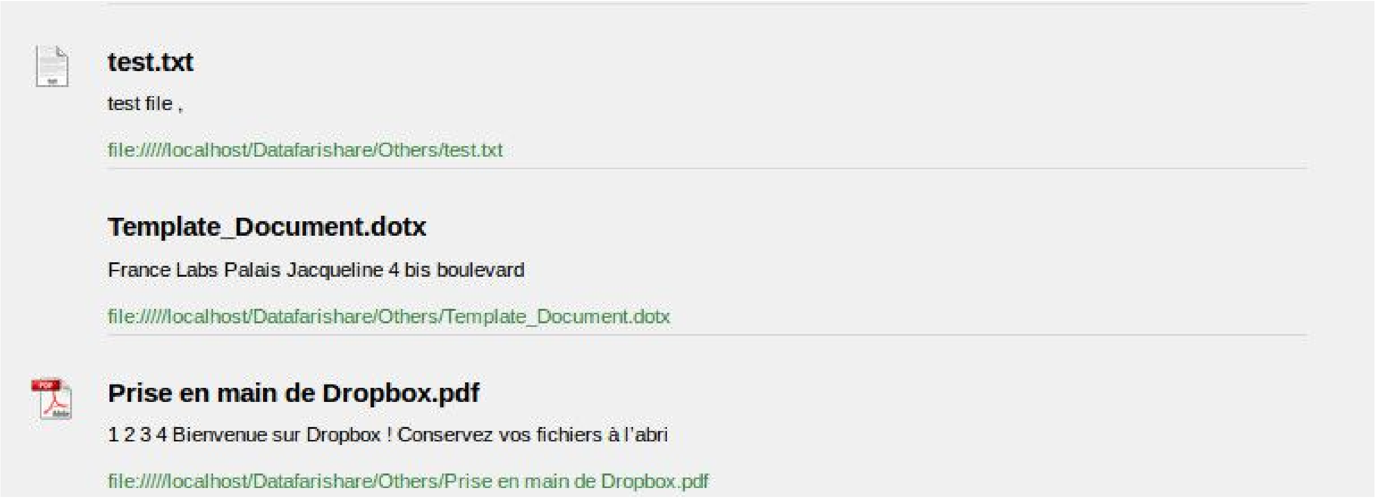
For developers

This section of the documentation is targeting a technical audience, more precisely developers who want to understand or extend the capacities of Datafari.

💡 For an overview of the available Datafari's build and deployment flows, have a look here Distributions HOW-TO

Add new icon for search results

It may happen that you have indexed documents with a specific extension which is not known by Datafari. If this is the case, the concerned documents will be displayed in the result view without any icon (here Template_Document.dotx):



test.txt

test file ,

file://///localhost/Datafarishare/Others/test.txt

Template_Document.dotx

France Labs Palais Jacqueline 4 bis boulevard

file://///localhost/Datafarishare/Others/Template_Document.dotx

Prise en main de Dropbox.pdf

1 2 3 4 Bienvenue sur Dropbox ! Conservez vos fichiers à l'abri

file://///localhost/Datafarishare/Others/Prise en main de Dropbox.pdf

To correct/avoid this, you will have to add a new icon for your concerned extension in 'DATAFARI_HOME_DIR/tomcat/webapps/Datafari/images/icons'. The file must be a png type and respect the following pattern : [extension_name]-icon-24x24.png

For example, the icon file to add to the icons folder in order to correct the problem showed above would be : dotx-icon-24x24.png

Of course you can make a copy of an existing icon from the icons directory and rename it.

Once the icon is available, it will be automatically used by Datafari.

Ajaxfrancelabs graphical framework

AjaxFranceLabs is a contribution from our company so that the community can manipulate Solr, Datafari and Constellio through an Ajax Framework.

Spiritually, it is a child of the AjaxSolr framework, and you will see many similarities, in particular at the architecture level. The AjaxSolr has been a great source of inspiration for us.

Still, the code itself has been done from scratch (except for 2 classes which are almost copy pastes), as there were parts that were rather different, because of the differences between Constellio and Datafari/Solr.

Although it is not part of the framework per se, we gave a particular care to the widgets that make use of our framework, because we know that you'll probably start with these widgets in order to get a hands on on our framework (you may even want to keep them as they are).

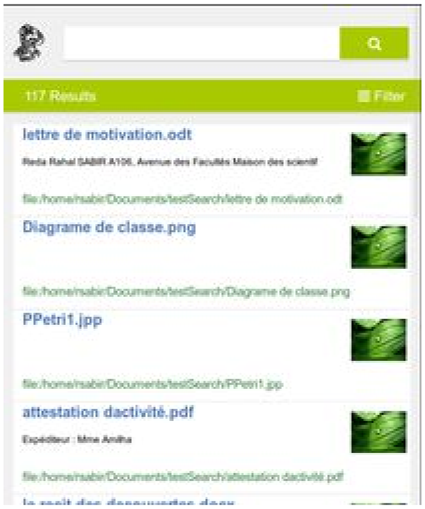
Displaying thumbnails for autocomplete and results list

This documentation explains how to enable the « pictures » mode for the results display, for instance if you are indexing videos or pictures.

> Note that this functionality is not complete yet, as of Datafari 2.0. For now the pointer to the image used is hardcoded. We still need to create a dedicated field in Solr that will hold the urls to the image, and to modify the widget so that it uses this field.

1. In searchView.jsp, you need to include « results-illustrate.css » which is in the css folder of datafari. Note that the css is already present, you just need to uncomment it.
2. In search.js, you need to replace resultWidget by with ResultIllustratedWidget, using the same attributes that resultWidget had.

This will make space for the image and show it at the left of the results in desktop mode and at the right of the results in the mobile mode. You can see below how it will appear :



lettre de motivation.odt

Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif

file:/home/rsabir/Documents/testSearch/lettre de motivation.odt

Diagrame de classe.png

file:/home/rsabir/Documents/testSearch/Diagrame de classe.png

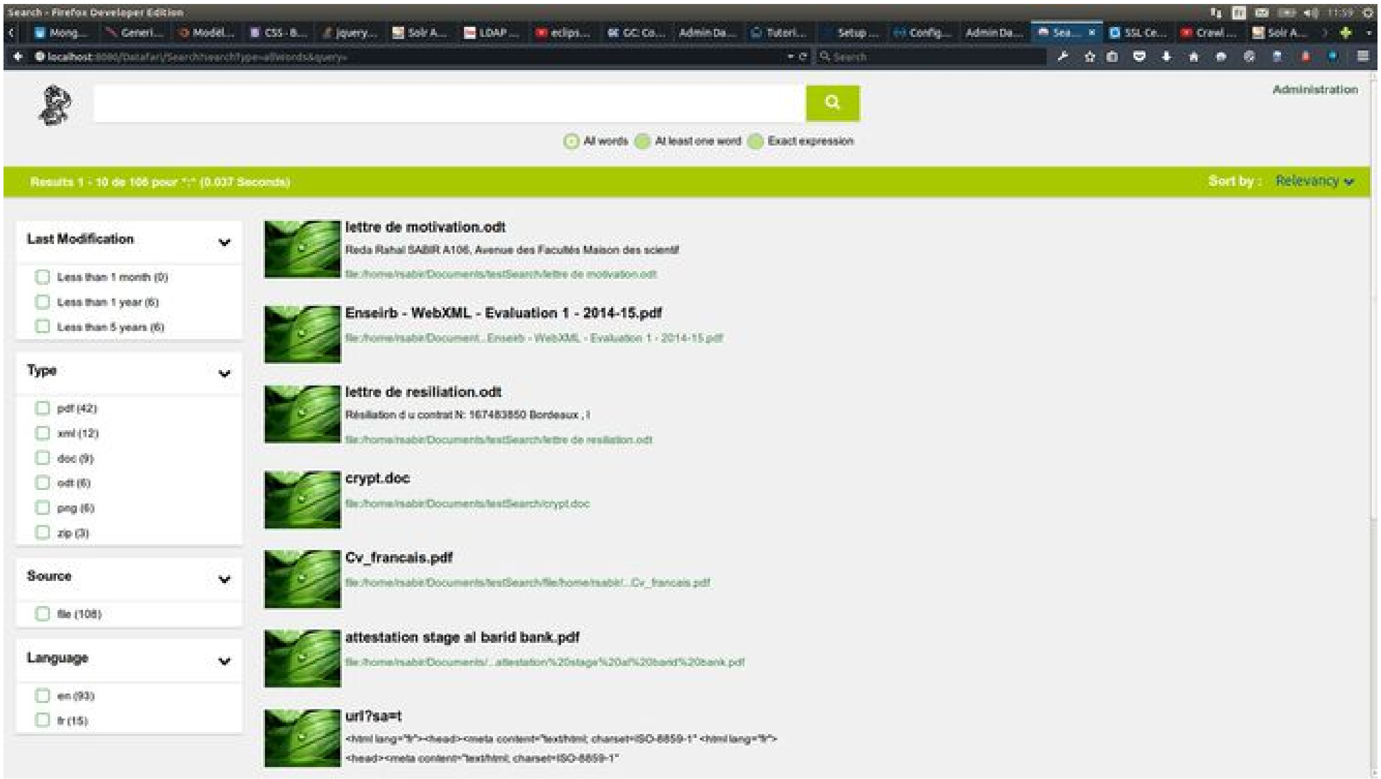
PPetri1.jpp

file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf

Expéditeur : Mme Amiha

file:/home/rsabir/Documents/testSearch/attestation dactivité.pdf


Search - Firefox Developer Edition
localhost:8080/Datafari/Search?searchType=allWords&query=
Administration
○ All words ○ At least one word ○ Exact expression
Sort by : Relevancy

Results 1 - 10 de 106 pour *:* (0.037 Seconds)

Last Modification

- □ Less than 1 month (0)
- □ Less than 1 year (6)
- □ Less than 5 years (6)

Type

- □ pdf (42)
- □ xml (12)
- □ doc (9)
- □ odt (6)
- □ png (6)
- □ zip (3)

Source

- □ file (108)

Language

- □ en (93)
- □ fr (15)

lettre de motivation.odt
Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif
file:/home/rsabir/Documents/testSearch/lettre de motivation.odt

Enseirb - WebXML - Evaluation 1 - 2014-15.pdf
file:/home/rsabir/Document...Enseirb - WebXML - Evaluation 1 - 2014-15.pdf

lettre de resiliation.odt
Résiliation d u contrat N: 167483850 Bordeaux , l
file:/home/rsabir/Documents/testSearch/lettre de resiliation.odt

crypt.doc
file:/home/rsabir/Documents/testSearch/crypt.doc

Cv_francais.pdf
file:/home/rsabir/Documents/testSearch/file/home/rsabir/...Cv_francais.pdf

attestation stage al barid bank.pdf
file:/home/rsabir/Documents/...attestation%20stage%20al%20barid%20bank.pdf

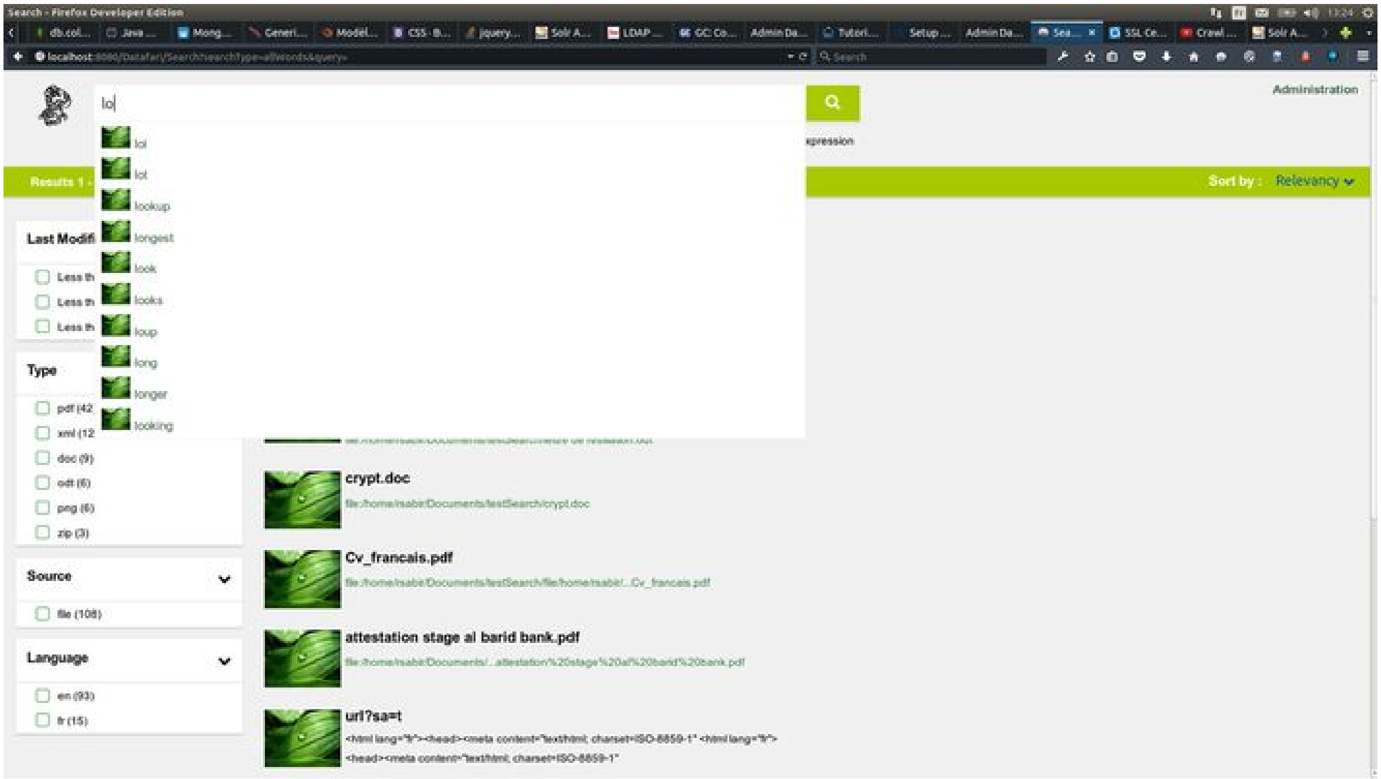
url?sa=t
<html lang="fr"><head><meta content="text/html; charset=ISO-8859-1" <html lang="fr">
<head><meta content="text/html; charset=ISO-8859-1"

Now to have the pictures in the autocomplete functionnality, do the following :

1. In search.js, replace « AjaxFranceLabs.SearchBarWidget » with « AjaxFranceLabs.SearchBarIllustratedWidget », using the same parameters. And then you have it !

Note that this functionnality uses the following two javascripts, which are included by default « *js/AjaxFranceLabs/widgets/SearchBarIllustrated.widget.js* » and « *js/AjaxFranceLabs/modules/AutocompleteIllustrated.module.js* » .

Note also that this functionality is a better fit for suggestions rather than for search autocomplete. For instance, in a digital asset management scenario, the autocomplete with the image would propose me when clicking on it directly the image rather than triggering a search on the solr. For now, we have not coded such a suggestion widget.



crypt.doc
file:/home/rsabir/Documents/testSearch/crypt.doc

Cv_francais.pdf
file:/home/rsabir/Documents/testSearch/file/home/rsabir/...Cv_francais.pdf

attestation stage al barid bank.pdf
file:/home/rsabir/Documents/...attestation%20stage%20al%20barid%20bank.pdf

url?sa=t
<html lang="fr"><head><meta content="text/html; charset=ISO-8859-1" <html lang="fr">
<head><meta content="text/html; charset=ISO-8859-1"

This widget is currently visually optimised for a fixed image size. If the source images do not have this size, we do a basic resize operation, which may lead to not-so-good-looking displays of the thumbnails.

Framework architecture and mechanism

Code Structure

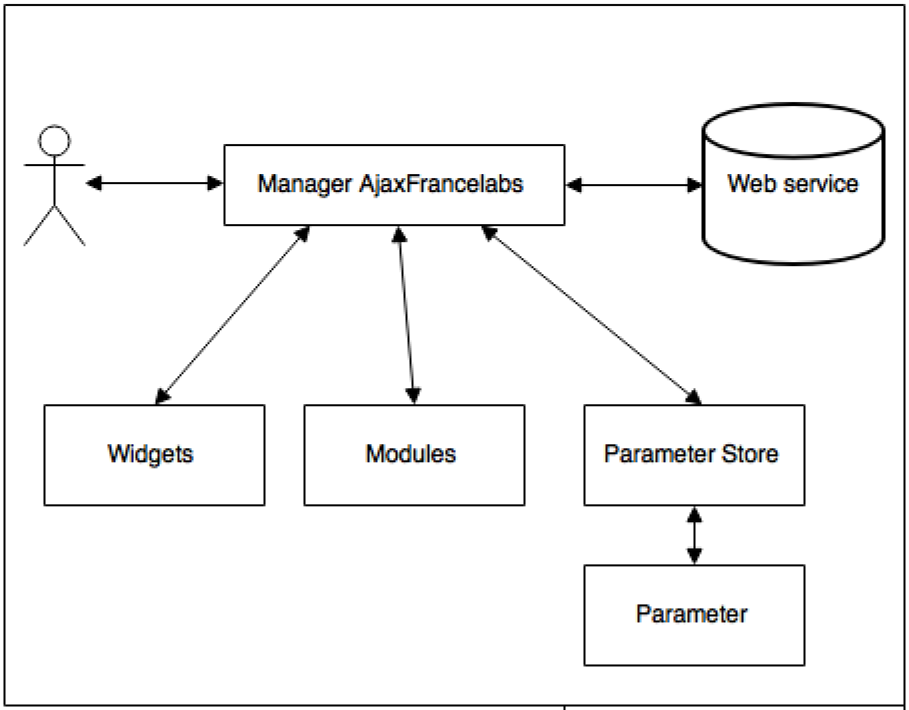
The framework contains the following folders:

- **core**: the main classes of the framework
- **manager**: implemented managers
- **module**: the different modules
- **widgets**: the different interface widgets

A widget is a part of the user interface (UI) which is attached to a DOM element.
A module is an extension of the widgets.

Eg: my result widget displays all the documents found by the search query and my favorite module will add “stars” to those documents title in order to save them to my favorite documents.

Architecture diagram



AjaxFrancelabs Architecture

Interaction between the different components

The manager acts as a master controller. The widgets and modules are dealing with the manager in order to get the information they need and to modify the query request.

The parameter store stores the different parameter necessary to process the search query.

When a request query has to be processed, the manager calls a web service which receives the search query. This web service answers back to the manager, which stores the result in a variable (response) that the different components can access.

Class diagram

Core

AjaxFranceLabs offers different utility methods:

- **string** tinyUrl(url, length)

  A method that reduces the length of the url given in parameter. Adds '…' at the end if the url have been tined. Leaves everything after the last / (page name) in the url. length is set to 75 by default.
- **string** tinyString(str, length)

  A method that reduces the length of the given string in parameter. Adds '…' at the end if the string have been tined. length is set to 25 by default.
- **string** truncate(string, length)

  Reduces the size of a string if the length of it is superior at length. This method will not slice a word in two if the max length is reach but keep it and add '…' at the end.
- <**void** addMultiElementClasses(source)

  Adds the following classes to the different elements of the source:
  - first | middle | last
  - odd | even
  - e-* where * is the index of the element in the source

  Thus you have all the necessary selectors for JavaScript and CSS.
- **void** clearMultiElementsClasses(source)

  Remove the classes first, middle, last, even, odd, e-* from the elements of source.
- **boolean** isString(obj)

  Returns true if obj is a string.

- **boolean** isRegExp(obj)

  Returns true if obj is a RegExp
- **boolean** equals(foo, bar)

  Compares the two objects given. Evaluates RegExp.
- **boolean** empty(mixed_var)

  Returns true if the mixed_var is empty (‘0’, 0, null, undefined, false, “”)
- **string** trim(str, charlist)

  Removes the invisible characters on the beginning and end of the string str. charlist is an optional parameter.
- **string** extractLast(term)

  Returns the last term of term (used for the autocomplete widget).
- **array** split(val)

  Splits val on the invisible character \u200c (used for the autocomplete widget).
- **object** extend(target, …)

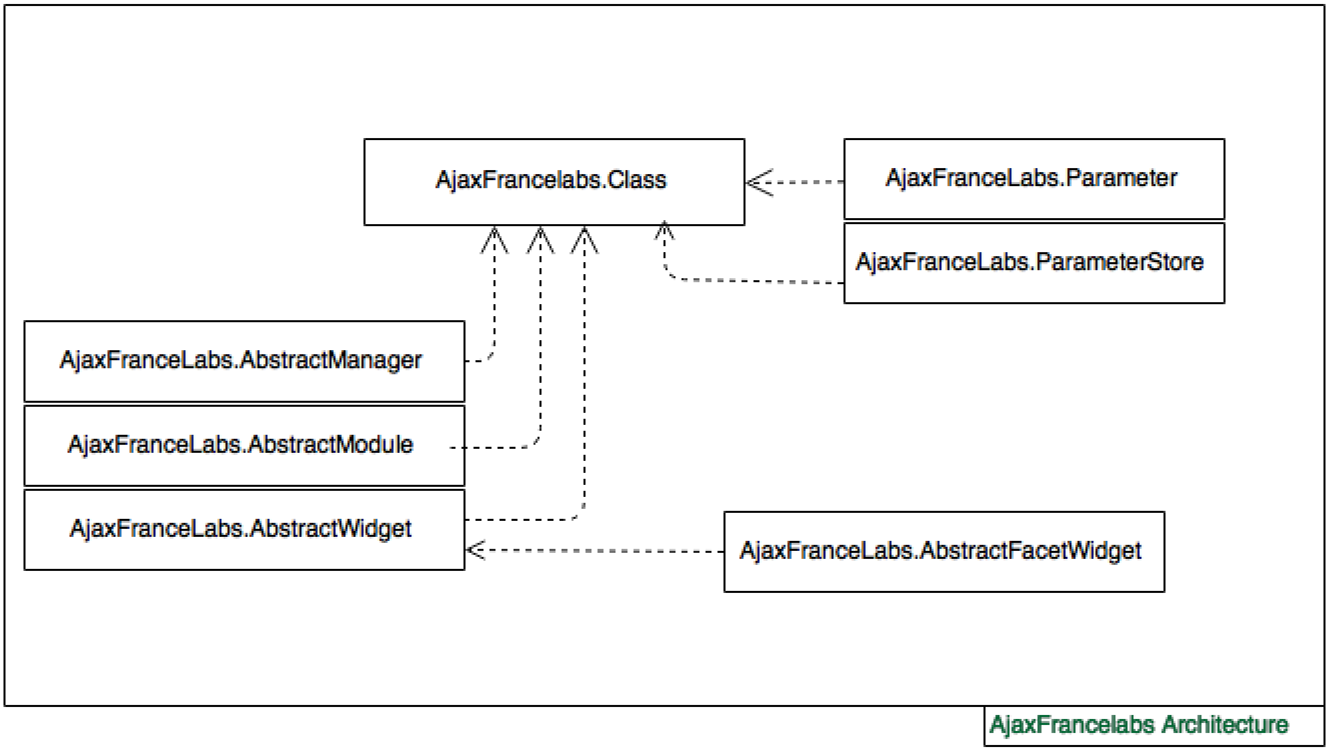
  Extends the object target with the other arguments given.
- **string** capitalize(string)

  Sets the first letter of a word to upper case. The word is given as a parameter.

The main classes of the framework are:

- AbstractManager
- AbstractWidget
- AbstractFacetWidget
- AbstractModule
- Parameter
- ParameterStore

Any class class has to extend one of these or the AjaxFranceLabs.Class. For this, use the .extend method provided by the AjaxFranceLabs.Class and any of the previous one.

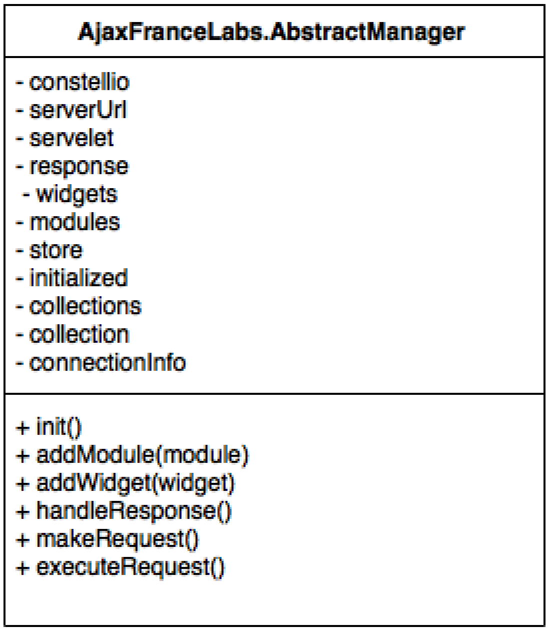

AjaxFrancelabs.Class
AjaxFranceLabs.Parameter
AjaxFranceLabs.ParameterStore
AjaxFranceLabs.AbstractManager
AjaxFranceLabs.AbstractModule
AjaxFranceLabs.AbstractWidget
AjaxFranceLabs.AbstractFacetWidget
AjaxFrancelabs Architecture

Architecture of the main classes of the framework

AbstractManager

The manager is the main class of the framework. It is the master controller, the crossroad of all the information going in and out.
This class has to be extended in order to implement executeRequest() which have to make the call to the web service.

Insert architecture of the abstract manager here



AjaxFranceLabs.AbstractManager

- constellio
- serverUrl
- servelet
- response
- widgets
- modules
- store
- initialized
- collections
- collection
- connectionInfo

+ init()
+ addModule(module)
+ addWidget(widget)
+ handleResponse()
+ makeRequest()
+ executeRequest()

Manager Class Diagream

AbstractManager class diagram

Variables

- **constellio:** boolean set to true by default. Set to false when using Solr. When set to true, the manager will load your constellio collections
- **serverUrl:** address where the web service calls have to be made
- **servlet:** servlet that have to be used for the web service calls
- **response:** contains the last response from a search query on the search engine
- **widgets:** contains all the widgets of the manager
- **modules:** contains all the modules of the manager
- **store:** a store that contains all the parameters of the search query
- **collections:** the list of collections of the Constellio
- **collection:** the collection currently used

```
connectionInfo: {
 autocomplete: {
  serverUrl: 127.0.0.1,
  servlet: 'select',
  queryString: 'callFunction=getInformations&type=user
 }
}
```

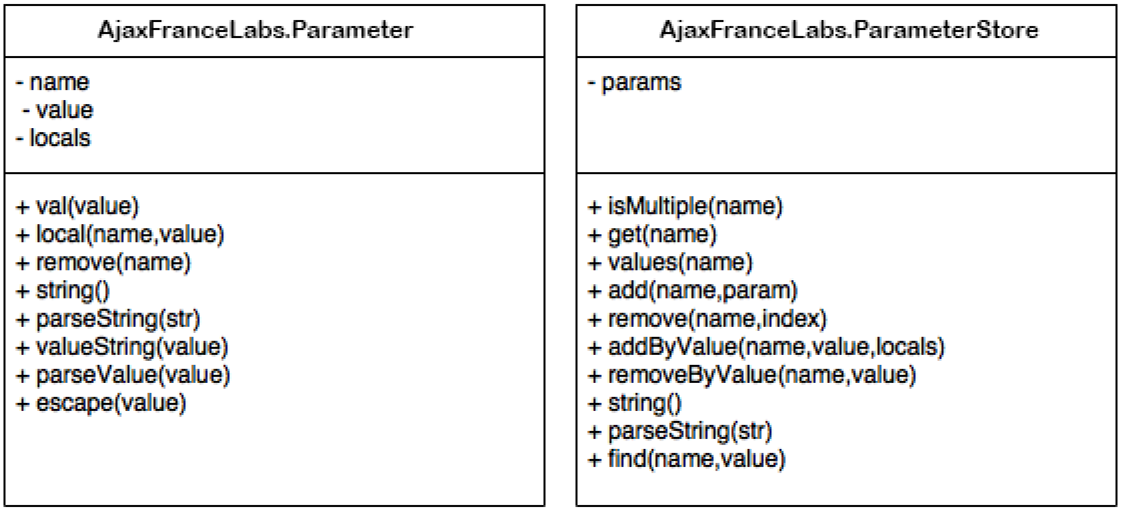
Methods

- **init:** import collections if using constellio, initializes the widgets and modules
- **addModule:** adds the module provided in parameter to the module collection (“array”)
- **addWidget:** adds the widget provided in parameter to the widget collection (“array”)
- **handleResponse:** executes the afterRequest method from all the widgets and modules after the search query has been processed
- **makeRequest:** executes the beforeRequest method from all the widgets and modules before processing the search query
- **executeRequest:** must be implemented in a Manager object, this is the method which calls the web service.

- **getWidgetByID:** gets the handle to the widget with ID specified as input.
- **makeDefaultRequest:** performs a select all (*:*) request and updates the address bar with the search query.

Parameter and ParameterStore

Insert architecture parameter/parameterstore



AjaxFranceLabs.Parameter

- name
- value
- locals

+ val(value)
+ local(name,value)
+ remove(name)
+ string()
+ parseString(str)
+ valueString(value)
+ parseValue(value)
+ escape(value)

AjaxFranceLabs.ParameterStore

- params

+ isMultiple(name)
+ get(name)
+ values(name)
+ add(name,param)
+ remove(name,index)
+ addByValue(name,value,locals)
+ removeByValue(name,value)
+ string()
+ parseString(str)
+ find(name,value)

Parameter and Parameter Store Class Diagram

Parameter and ParameterStore class diagram

Parameter

Variables

- **name:** name of the parameter
- **value:** value of the parameter
- **locals:** local variables for nested request

Methods

- **val:** if a parameter is given, change the value of the parameter, return the value instead
- **local:** if value is given, changes the value of the local parameter, returns its value instead
- **remove:** removes a local parameter
- **string:** toString method of the parameter
- **parseString:** parses a string to create the parameter
- **valueString:** transforms value into a string if it is an array and encodes special characters
- **parseValueString:** decodes special characters and transforms the string in array if necessary
- **escapeValue:** quotes the string

ParameterStore

Variables

- **params:** a collection of the parameters

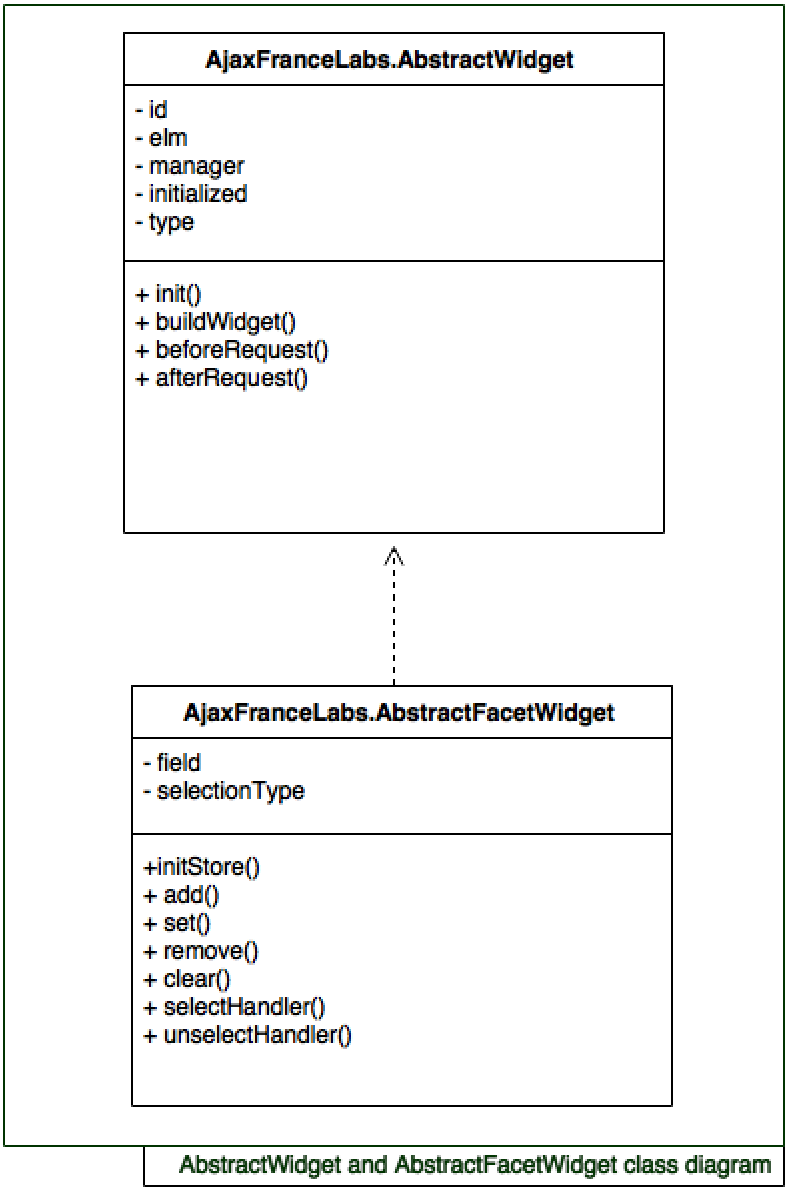
Methods

- **isMultiple:** returns true if the parameter has multiple value
- **get:** returns the parameter, creates it if it does not exists
- **values:** return the value(s) of the parameter
- **add:** add a new parameter, creates an empty Parameter if only name is given
- **remove:** deletes a parameter
- **addByValue:** creates a new Parameter
- **removeByValue:** removes a parameter with the given name and value

- **string:** the toString method of the parameterStore that returns all the parameters on a string form
- **parseString:** parses a string to creates the parameters
- **find:** check if a parameter exists with the given name and value
- **isParamDefined:** helper function to check whether the given parameter name in input is defined or not inside the parameter store.

AbstractWidget

All the widgets have to extend the AjaxFranceLabs.AbstractWidget class or the AjaxFranceLabs.AbstractFacetWidget class.



AbstractWidget and AbstractFacetWidget class diagram

AbstractWidget

Variables

- **id:** a unique id for the widget
- **elm:** element where the widget will be build
- **manager:** a reference to the widget’s manager
- **type:** the type of the widget

Methods

- **buildWidget:** a method that build the main DOM structure of your widget.
- **beforeRequest:** code that have to be executed before the manager send the search query to the web service. This is mainly used when you have to add parameters to the search query, like the content of the search bar.
- **afterRequest:** code that will be executed after the search query is done. This is mainly used when you have to update the content of your widget, for example, empty previous result list and include the new result instead.
- **reset:** implement this method to reset the widget (notably after user input).

AbstractFacetWidget

Variables

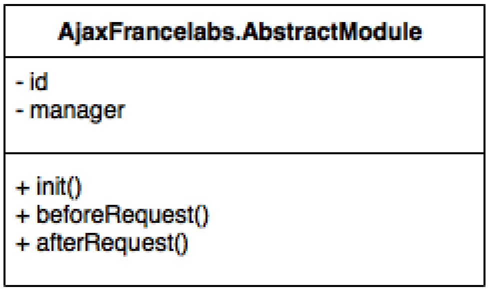
- **field:** the referring field
- **selectionType:** can take 3 values depending on how you can select facet values:
    - **AND:** intersection of facet values
    - **OR:** document contains at least one facet value
    - **ONE:** can select one facet value at a time
- **returnUnselectedFacetValues:** if set to true, values for the unselected facet will be returned anyway (default: false)

Methods

- **selectHandler:** callback method when a facet element is selected.
- **unselectHandler:** callback method when a facet element is unselected.

AbstractModule

Modules are extension of widgets, they will modify them once they are built.



AbstractModule class diagram

AbstractModule class diagram

Variables

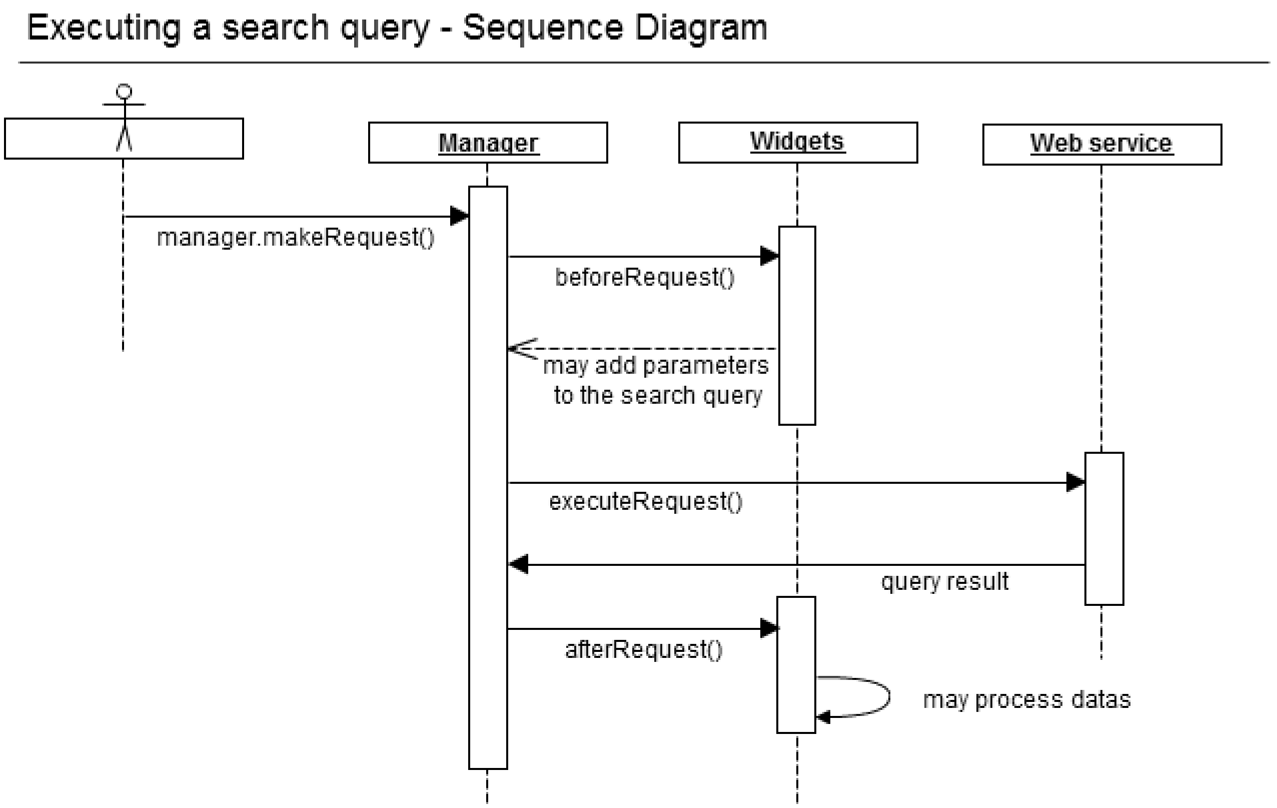
- **id:** a unique id for the module
- **manager:** a reference to the module’s manager

Methods

- **beforeRequest:** code that have to be executed before the manager send the search query to the web service.
- **afterRequest:** code that will be executed after the search query is done.

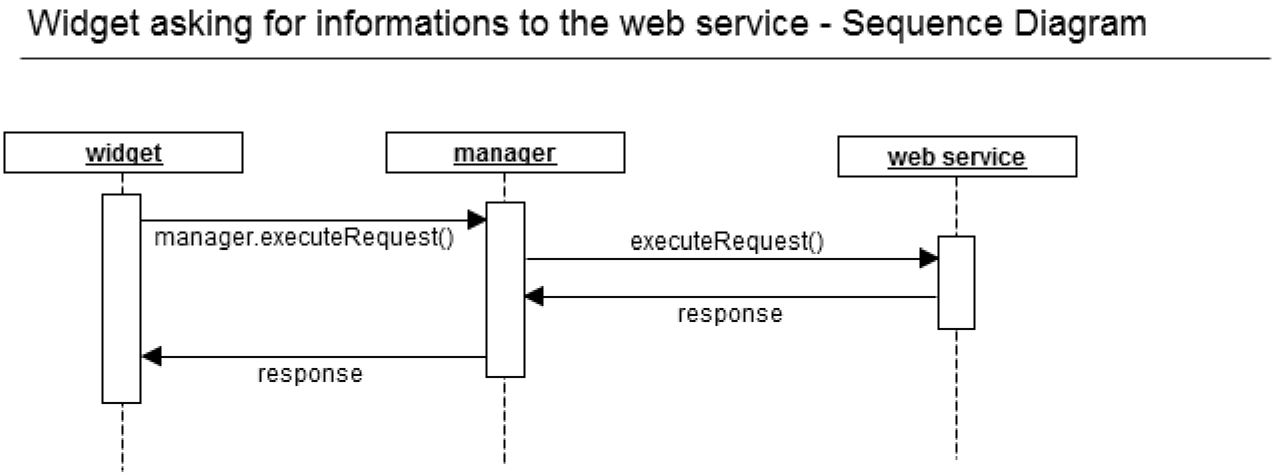
Sequence diagram

Execution of a search query



1. The element (widget, external user or program) triggers a request to the search web service from the manager using the method **makeRequest()**
2. The manager first calls the **beforeRequest()** method of all the widgets and modules it controls. Thus the different widget can, for example, add parameters to the search query
3. The manager then executes the search query on the web service using the method **executeRequest()** and wait for its response
4. When the manager receives the response, it stores it into a local variable (reponse) accessible from all the widgets and modules it controls
5. Finally, the manager executes the **afterRequest()** method of all the widgets and modules it controls, thus, they can update their data

Widget calling a web service



In order to illustrate this example, I will take the case of the AutocompleteWidget

1. A user types some text in an input field which has an autocomplete widget attached to it
2. The autocomplete widget retrieves the data entered by the user into the field and triggers the **executeRequest()** method from its manager to retrive the suggestions
3. The manager answers to the autocomplete widget when it receives the results
4. The autocomplete widget displays the results

Quickly changing the main colors

To customize the main colors of the UI, you should modify *results-color.css* and *color.css* :

- To change the green that it is used as background, you should change the attribute background of .bc-color in *color.css*. You should also change the footer.bc-color. We have separated the color of the footer from the others because of the effect of the shadow that impacts the apparent color.
- To change the green used in the pagination and the checkbox of the facets, you should change the line 126 in *color.css* where the css selectors are :

```
#facets .checkboxIcon,
#facets_mobile input[type="checkbox"]:checked + label
span.checkboxIcon,
.pagerModule .pages .page,
.pagerModule i
```

- To change the green used in the Searchbox icons (all words, at least one word), edit the file *check_radio_sheet.png* located in images folder with a graphics editor.
- To change the color of each document's link, edit the file results-color.css. Change the color attribue at the line :

```
.resultWidget .doc .res .address { color: #378140; }
```

Tutorial

This demonstration is an introduction to the framework AjaxFranceLabs and will demonstrate how to use it.

In order to do that, you will need to download the AjaxFranceLabs framework and include it to your workspace.

Here are the first steps:

1. Download the latest update of the version 1 of jQuery on jQuery's website
2. Create a **main.js** file where we will add all our code to create the interface
3. Include this two JavaScript files to the html page
4. Include the **main.css** file, the result.css from the framework and the font awesome library as bellow.

In a first place, we will create the AjaxFranceLabs manager, configure it and include the searchBar widget which will allow us to process searches using keywords.
We are going to include the required files as well as their dependencies,

```
<!-- index.html - Adding a SearchBar widget -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
```

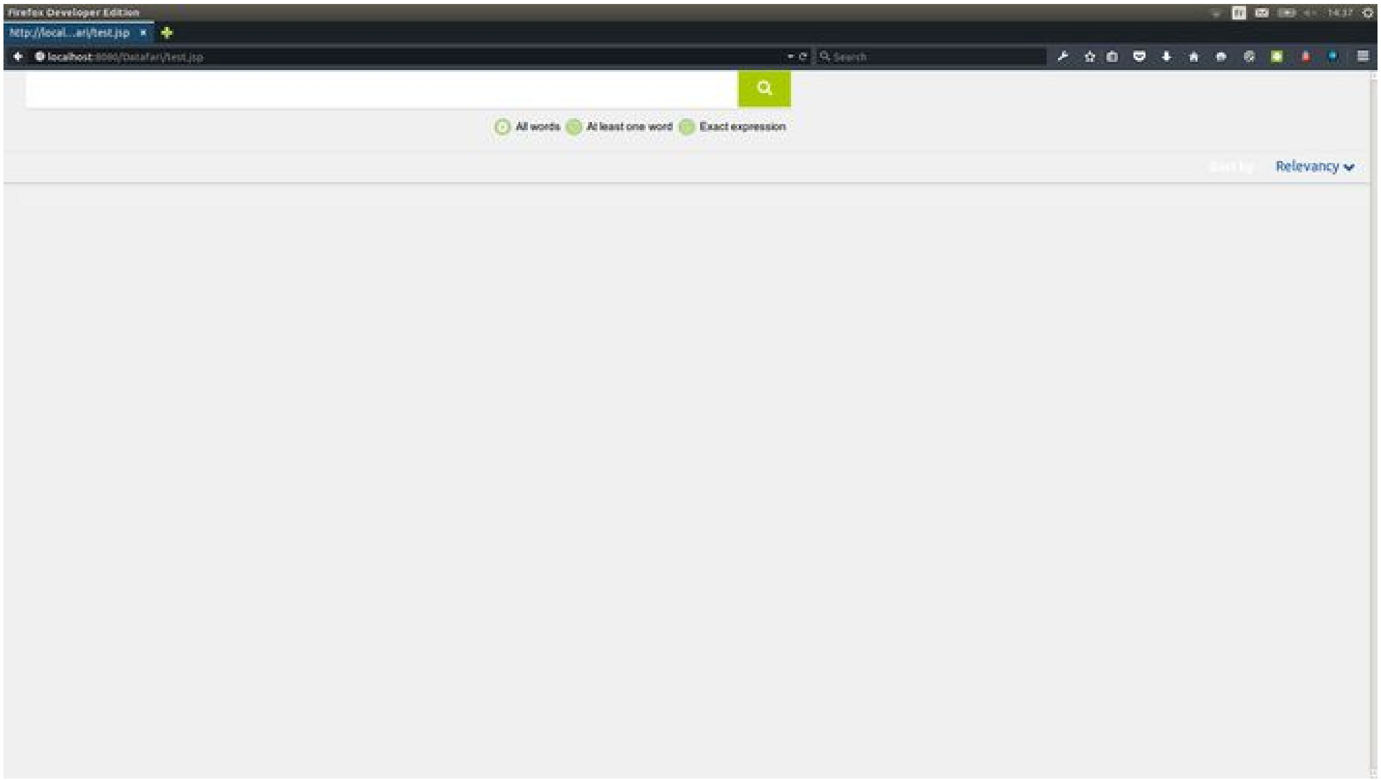
```
css">

  </head>
  <body>

    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
    <script type="text/javascript"
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <!-- The framework main file -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <!-- Required file for the manager -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
    <!-- Required file for the manager -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <!-- The abstract manager -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <!-- An implementation of the manager -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <!-- The AbstractWidget file -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <!-- The SearchBar widget, implementing AbstractWidget -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
          <div id="solr">
            <div id="searchBar"></div><!-- the widget will be inserted
here -->
```

```
          </div>
        </body>
    </html>
```

Now let's create the **main.js** file .

```
/* main.js */
$(function($){
 /* Now, we had the widget to the manager */
 //Version 1 - if you do not need to have a variable refering to the
widget
 Manager.addWidget(
     new AjaxFranceLabs.SearchBarWidget({
      elm: $('#searchBar'),
         id: 'searchBar'
     })
 );

    //Version 2
    var searchBarWidget = new AjaxFranceLabs.SearchBarWidget({
        elm: $('#searchBar'),
        id: 'searchBar'
    });
    Manager.addWidget(searchBarWidget);

    /* And we initialize the manager */
    Manager.init();
});
```

The display should look like something like this:


Firefox Developer Edition
http://local...ari/test.jsp
localhost:8080/Datafari/test.jsp
Search
All words
At least one word
Exact expression
Sort by
Relevancy

Well, now it would be good to show some results. Lets add the ResultWidget!

In the html file

```
<!-- index.html - Adding a Result widget -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
css">
  </head>
  <body>
    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
    <script type="text/javascript"
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
```

```
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <!-- The ResultWidget -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Result.widget.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
    <div id="solr">
      <div id="searchBar"></div><!-- the widget will be inserted here
-->
      <div class="col left"></div>
      <div class="col right">
    <div id="results"></div><!-- the widget will be inserted here -->
      </div>
      <div class="clear"></div>
    </div>
  </body>
```

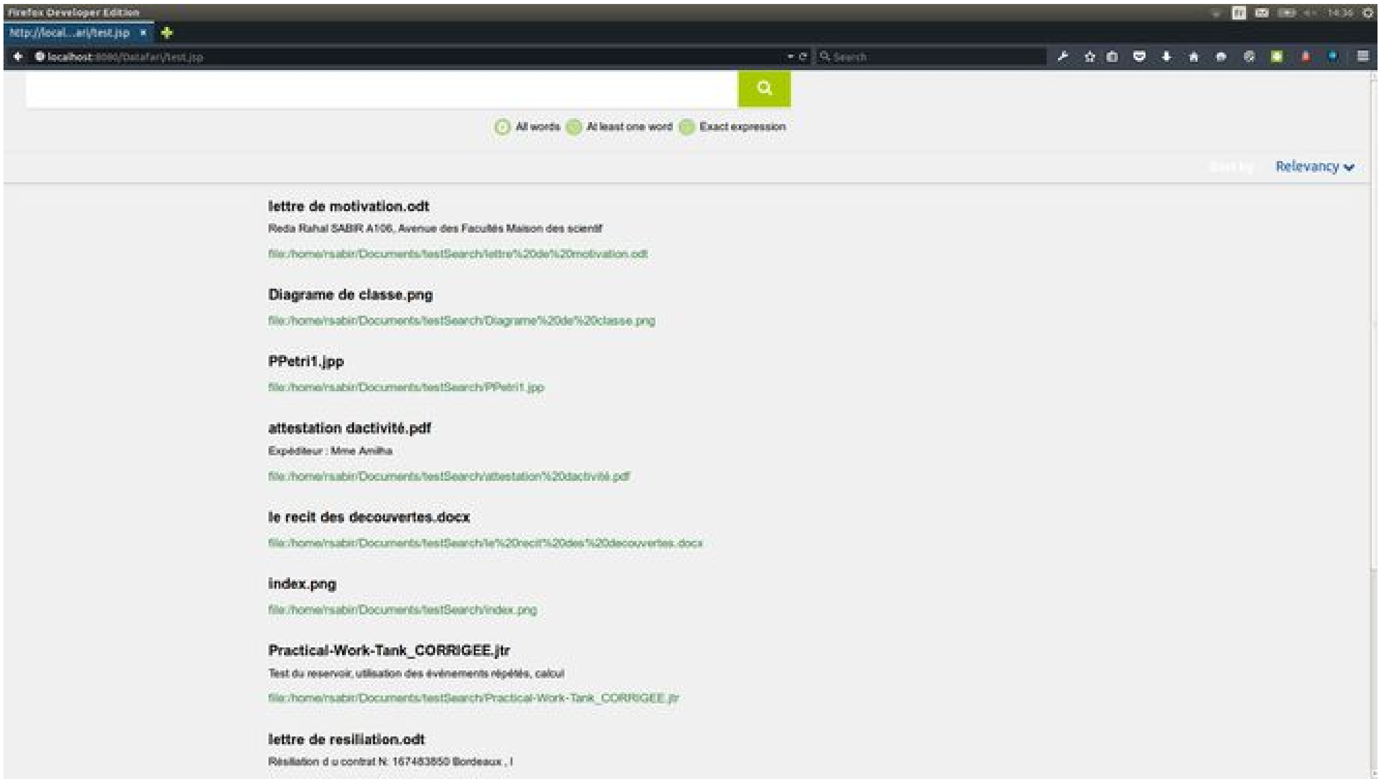
```
</html>
```

And the javascript file:

```
/* searchTutorial.js */
$(function($){

    Manager.addWidget(new AjaxFranceLabs.SearchBarWidget({
        elm: $('#searchBar'),
        id: 'searchBar'
    }));
    /* The result widget */
    Manager.addWidget(new AjaxFranceLabs.ResultWidget({
        elm: $('#results'),
        id: 'documents',
        afterRequest : function() { /* implements the method displaying
the documents */
            var data = this.manager.response;
            $(this.elm).find('.doc_list').empty();
            if (data.response.numFound === 0) {
                $(this.elm).find('.doc_list').append('<span
class="noResult">No document found.</span>');
            } else {
                var self = this;
                $.each(data.response.docs, function(i, doc) {
                    var title = (doc.title === undefined) ? doc.url :
doc.title;
                    if(data.highlighting === undefined ||
(data.highlighting[doc.id].content_fr === undefined &&
data.highlighting[doc.id].content_en === undefined)){
                        if(doc.content_fr)
                            content = doc.content_fr;
                        else
                            content = doc.content_en;
                        content = AjaxFranceLabs.truncate(content, 200)
                    }
                    else {
                        if(data.highlighting[doc.id].content_fr)
                            content =
data.highlighting[doc.id].content_fr;
                        else
                            content =
data.highlighting[doc.id].content_en;
                    }
                    $(self.elm).find('.doc_list').append('<div
class="doc e-' + i + '" doc_id="'+doc.id+'">');
                    $(self.elm).find('.doc:last').append('<div
class="res">');
                    $(self.elm).find('.doc:last .res')
                        .append('<a class="title" target="_blank"
href="' + doc.url + '">')
                        .append('<p class="description">')
                        .append('<p class="address">');
                    $(self.elm).find('.doc:last .title')
                        .append('<span>' + title + '</span>');
                    $(self.elm).find('.doc:last .description')
                        .append('<span>' + content + '</span>');
                    $(self.elm).find('.doc:last .address')
                        .append('<span>' +
AjaxFranceLabs.tinyUrl(doc.url) + '</span>');
                });

AjaxFranceLabs.addMultiElementClasses($(this.elm).find('.doc'));
                if (this.pagination !== false) {
                    this.pagination.afterRequest(data);
                }
            }
        }
    }));

    /* Lets simulate an empty search query in order to fill the results
when loading the page */
    /* The makeRequest method actually initializes the manager if this
has not been done before */
    Manager.makeRequest();
```

```
});
```

You should get the following result:


Firefox Developer Edition
http://local...ari/test.jsp
localhost:8080/Datafari/test.jsp
Search
○ All words ○ At least one word ○ Exact expression
Sort by Relevancy

lettre de motivation.odt
Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif
file:/home/rsabir/Documents/testSearch/lettre%20de%20motivation.odt

Diagrame de classe.png
file:/home/rsabir/Documents/testSearch/Diagrame%20de%20classe.png

PPetri1.jpp
file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf
Expéditeur : Mme Amiha
file:/home/rsabir/Documents/testSearch/attestation%20dactivité.pdf

le recit des decouvertes.docx
file:/home/rsabir/Documents/testSearch/le%20recit%20des%20decouvertes.docx

index.png
file:/home/rsabir/Documents/testSearch/index.png

Practical-Work-Tank_CORRIGEE.jtr
Test du reservoir, utilisation des événements répétés, calcul
file:/home/rsabir/Documents/testSearch/Practical-Work-Tank_CORRIGEE.jtr

lettre de resiliation.odt
Résiliation d u contrat N: 167483850 Bordeaux , l

Now we have our search bar and the results displayed, why not adding the search informations like the number of documents returned, the query execution time. Lets add the SearchInformation widget.

The html file

```
<!-- index.html - Adding a SearchInformation widget -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
css">
  </head>
  <body>

    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
    <script type="text/javascript"
```

```
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Result.widget.js"></script>
    <!-- The SearchInformationWidget -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchInformation.widget.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
    <div id="solr">
       <div id="searchBar"></div>
       <div id="result_information"></div><!-- the widget will be
inserted here -->
       <div class="col left"></div>
       <div class="col right">
        <div id="results"></div>
       </div>
       <div class="clear"></div>
    </div>
  </body>
</html>
```

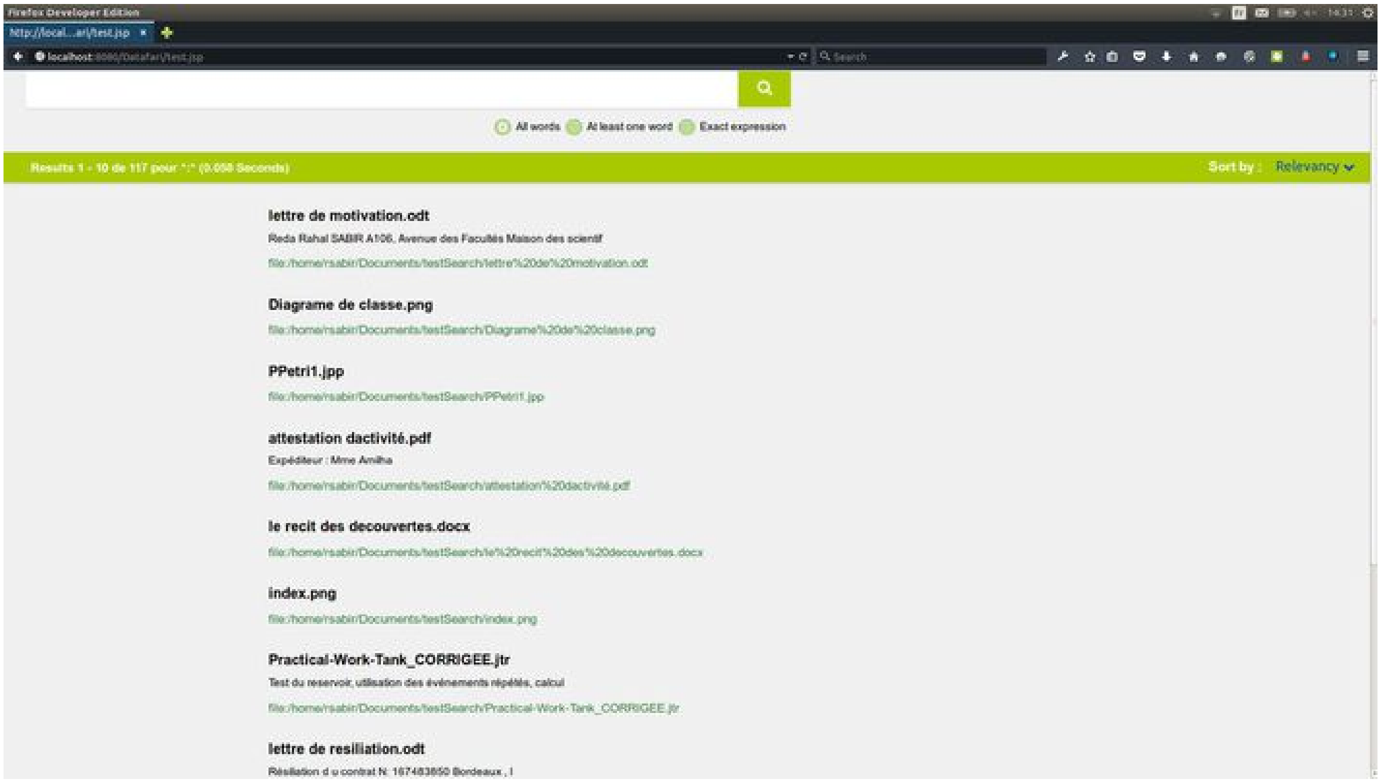
And the javascript file :

```
/* searchTutorial.js */
$(function($){

    Manager.addWidget(new AjaxFranceLabs.SearchBarWidget({
        elm: $('#searchBar'),
        id: 'searchBar'
    }));

    Manager.addWidget(new AjaxFranceLabs.ResultWidget({
        elm: $('#results'),
        id: 'documents',
        afterRequest : function() { /* implements the method displaying
the documents */
            var data = this.manager.response;
            $(this.elm).find('.doc_list').empty();
            if (data.response.numFound === 0) {
                $(this.elm).find('.doc_list').append('<span
class="noResult">No document found.</span>');
            } else {
                var self = this;
                $.each(data.response.docs, function(i, doc) {
                    var title = (doc.title === undefined) ? doc.url :
doc.title;
                    if(data.highlighting === undefined ||
(data.highlighting[doc.id].content_fr === undefined &&
data.highlighting[doc.id].content_en === undefined)){
                        if(doc.content_fr)
                            content = doc.content_fr;
                        else
                            content = doc.content_en;
                        content = AjaxFranceLabs.truncate(content, 200)
                    }
                    else {
                        if(data.highlighting[doc.id].content_fr)
                            content =
data.highlighting[doc.id].content_fr;
                        else
                            content =
data.highlighting[doc.id].content_en;
                    }
                    $(self.elm).find('.doc_list').append('<div
class="doc e-' + i + '" doc_id="'+doc.id+'">');
```

```
                    $(self.elm).find('.doc:last').append('<div
class="res">');
                    $(self.elm).find('.doc:last .res')
                        .append('<a class="title" target="_blank"
href="' + doc.url + '">')
                        .append('<p class="description">')
                        .append('<p class="address">');
                    $(self.elm).find('.doc:last .title')
                        .append('<span>' + title + '</span>');
                    $(self.elm).find('.doc:last .description')
                        .append('<span>' + content + '</span>');
                    $(self.elm).find('.doc:last .address')
                        .append('<span>' +
AjaxFranceLabs.tinyUrl(doc.url) + '</span>');
                });

AjaxFranceLabs.addMultiElementClasses($(this.elm).find('.doc'));
                if (this.pagination !== false) {
                    this.pagination.afterRequest(data);
                }
            }
        }
    }));

    /* The SearchInformation widget */
    Manager.addWidget(new AjaxFranceLabs.SearchInformationWidget({
        elm: $('#result_information'),
        id: 'searchInformation'
    }));

    Manager.makeRequest();
```

```
});
```

You should get the following display:



Firefox Developer Edition
http://local...ari/test.jsp
localhost:8080/Datafari/test.jsp
Search
All words
At least one word
Exact expression
Sort by :
Relevancy

Results 1 - 10 de 117 pour *:* (0.058 Seconds)

lettre de motivation.odt
Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif
file:/home/rsabir/Documents/testSearch/lettre%20de%20motivation.odt

Diagrame de classe.png
file:/home/rsabir/Documents/testSearch/Diagrame%20de%20classe.png

PPetri1.jpp
file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf
Expéditeur : Mme Amiha
file:/home/rsabir/Documents/testSearch/attestation%20dactivité.pdf

le recit des decouvertes.docx
file:/home/rsabir/Documents/testSearch/le%20recit%20des%20decouvertes.docx

index.png
file:/home/rsabir/Documents/testSearch/index.png

Practical-Work-Tank_CORRIGEE.jtr
Test du reservoir, utilisation des événements répétés, calcul
file:/home/rsabir/Documents/testSearch/Practical-Work-Tank_CORRIGEE.jtr

lettre de resiliation.odt
Résiliation d u contrat N: 167483850 Bordeaux , l

For this part of the tutorial, we will continue by adding the pagination feature.

The html file:

```
<!-- index.html - Adding the pagination  -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
css">
  </head>
  <body>
    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
    <script type="text/javascript"
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
```

```
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <!-- Add this class -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractModule.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <!-- Add this module -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/modules/Pager.module.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Result.widget.js"></script>
    <!-- The SearchInformationWidget -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchInformation.widget.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
    <div id="solr">
      <div id="searchBar"></div>
      <div id="result_information"></div> <!-- the widget will be
inserted here -->
      <div class="col left"></div>
      <div class="col right">
    <div id="results"></div>
      </div>
      <div class="clear"></div>
    </div>
  </body>
```

```
</html>
```

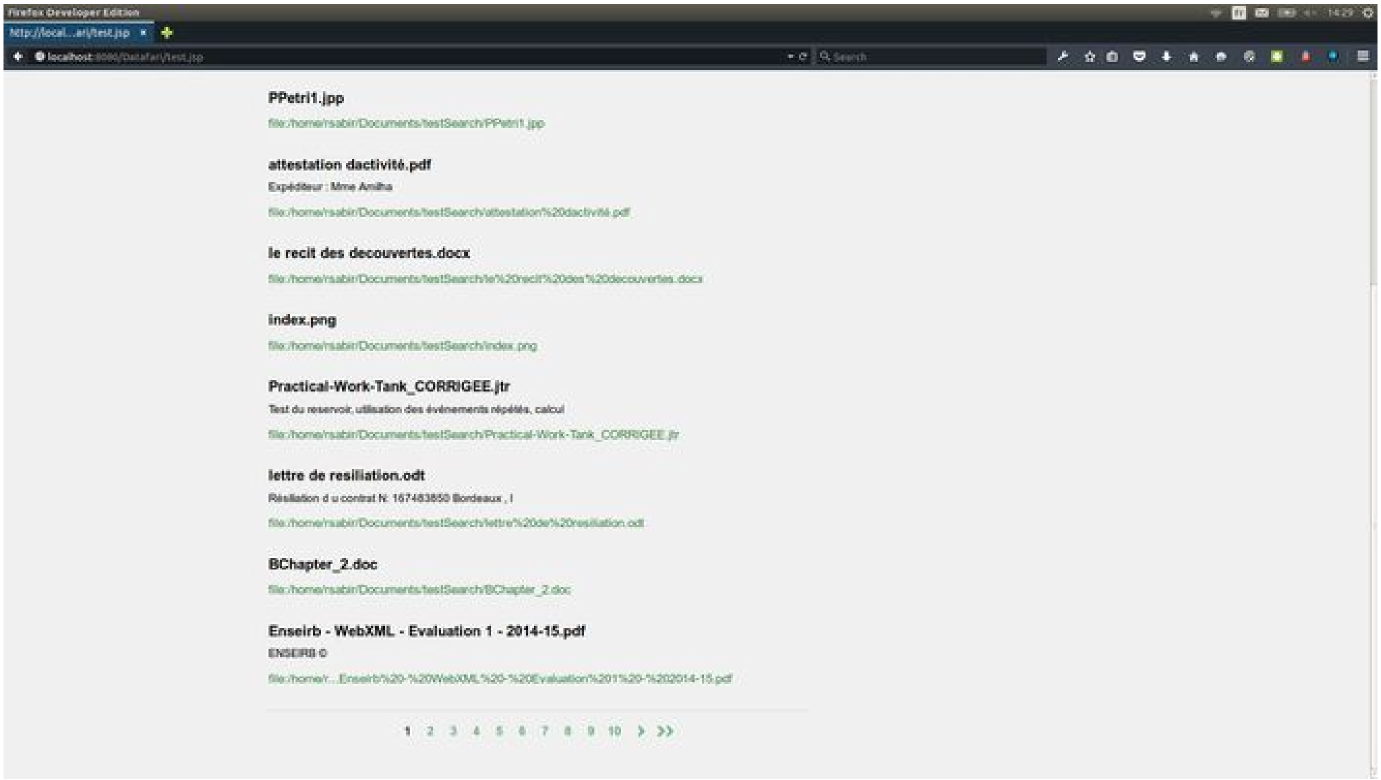
And the javascript file

```
/* searchTutorial.js */
$(function($){

Manager.addWidget(new AjaxFranceLabs.SearchBarWidget({
    elm: $('#searchBar'),
    id: 'searchBar'
}));
/* The result widget */
Manager.addWidget(new AjaxFranceLabs.ResultWidget({
    elm: $('#results'),
    id: 'documents',
    pagination : true, /* Add this */
    afterRequest : function() {
        var data = this.manager.response;
        $(this.elm).find('.doc_list').empty();
        if (data.response.numFound === 0) {
            $(this.elm).find('.doc_list').append('<span
class="noResult">No document found.</span>');
        } else {
            var self = this;
            $.each(data.response.docs, function(i, doc) {
                var title = (doc.title === undefined) ? doc.url :
doc.title;
                if(data.highlighting === undefined ||
(data.highlighting[doc.id].content_fr === undefined &&
data.highlighting[doc.id].content_en === undefined)){
                    if(doc.content_fr)
                        content = doc.content_fr;
                    else
                        content = doc.content_en;
                    content = AjaxFranceLabs.truncate(content, 200)
                }
                else {
                    if(data.highlighting[doc.id].content_fr)
                        content = data.highlighting[doc.id].content_fr;
                    else
                        content = data.highlighting[doc.id].content_en;
                }
                $(self.elm).find('.doc_list').append('<div class="doc
e-' + i + '" doc_id="'+doc.id+'">');
                $(self.elm).find('.doc:last').append('<div
class="res">');
                $(self.elm).find('.doc:last .res')
```

```
                    .append('<a class="title" target="_blank" href="' +
doc.url + '">')
                    .append('<p class="description">')
                    .append('<p class="address">');
                $(self.elm).find('.doc:last .title')
                    .append('<span>' + title + '</span>');
                $(self.elm).find('.doc:last .description')
                    .append('<span>' + content + '</span>');
                $(self.elm).find('.doc:last .address')
                    .append('<span>' + AjaxFranceLabs.tinyUrl(doc.url) +
'</span>');
            });

AjaxFranceLabs.addMultiElementClasses($(this.elm).find('.doc'));
            if (this.pagination !== false) {
                this.pagination.afterRequest(data);
            }
        }
    }
}));


Manager.addWidget(new AjaxFranceLabs.SearchInformationWidget({
    elm: $('#result_information'),
    id: 'searchInformation'
}));

Manager.makeRequest();
```

```
});
```

You should get the following result:



Firefox Developer Edition
http://local...ari/test.jsp
localhost:8080/Datafari/test.jsp
Search

PPetri1.jpp

file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf

Expéditeur : Mme Amiha

file:/home/rsabir/Documents/testSearch/attestation%20dactivité.pdf

le recit des decouvertes.docx

file:/home/rsabir/Documents/testSearch/le%20recit%20des%20decouvertes.docx

index.png

file:/home/rsabir/Documents/testSearch/index.png

Practical-Work-Tank_CORRIGEE.jtr

Test du reservoir, utilisation des événements répétés, calcul

file:/home/rsabir/Documents/testSearch/Practical-Work-Tank_CORRIGEE.jtr

lettre de resiliation.odt

Résiliation d u contrat N: 167483850 Bordeaux , l

file:/home/rsabir/Documents/testSearch/lettre%20de%20resiliation.odt

BChapter_2.doc

file:/home/rsabir/Documents/testSearch/BChapter_2.doc

Enseirb - WebXML - Evaluation 1 - 2014-15.pdf

ENSEIRB ©

file:/home/r...Enseirb%20-%20WebXML%20-%20Evaluation%201%20-%202014-15.pdf

1 2 3 4 5 6 7 8 9 10 > >>

Easy right? Regarding to the documentation, the pagination variable can take 3 values, true, false, a PagerWidget.
By default this value is set to true and will automaticaly create a PagerWidget with the default options. But, if you want to modify some methods or values of the widget, you can create another one and assign it to the pagination variable.

The next thing we will do is including the autocomplete widget to the search bar.

The autocomplete widget is a direct extension of the autocomplete plugin from jQuery UI, so we will need to download it and include it to our javascript files.

```
<!-- index.html - Adding the autocomplete functionnality -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
css">
  </head>
  <body>
```

```
    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
    <script type="text/javascript"
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractModule.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <!-- Add this module -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/modules/Autocomplete.module.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/modules/Pager.module.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Result.widget.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchInformation.widget.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
    <div id="solr">
      <div id="searchBar"></div>
      <div id="result_information"></div><!-- the widget will be
inserted here -->
      <div class="col left"></div>
      <div class="col right">
    <div id="results"></div>
      </div>
      <div class="clear"></div>
    </div>
  </body>
```

```
</html>
```

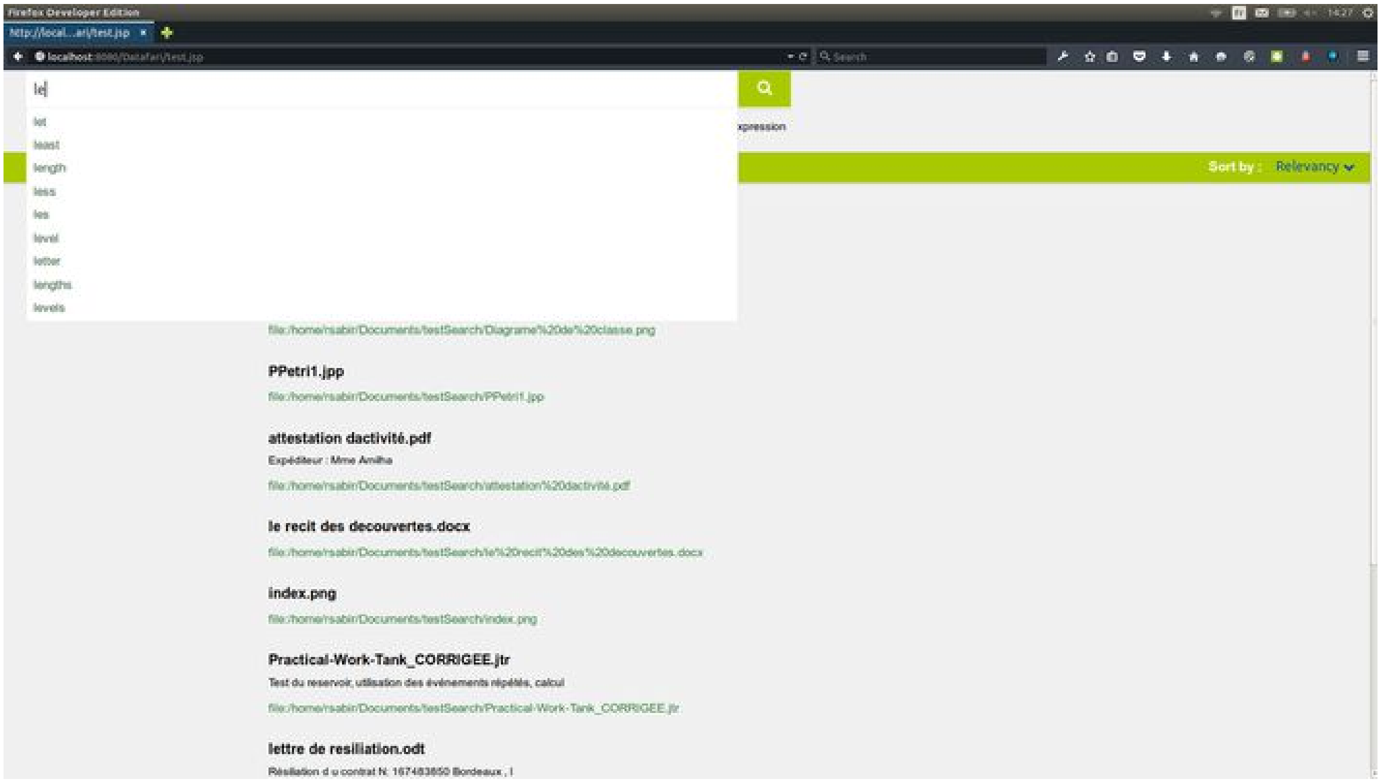
And the javascript file

```
/* searchTutorial.js */
$(function($){

    Manager.addWidget(new AjaxFranceLabs.SearchBarWidget({
        elm: $('#searchBar'),
        id: 'searchBar',
        autocomplete : true //add this line for the autocompletion
    }));

    Manager.addWidget(new AjaxFranceLabs.ResultWidget({
        elm: $('#results'),
        id: 'documents',
        pagination : true,
        afterRequest : function() {
            var data = this.manager.response;
            $(this.elm).find('.doc_list').empty();
            if (data.response.numFound === 0) {
                $(this.elm).find('.doc_list').append('<span
class="noResult">No document found.</span>');
            } else {
                var self = this;
                $.each(data.response.docs, function(i, doc) {
                    var title = (doc.title === undefined) ? doc.url :
doc.title;
                    if(data.highlighting === undefined ||
(data.highlighting[doc.id].content_fr === undefined &&
data.highlighting[doc.id].content_en === undefined)){
                        if(doc.content_fr)
                            content = doc.content_fr;
                        else
                            content = doc.content_en;
                        content = AjaxFranceLabs.truncate(content, 200)
                    }
                    else {
                        if(data.highlighting[doc.id].content_fr)
                            content =
data.highlighting[doc.id].content_fr;
                        else
                            content =
data.highlighting[doc.id].content_en;
                    }
                    $(self.elm).find('.doc_list').append('<div
class="doc e-' + i + '" doc_id="'+doc.id+'">');
```

```
                    $(self.elm).find('.doc:last').append('<div
class="res">');
                    $(self.elm).find('.doc:last .res')
                        .append('<a class="title" target="_blank"
href="' + doc.url + '">')
                        .append('<p class="description">')
                        .append('<p class="address">');
                    $(self.elm).find('.doc:last .title')
                        .append('<span>' + title + '</span>');
                    $(self.elm).find('.doc:last .description')
                        .append('<span>' + content + '</span>');
                    $(self.elm).find('.doc:last .address')
                        .append('<span>' +
AjaxFranceLabs.tinyUrl(doc.url) + '</span>');
                });

AjaxFranceLabs.addMultiElementClasses($(this.elm).find('.doc'));
                if (this.pagination !== false) {
                    this.pagination.afterRequest(data);
                }
            }
        }
    }));


    Manager.addWidget(new AjaxFranceLabs.SearchInformationWidget({
        elm: $('#result_information'),
        id: 'searchInformation'
    }));

    Manager.makeRequest();
```

```
});
```

You should get the following:


Firefox Developer Edition
http://local...ari/test.jsp
localhost:8080/Datafari/test.jsp
Search
le
let
least
length
less
les
level
letter
lengths
levels
xpression
Sort by : Relevancy

file:/home/rsabir/Documents/testSearch/Diagrame%20de%20classe.png

PPetri1.jpp

file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf

Expéditeur : Mme Amiha

file:/home/rsabir/Documents/testSearch/attestation%20dactivité.pdf

le recit des decouvertes.docx

file:/home/rsabir/Documents/testSearch/le%20recit%20des%20decouvertes.docx

index.png

file:/home/rsabir/Documents/testSearch/index.png

Practical-Work-Tank_CORRIGEE.jtr

Test du reservoir, utilisation des événements répétés, calcul

file:/home/rsabir/Documents/testSearch/Practical-Work-Tank_CORRIGEE.jtr

lettre de resiliation.odt

Résiliation d u contrat N: 167483850 Bordeaux , l

Easy right? Regarding to the documentation, the pagination variable can take 3 values, true, false, a PagerWidget.
By default this value is set to true and will automaticaly create a PagerWidget with the default options. But, if you want to modify some methods or values of the widget, you can create another one and assign it to the pagination variable.

Lets go a bit further now and add faceting.

The html file

```
<!-- index.html - Adding some facets -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet" type="text/css" href="css/results.css">
    <link rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.
css">
  </head>
  <body>

    <script type="text/javascript"
src="js/jquery-1.8.1.min.js"></script>
```

```
    <script type="text/javascript"
src="js/jquery-ui-1.8.23.min.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/uuid.core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/i18njs.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Core.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/Parameter.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/ParameterStore.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractManager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/manager/Manager.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractModule.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractWidget.js"></script>
    <!-- Add this class which our widget extends -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/core/AbstractFacetWidget.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/modules/Autocomplete.module.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/modules/Pager.module.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchBar.widget.js"></script>
    <script type="text/javascript" src="js/main.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Result.widget.js"></script>
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/SearchInformation.widget.js"></script>
    <!-- Add this widget -->
    <script type="text/javascript"
src="js/AjaxFranceLabs/widgets/Table.widget.js"></script>
    <script type="text/javascript" src="js/searchTutorial.js"></script>
    <div id="solr">
      <div id="searchBar"></div>
      <div id="result_information"></div>
      <div class="col left">
    <!-- Add this block -->
    <div id="facets">
      <div id="facet_langue"></div><!-- First widget -->
      <div id="facet_source"></div><!-- Second widget -->
    </div>
      </div>
      <div class="col right">
    <div id="results"></div>
```

```
      </div>
      <div class="clear"></div>
    </div>
  </body>
```

```
</html>
```

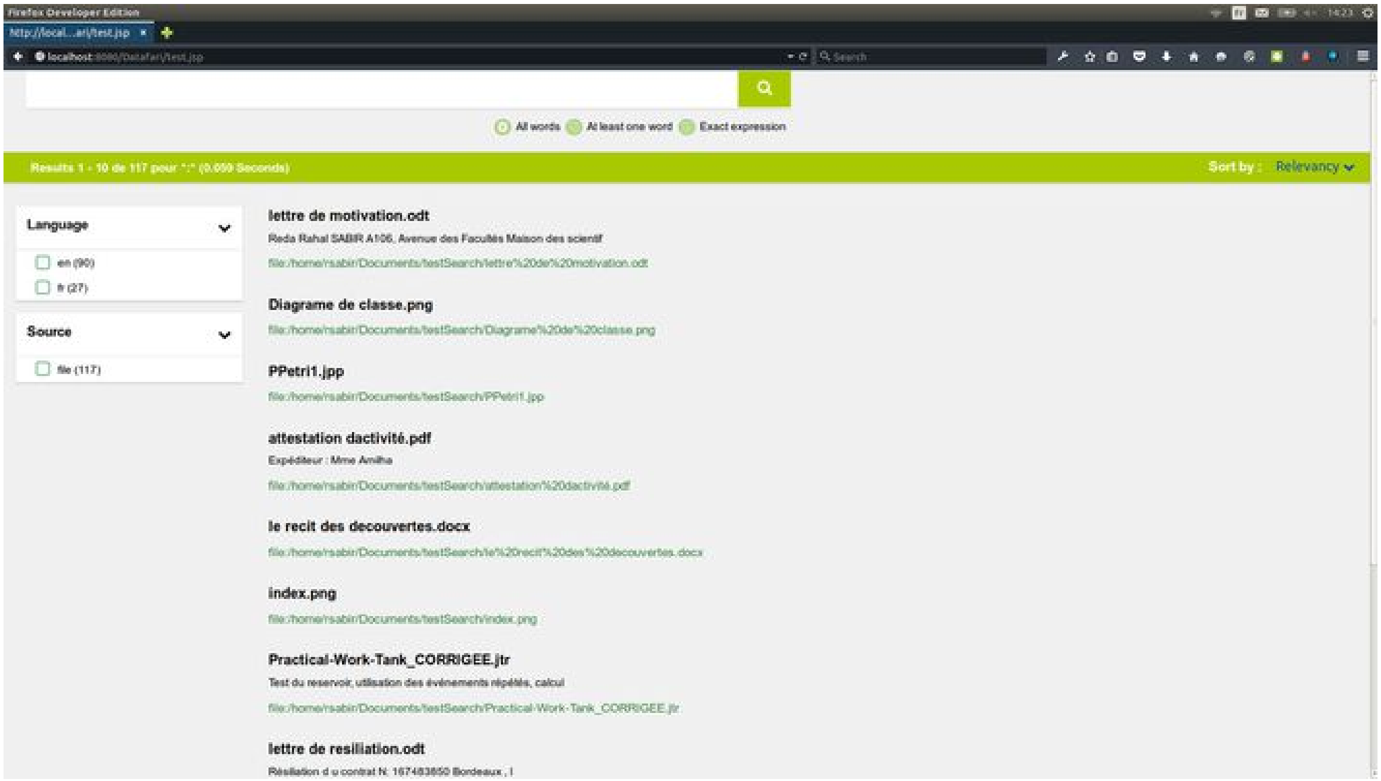
And the javascript file:

```
/* searchTutorial.js */
$(function($){

    Manager.addWidget(new AjaxFranceLabs.SearchBarWidget({
        elm: $('#searchBar'),
        id: 'searchBar',
        autocomplete : true //add this line for the autocompletion
    }));

    Manager.addWidget(new AjaxFranceLabs.ResultWidget({
        elm: $('#results'),
        id: 'documents',
        pagination : true,
        afterRequest : function() {
            var data = this.manager.response;
            $(this.elm).find('.doc_list').empty();
            if (data.response.numFound === 0) {
                $(this.elm).find('.doc_list').append('<span
class="noResult">No document found.</span>');
            } else {
                var self = this;
                $.each(data.response.docs, function(i, doc) {
                    var title = (doc.title === undefined) ? doc.url :
doc.title;
                    if(data.highlighting === undefined ||
(data.highlighting[doc.id].content_fr === undefined &&
data.highlighting[doc.id].content_en === undefined)){
                        if(doc.content_fr)
                            content = doc.content_fr;
                        else
                            content = doc.content_en;
                        content = AjaxFranceLabs.truncate(content, 200)
                    }
                    else {
                        if(data.highlighting[doc.id].content_fr)
                            content =
data.highlighting[doc.id].content_fr;
                        else
                            content =
data.highlighting[doc.id].content_en;
                    }
                    $(self.elm).find('.doc_list').append('<div
class="doc e-' + i + '" doc_id="'+doc.id+'">');
```

```
                    $(self.elm).find('.doc:last').append('<div
class="res">');
                    $(self.elm).find('.doc:last .res')
                        .append('<a class="title" target="_blank"
href="' + doc.url + '">')
                        .append('<p class="description">')
                        .append('<p class="address">');
                    $(self.elm).find('.doc:last .title')
                        .append('<span>' + title + '</span>');
                    $(self.elm).find('.doc:last .description')
                        .append('<span>' + content + '</span>');
                    $(self.elm).find('.doc:last .address')
                        .append('<span>' +
AjaxFranceLabs.tinyUrl(doc.url) + '</span>');
                });

AjaxFranceLabs.addMultiElementClasses($(this.elm).find('.doc'));
                if (this.pagination !== false) {
                    this.pagination.afterRequest(data);
                }
            }
        }
    }));


    Manager.addWidget(new AjaxFranceLabs.SearchInformationWidget({
        elm: $('#result_information'),
        id: 'searchInformation'
    }));

    Manager.addWidget(new AjaxFranceLabs.TableWidget({
        elm : $('#facet_langue'),
        id : 'facet_langue',
        field : 'language',
        name : window.i18n.msgStore['language'],
        pagination : true,
        selectionType : 'OR',
        returnUnselectedFacetValues : true
    }));

Manager.addWidget(new AjaxFranceLabs.TableWidget({
        elm : $('#facet_source'),
        id : 'facet_source',
        field : 'source',
        name : window.i18n.msgStore['source'],
        pagination : true,
        selectionType : 'OR',
        returnUnselectedFacetValues : true
    }));
    Manager.makeRequest();
```

```
  });
```

You should get the following:



Results 1 - 10 de 117 pour "*:*" (0.059 Seconds)

lettre de motivation.odt
Reda Rahal SABIR A106, Avenue des Facultés Maison des scientif
file:/home/rsabir/Documents/testSearch/lettre%20de%20motivation.odt

Diagrame de classe.png
file:/home/rsabir/Documents/testSearch/Diagrame%20de%20classe.png

PPetri1.jpp
file:/home/rsabir/Documents/testSearch/PPetri1.jpp

attestation dactivité.pdf
Expéditeur : Mme Amiha
file:/home/rsabir/Documents/testSearch/attestation%20dactivité.pdf

le recit des decouvertes.docx
file:/home/rsabir/Documents/testSearch/le%20recit%20des%20decouvertes.docx

index.png
file:/home/rsabir/Documents/testSearch/index.png

Practical-Work-Tank_CORRIGEE.jtr
Test du reservoir, utilisation des événements répétés, calcul
file:/home/rsabir/Documents/testSearch/Practical-Work-Tank_CORRIGEE.jtr

lettre de resiliation.odt
Résiliation d u contrat N: 167483850 Bordeaux , l

Widgets and Modules

SearchBar widget

A widget displaying a search bar and search options.

It contains a link useful to switch to advanced search (if that widget is activated): : in this case a default query is performed and the inputs and URL reset.

Variables

- **autocomplete:** can take 3 values:
  - **false(default):** no autocomplete.
  - **true:** autocomplete automaticaly created.
  - **AutocomepleteWidget:** an autocomplete widget.
- **autocompleteOptions:** options for the autocomplete widget if autocomplete set to true. Check the code to see the default structure.
- **removeContentButton:** if set to true, creates a button which appears when there is text in the search bar in order to remove it and execute a request (default: false).
- **noRequest:** if true, inhibits the request to be sent to Solr (default: false).
- **updateBrowserAddressBar:** if true, the address bar URL is updated with the query parameter, e.g. ?searchType=AllWords&query=datafari (default: true).
- **removeContentButton:** if true, adds a button that clears the search bar (default: false).

Methods

- **buildWidget:** an implementation of the parent class method. It contains the onClick event handler for the advanced search link.
- **beforeRequest:** an implementation of the parent class method.

- **afterRequest:** an implementation of the parent class method; closes the autocomplete menu.
- **reset:** an implementation of the parent class method: it resets the inputs of the widget (useful when switching between basic and advanced search).
- **updateAddressBar:** updates the address bar URL with the search term, e.g. ?searchType=AllWords&query=datafari
- **clean:** resets the pagination and the fq and start parameters of the manager's store.
- **makeRequest:** if the noRequest variable is false, cleans the results area and facets, updates the address bar and performs the search request.

Note that the Autocomplete widget is also an extension of the Autocomplete plugin from jQuery UI and so it has all its variables and methods too.

Result widget

A widget displaying the search result.

Variables

- **pagination:** can take 3 values
  - **false(default):** no pagination.
  - **true:** pagination automaticaly created.
  - **PagerWidget:** a pager widget.

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

SubClassResult widget

A widget that implements Datafari specific code and parameters the Result widget. We did not test this widget towards a direct Solr system. This widget displays the search result. It displays results differently depending on the activation of the LikesAndFavorites widget.

Variables

- **pagination:** can take 3 values
  - **false(default):** no pagination.
  - **true:** pagination automaticaly created.
  - **PagerWidget:** a pager widget.
- **elmSelector**: used to store the results to be displayed. Corresponds to the jQuery selector.
- **firstTimeWayPoint**: used in the mobile mode, to know when we reach the bottom of the page, to trigger the spinner in order to load and display the further results.
- **isMobile**: checks whether we are in mobile mode or not.
- **mutex_locked :** is a variable used as a mutex for the mobile mode

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

ResultIllustrated widget

A widget that extend Result Widget and that displays image in result.

Variables

- **pagination:** can take 3 values
    - **false(default):** no pagination.
    - **true:** pagination automaticaly created.
    - **PagerWidget:** a pager widget.

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

LikesAndFavorites Widget

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- function that is runned after getting the likes and favorites of the user
- **showError:** function that analyze the error and display the corresponding error message

---

SearchInformation widget

A widget displaying the search informations like the number of document found, the query execution time, ...

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

Table widget

A widget displaying a facet values in a table.

Variables

- **name:** a name for the facet to be displayed.
- **pagination:** can take 3 values:
    - **false(default):** no pagination.
    - **true:** pagination automaticaly created.
    - **PagerWidget:** a pager widget.
- **nbElmToDisplay:** the number of facets to display (default: 10).
- **sort:** can take 3 values
    - **occurences(default):** sort the facets by the number of occurences.
    - **AtoZ:** sort the facets name by alphabetical order.
    - **ZtoA:** sort the facets name in a reversed akphabetical order.
- **maxDisplay:** maximum number of facets (default: 40).
- **checkedOnTop:** if set to true, the checked facets will be sorted on the top before the unchecked facets (default: true).
- **mappingValues (optional):** used to override the values of the facet by custom ones. This variable has to be initialized in the constructor call (search.js) and respect the following format : {originalFacetValue1 : customValue1, originalFacetValue2 : customValue2, ... }

All the possible facet values don't have to be mapped to custom ones, you can fill this optional variable with the only needed ones, the other values will stay the same.

Methods

- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **update:** a method updating the content of the widget.
- **assocTags:** a method maping the search query facets result.
- **sortBy:** the method sorting the facets, takes the sort mode as parameter.
- **clickHandler:** a callback called when a facet is selected or unslected.

FacetDuplicates widget :

This widget displays duplicate documents that are in the index. It is very specific, not recommended for instance if you have an ecommerce website. It is more useful in an intranet context, where you may be interested to know which documents are present several times in your repositories.

The class is of type « final ». For each hash of a given document, the widget retrieves the corresponding document name (the corresponding field name for the document name needs to be identified. By default in Datafari, it is *title_en or title_fr*). For more information, refer to the specific deduplication documentation.

Variables

- **name:** a name for the facet to be displayed.
- **pagination:** can take 3 values:
  - **false(default):** no pagination.
  - **true:** pagination automaticaly created.
  - **PagerWidget:** a pager widget.
- **nbElmToDisplay:** the number of facets to display (default: 10).
- **sort:** can take 3 values
  - **occurences(default):** sort the facets by the number of occurences.
  - **AtoZ:** sort the facets name by alphabetical order.
  - **ZtoA:** sort the facets name in a reversed akphabetical order.
- **maxDisplay:** maximum number of facets (default: 40).
- **checkedOnTop:** if set to true, the checked facets will be sorted on the top before the unchecked facets (default: true).

Methods

- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **update:** sends an ajax request for each hash to get the name of the document. It will also build the html that will be used to display the facet. If there is no duplicate document, the facet will not be displayed.
- **assocTags:** a method maping the search query facets result.
- **sortBy:** the method sorting the facets, takes the sort mode as parameter.
- **clickHandler:** a callback called when a facet is selected or unslected.

HierarchicalFacet widget

A widget displaying a facet values in a hierarchical way. Available as of Datafari v2.2.

Prerequisites

In order to make this widget work, it is mandatory that the selected facet field represents a path tokenization where each value is prefixed by its

depth level.
For example, the path '/home/france/labs' will be tokenized as this : 0/home, 1/home/france, 2/home/france/labs

Variables

- **name:** a name for the facet to be displayed.
- **pagination:** can take 3 values:
  - **false(default):** no pagination.
  - **true:** pagination automaticaly created.
  - **PagerWidget:** a pager widget.
- **nbElmToDisplay:** the number of facets to display (default: 10).
- **sort:** can take 3 values
  - **occurences(default):** sort the facets by the number of occurences.
  - **AtoZ:** sort the facets name by alphabetical order.
  - **ZtoA:** sort the facets name in a reversed akphabetical order.
- **maxDisplay:** maximum number of facets (default: 40).
- **checkedOnTop:** if set to true, the checked facets will be sorted on the top above the unchecked facets (default: true).
- **rootLevel:** the root depth level of the hierarchical tree (default: 0)
- **maxDepth:** the maximum depth to be displayed, starting from the rootLevel (default: 3)
- **separator:** the separator used between each depth level (default '/')

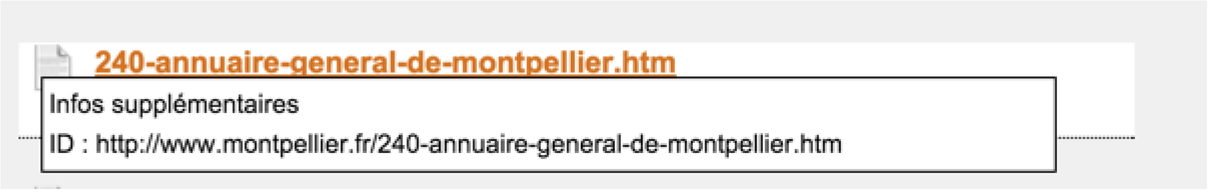
Methods

- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **update:** a method updating the content of the widget.
- **assocTags:** a method mapping the search query facets result.
- **sortBy:** the method sorting the facets, takes the sort mode as parameter.
- **clickHandler:** a callback called when a facet is selected or unslected.
- **selectNoRequest:** a method that does the same as the selectHandler parent class method but that doesn't perform the request at the end
- **unselectNoRequest:** a method that does the same as the unselectHandler parent class method but that doesn't perform the request at the end
- **displayLevel:** a method which constructs the hierarchy tree with the search query facets result

---

PrevisualizeResult widget

A widget that extend Result Widget and that displays a preview window at mousover on the document.



240-annuaire-general-de-montpellier.htm

Infos supplémentaires

ID : http://www.montpellier.fr/240-annuaire-general-de-montpellier.htm

Prerequisites

In order to make this widget work, you have to customize the content that you want in the window and to activate the widget.

To activate the widget :

- in datafari/WebContent/js/search.js :

Uncomment the line :

```
Manager.addWidget(new AjaxFranceLabs.PrevisualizeResultWidget());
```

- in datafari/WebContent/searchView.jsp

Uncomment the lines :

```
<script
type="text/javascript" src="js/AjaxFranceLabs/widgets/PrevisualizeResult
.widget.js" charset="utf-8"></script>
```

and

```
<div id="previsualize"></div>
```

To customize the information displayed :

Go to datafari/WebContent/js/AjaxFranceLabs/widgets/PrevisualizeResult.widget.js and edit the line to display whatever you want instead of the ID of the document :

```
$($('.doc_list .previsualizetemplate')[index]).append('<div
id="previsualizeid">ID : '+docs[index].id+' </div>' );
```

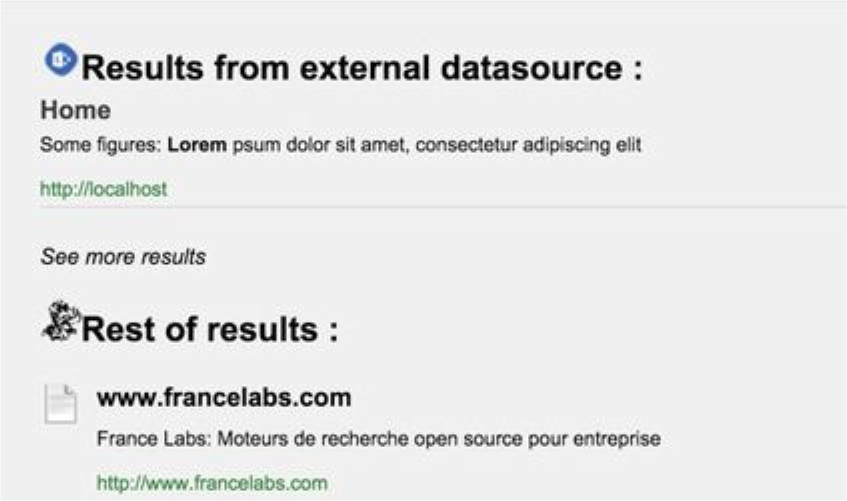
Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **showError:** function that analyze the error and display the corresponding error message

---

ExternalResult widget

A widget that allows to have a federated search in Datafari ie display results from another search engine (Solr, Sharepoint for example) and display below the normal results from Datafari.



Results from external datasource :

Home

Some figures: **Lorem** psum dolor sit amet, consectetur adipiscing elit

http://localhost

See more results

Rest of results :

www.francelabs.com

France Labs: Moteurs de recherche open source pour entreprise

http://www.francelabs.com

Prerequisites

In order to make this widget work, you have to parse the response from your external datasource.

To activate the widget :

- in datafari/WebContent/js/search.js :

Uncomment the line :

```
Manager.addWidget(new AjaxFranceLabs.ExternalResultWidget());
```

- in datafari/WebContent/searchView.jsp

Uncomment the lines :

```
<script type="text/javascript" src="js/AjaxFranceLabs/widgets/ExternalResult.widget.js" charset="utf-8"></script>
```

and

```
<div id="external"></div>
```

to select the data to parse :

- in datafari/WebContent/js/AjaxFranceLabs/widgets/ExternalResult.widget.js

Edit the line :

```
var urldatasource = 'http://localhost:8080/Datafari/externaldata.json';
```

Change the url for the data you want to parse. And adapt the code below to parse the info you want.

We provide some example data into Datafari if you want to test it out of the box :

- Copy datafari/dev-tools/example-data/externaldata.json into datafari/WebContent

Methods

- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **showError:** function that analyze the error and display the corresponding error message

---

Table Facet Queries widget

A widget displaying facet queries in a table.

Variables

- **name:** a name for the facet to be displayed.
- **field:** the field that will be used in the queries
- **queries:** the list of queries
- **pagination:** can take 3 values:
    - **false(default):** no pagination.
    - **true:** pagination automaticaly created.
    - **PagerWidget:** a pager widget.
- **nbElmToDisplay:** the number of facets to display (default: 10).
- **maxDisplay:** maximum number of facets (default: 40).

Spellcheck widget

A widget spellchecking a query string. (eg: can offer you to correct constelli**a** in constelli**o**).

This widget works with both basic and advanced search.

Note: this requires enabling a spellchecker component in the searchhandler of Solr (done in the xml config files), whether you use a pure Solr or a Constellio Solr.

Variables

There are no specific variables: only the ones inherited from AjaxFranceLabs.AbstractWidget.

Methods

- **buildWidget:** an implementation of the parent class method: builds the widget.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method: here there is the logic of the spellchecker itself. It checks whether the result of the search query has some suggestions and, if so, it calls doSpellCheckerQuery to perform a new search with the spellchecked input, contained in the collations array.
- **doSpellcheckerQuery:** performs a new search with the spellchecked input.

AdvancedSearch widget

A widget enabling you to specify on which fields to perform a search.

The result of the search query is spellchecked by SpellChecker widget (if that widget is activated).

It contains a link useful to switch to basic search (if that widget is activated): in this case a default query is performed and the inputs and URL reset.

By default the AdvancedSearch is desactivated from the splash page.

The AdvancedSearch widget is built with 2 other classes:

- **AdvancedSearchTable:** which is a part of the advanced search form, containing fields.
- **AdvancedSearchField:** which are the different fields provided by the advanced search.

Variables

- **tables:** an array containing all the tables.
- **resizable:** if set to true, display a resize bar on the bottom of the advanced search form. Requires jQuery UI Resizable plugin.
- **options:** options given to the jQuery UI Resizable plugin.
- **optionsExtend:** if set to true, extends the default parameters defined in this widget (default: true).

Methods

- **buildWidget:** an implementation of the parent class method: builds the widget and links the basic search link to the basic search bar.
- **beforeRequest:** an implementation of the parent class method: builds the search query with the inputs from AdvancedSearchFields and calls updateAddressBar.
- **addTable:** adds the AdvancedSearchTable object given in parameter to the table array.
- **reset:** an implementation of the parent class method: resets the inputs of the widget and calls the reset method on the underlying elements table and fields (useful when switching between basic and advanced search).
- **makeRequest:** cleans the results area (list and facets) and performs a default search (*:*)
- **cleanResults:** cleans the results area (list and facets)
- **updateAddressBar:** updates the URL in the address bar with the advanced search query created by beforeRequest.

AdvancedSearchTable

Extends AjaxFranceLabs.Class

Variables

- **parent:** the advancedSearch DOM element.
- **elm:** the AdvancedSearchTable DOM element itself.
- **title:** a name for the table (default: empty string).
- **description:** a description for the table (default: empty string).
- **fieldStore:** an array containing the AdvancedSearchField fields related to the table.
- **manager:** a reference to the widget's manager.

Methods

- **init:** an init method that creates the class.
- **addField:** a method adding an AdvancedSearchField field to the fieldStore array.
- **reset:** an implementation of the parent class method: resets the inputs of this class and calls the reset method on the underlying fields.

AdvancedSearchField

Extends AjaxFranceLabs.Class

Variables

- **parent:** the advancedSearchTable DOM element.
- **elm:** the AdvancedSearchField DOM element itself.
- **label:** a name for the field (default: empty string).
- **range:** if set to true, will create two inputs instead of one to enable an input range (default: false).
- **description:** a description for the field (default: empty string).

- **multiInput:** if set to true, will add a link which allows the user to add more inputs for this field (default: false).
- **addInputLabel:** text to append to the link adding inputs (default: "Add input").
- **autocomplete:** can take 3 values:
    - **false(default):** no autocomplete.
    - **true:** autocomplete automaticaly created.
    - **AutocomepleteWidget:** an autocomplete widget.
- **autocompleteOptions:** options for the autocomplete widget if autocomplete set to true. Check the code to see the default structure.
- **manager:** a reference to the widget's manager.
- **filter:** if set to true, the widget will add the content of the field to a fq parameter instead of appending it to the q parameter (default: false).
- **field:** the field in the search engine index this AdvancedSearchField refers to.

Methods

- **init:** an init method, that creates the class.
- **getValue:** a method returning the value of this field, after call to buildSearchTerm method.
- **reset:** an implementation of the parent class method: resets the inputs of this class.
- **buildSearchTerm:** expands the field's search term with the available languages:
  e.g. content:giovanni becomes ((content_en:giovanni) OR (content_it:giovanni))

Information about dynamic query composition:

Since on Solr side we store the title and content on separated fields based on the detected language, the query in input is then translated into the logic OR between the Solr fields that are functionally the same thus having language specialization:
E.g.: content:datafari => (content_fr:datafari) OR (content_en:datafari) OR (content_it:datafari) OR (...).

Then every "expanded" logical term is composed with the other "expanded" search terms by means of the radio buttons.

Please note that the query expansion based on languages is dynamic and it's based on the available languages array in i18n variable.

How to add a new field:

You need create a new field object and add it to the table object, as follows:
in createAdvancedSearchTable function of both search.js

Add new field to advanced search

```
// Table is already created

var ast = new new AjaxFranceLabs.AdvancedSearchTable({ ... });

// Add your field NEW_FIELD

var asf = new AjaxFranceLabs.AdvancedSearchField({
  parent : '#advancedSearchTable',
  label : window.i18n.msgStore['advancedSearch-NEW_FIELD-label'],
  description : window.i18n.msgStore['advancedSearch-NEW_FIELD-descr'],
  field : 'NEW_SOLR_FIELD'
 });

ast.addField(asf);
```

Menu widget

A widget building a dropdown menu.

Variables

- **name:** the parent element text to display.
- **menuItems:** a collection of the elements to add to the menu ({name: ..., link: ...}).

Methods

- **buildWidget:** an implementation of the parent class method.

Slider widget (beta)

A widget that will help you to select values as range or min/max for a chosen field.

This widget uses the jQuery UI Slider plugin.

Variables

- **name:** String - name of the widget that will appear as the title in the search view
- **field:** String - the field name in the Solr core that will be used by the widget
- **fieldType:** String - the type of the chosen field. Only two values are accepted by the widget, 'date' and 'numeric'. (notice that the lowercases matter)
- **elm:** String - the id of the div in the search view that will contain the widget
- **range:** Boolean - set to true if you want a slider with a range, false otherwise
- **min:** Integer - the minimum value of the slider (also used as the default minimum value if the range slider has been selected). Can be negative
- **max:** Interger - the maximum value of the slider (also used as the default maximum value if the range slider has been selected). Can be negative
- **unit:** String - this field is only used when the fieldType has been set to 'date'. In that case you have three options 'YEAR', 'MONTH' or 'DAY'. When a date field type is used with this widget, the resulting Solr query filter that will be applied by the widget will be "(field):[NOW(+/-)(slider_value)(unit)]. So keep in mind that the slider will be based on the current date to create the query. For now we didn't have implemented a parameter to change this behavior
- **defaultValue:** Integer - only used when the slider is not in the 'range mode'. Default value when the widget is loaded for the first time
- **step:** Integer - defines the number which represents the minimum step performed when the user slide
- **comparator:** String - only used when the widget is not in the 'range mode'. Two possible values, 'greater' or 'less' (lowercases matter). When the widget is not in the 'range mode' you have the choice to define if the chosen field value must be greater than or less than the selected one in the slider. For now we didn't have implemented the 'equal' comparator.

Methods

- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

Suggest widget

A widget creating a result widget that proposes other suggested documents to your search query. The name may be misleading, this is not the suggest module for an autocomplete, it is rather for a "more like this" behavior.

Variables

- **boots:** a list of boosts to apply. { field1: boost_value1, field2: boost_value2, ... }.
- **resultWidget:** the result widget which will contain the suggestions, internal use.

Methods

- **init:** an implementation of the parent class method.
- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

OntologySuggestion widget

A widget creating a result widget that proposes the most popular parent and child label of all the results. The name may be misleading, this is not the suggest module for an autocomplete, it is rather for a "related subjects" behavior.

Prerequisites

In order to make this widget work, it is mandatory that the *ontologyEnabled* parameter of the 'datafari.properties' file is set to true, as the documents (at least a few of them) in the results have been enriched by the OntologyUpdateProcessor

Variables

- parentLabelsField: the field in your schema that contains the parent documents labels of the ontology
- childLabelsField: the field in your schema that contains the child documents labels of the ontology
- useLanguage: (boolean) should use the Datafari selected language for both parentLabelsField and childLabelsField ? If *true*, be sure that you used the **OntologyUpdateProcessor** with the parameter **userLanguages** set to *true* during the crawl of the documents by MCF

Methods

- **init:** an implementation of the parent class method.
- **buildWidget:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

Pager module

A module creating a pagination.

Variables

- **source:** the source elements to apply the pagination on. Optional if the elements are retrieved by an Ajax request.
- **prevLabel:** the content of the previous nav button (default: <).
- **nextLabel:** the content of the next nav button (default: >).
- **firstLabel:** the content of the first nav button (default: <<).
- **lastLabel:** the content of the last nav button (default: >>).
- **nbElmToDisplay:** the number of elements to display (default: 10).
- **nbPage:** the number of pages, internal use.
- **nbPageDisplayed:** the number of pages to display in the navigation bar.
- **nbElements:** the number of elements in source, internal use.
- **fastNavigationAlwaysVisible:** if set to true, will always display the navigation button even if they are useless (eg: first and previous button when you are on the first page. Default: false).
- **display:** the css display value of the pagination elements.
- **pageSelected:** the page actually selected (default: 0).

Methods

- **init:** an implementation of the parent class method.
- **updatePages:** the method updating the navigation bar (eg: elements to display)
- **clickHandler:** a callback function called when a click on an element of the navigation is detected.
- **updateList:** a method always called when a click on an element of the navigation is detected. This method is empty by default and must be implemented if you need it.

Autocomplete module

This module creates a list of suggestions when you type in an input field. Note: this requires enabling a suggest component as a requesthandler of Solr (done in the xml config files), in case you use a pure Solr. If you are using Constellio, this config is normally already activated in its Solr. For Constellio 1.3, it also requires a modification of the web service servlet in order to reach the Constellio autocomplete service. Contact us if you need help.

This module is also an extension of the jQuery UI Autocomplete plugin and also has all its parameters and methods.

Variables

- **options:** options given to the jQuery UI Autocomplete plugin.
- **optionsExtend:** if set to true, extends the default parameters defined in this module (default: true).
- **render:** an empty variable. You can define it a function changing the way an element is rendering in the autocomplete.
- **field:** if the values retrived must be focused on a specific field of your search engine index. Note that this option will not work with a standard 1.3 Constellio. It requires a modification of its web service. This field variable becomes the variable "f" in the generated query to the server
- **openOnFieldFocus:** if set to true, executes a request to retrieve data when cursor entering the input field (default: false).
- **valueSelectFormat:** a function that format the data selected in the autocomplete results (eg: if you want to put it between double quotes).
- **singleValue:** if set to true, when a value is choson in the autocomplete result, the actual value in the input field will be replaced by the selected one.
- **searchForItSelf:** if set to true, also search the term itself (default: false). This means that the list will contain the word currently typed in addition to the words which have at least one more letter

Methods

- **init:** an implementation of the parent class method.

Collection widget

A widget displaying your constellio's collections. Obviously, this widget works only for Constellio. Note that this requires that your manager was initialised with the array of Constellio collections. See AbstractManager.collections for that. Note also that it requires a modification of the Constellio 1.3 web service servlet. Contact us to know what to modify.

Methods

- **buildWidget:** an implementation of the parent class method.

Capsule widget

A widget that gives you tips about your search. Does not work for Solr. Note that this only works for Constellio 1.3, with a modified web service in order to support the request.

Methods

- **init:** an implementation of the parent class method.
- **buildWidget:** an implementation of the parent class method.
- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.

Favorite documents and searches

A module adding the possibility to create a favorite system on your searches and documents. Does not work for Solr. Note that this only works for Constellio 1.3, with a modified web service in order to support the request.

Variables

- **userId:** the id of the user manipulating its favorites. This is actually optional if your server is managing the user sessions.

Methods

- **beforeRequest:** an implementation of the parent class method.
- **afterRequest:** an implementation of the parent class method.
- **removeFavoriteDocument:** takes as parameter the id of the favorite document and a callback method.
- **removeFavoriteSearch:** takes as parameter the id of the favorite search and a callback method.
- **removeSearchInHistory:** takes as parameter the id of the search to remove in the history and a callback method.
- **getFavoriteDocuments:** takes as parameter a callback function, the first element to start on and the number of elements to retrieve.
- **getFavoriteSearches:** takes as parameter a callback function, the first element to start on and the number of elements to retrieve.
- **getSearchHistory:** takes as parameter a callback function, the first element to start on and the number of elements to retrieve.

LoginDatafariFormWidget

The widget containing the login to admin UI form.

The code of this widget has been migrated from embedded JS of login.jsp to a separate widget.

Variables

No specific variables for this widget, only the ones inherited from AjaxFranceLabs.AbstractWidget.

Methods

- **buildWidget:** an implementation of the parent class method. Creates the widget.
- **JQuery anonymous function $(function($) {...}:** adds the widget to the manager.

LoginDatafariLinksWidget

The widget containing the login to admin UI form.

The code of this widget has been migrated from embedded JS of index.jsp to a separate widget.

Variables

No specific variables for this widget, only the ones inherited from AjaxFranceLabs.AbstractWidget.

Methods

- **buildWidget:** an implementation of the parent class method. Creates the widget.

LanguageSelectorWidget

This widget enables the user to change the language of the UI, among the available ones.

When a language change is detected by the application, it gets reset and a new splash page is displayed.

To add a language, you need to add an option in the combo box (option DOM element of buildWidget).

Variables

No specific variables for this widget, only the ones inherited from AjaxFranceLabs.AbstractWidget.

Methods

- **buildWidget:** an implementation of the parent class method. Creates the widget.

Add custom fields in Datafari and display them

Let's say we want to add a new metadata field in Solr named meta:author in the source and named **authorname** in Datafari.
Let's see each step to display the field into the Datafari UI and permit to search into the field with Solr.

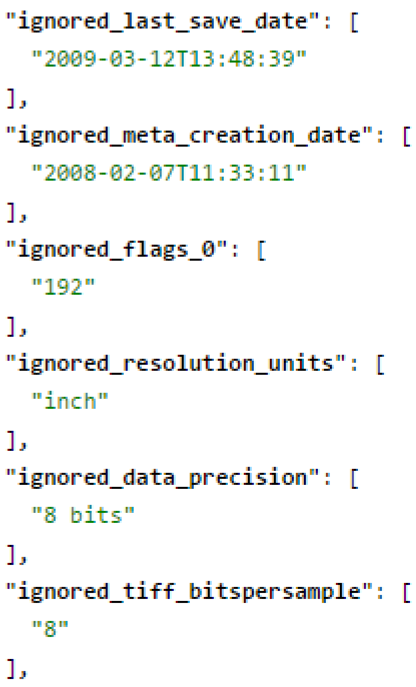
1) Edit solrconfig.xml

We want to map the original metadata of the source file : **meta:author** to a new Solr field named authorname. So we have to edit the Solr cell request handler :

```
<requestHandler name="/update/extract"
              startup="lazy"
              class="solr.extraction.ExtractingRequestHandler" >
<lst name="defaults">
        <str name="scan">false</str>
        <str name="captureAttr">true</str>
        <str name="lowernames">true</str>
        <str name="fmap.language">ignored_</str>
        <str name="fmap.meta_author">authorname</str>
        <str name="fmap.source">ignored_</str>
        <str name="uprefix">ignored_</str>
        <str name="update.chain">datafari</str>
        <bool name="ignoreTikaException">true</bool>
```

The correct syntax is meta_author (and not meta:author) because of the line
`<str name="lowernames">true</str>`
The documentation says : "lowernames=true|false - Map all field names to lowercase with underscores"

You can also see in the configuration that we store all the ignored metadata in the dynamic field ignored. I invite you to change the configuration of the field in the schema.xml and to change stored=false to store=true to see all the metadata found by Tika (and to see the correct syntax to map the fields into Solr) For example :



```
"ignored_last_save_date": [
  "2009-03-12T13:48:39"
],
"ignored_meta_creation_date": [
  "2008-02-07T11:33:11"
],
"ignored_flags_0": [
  "192"
],
"ignored_resolution_units": [
  "inch"
],
"ignored_data_precision": [
  "8 bits"
],
"ignored_tiff_bitspersample": [
  "8"
],
```

2) Edit schema.xml

We want now to add the new field into the Solr schema. So add the following line :

```
<field name="authorname" type="text_en" indexed="true" stored="true" multiValued="true"/>
```

Ok so far we can launch the indexation with ManifoldCF and the new field is well present in Solr.

3) Add the new field to the search Edit solrconfig.xml, in the select request handler add the field :

```
<str name="qf">author title_en^50 title_fr^50 content_fr^10 content_en^10 source^20 extension^20
        </str>
```

After the core reloads, we can now search and find the data of the new field.

4) Configure the Datafari UI into datafari/tomcat/webapps/Datafari/js/main.js (source code) or Datafari/tomcat/webapps/Datafari/js/main.js (installed version) Change the line :

```
Manager.store.addByValue("fl", 'title,url,id,extension');
```

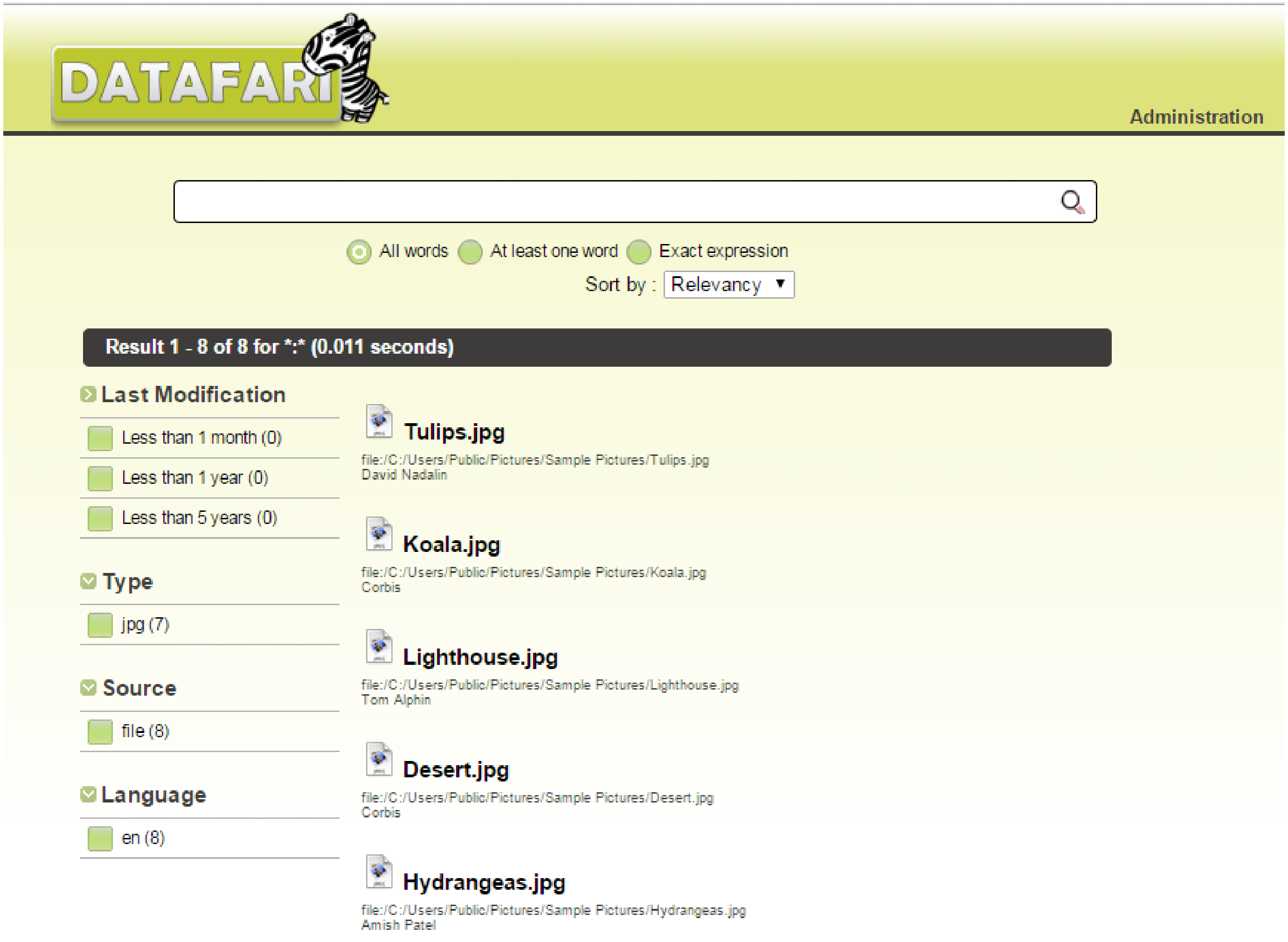
And add the field you want to add, here autorname

```
Manager.store.addByValue("fl", 'title,url,id,extension, authorname');
```

The last step is to change the Javascript file search.js : datafari/WebContent/js/search.js (source code) or Datafari/tomcat/webapps/Datafari\js/search.js (installed version) Add the display of your field adding the code : doc.subject where you want to add it. For example if you want to add it after the URL of the document : (I made a mistake in my previous answer, it is correct now)

```
elm.find('.doc:last .address').append('<span>' + AjaxFranceLabs.tinyUrl(decodeURIComponent(url)) +
'</span>')
elm.find('.doc:last .address').append('<div id="author">' + doc.author );
```

And finally your Datafari UI should be like that with the field author at the end of each document :



DATAFARI

Administration

○ All words ○ At least one word ○ Exact expression

Sort by : Relevancy ▼

Result 1 - 8 of 8 for *:* (0.011 seconds)

Last Modification

- 🟩 Less than 1 month (0)
- 🟩 Less than 1 year (0)
- 🟩 Less than 5 years (0)

Type

- 🟩 jpg (7)

Source

- 🟩 file (8)

Language

- 🟩 en (8)

Tulips.jpg
file:/C:/Users/Public/Pictures/Sample Pictures/Tulips.jpg
David Nadalin

Koala.jpg
file:/C:/Users/Public/Pictures/Sample Pictures/Koala.jpg
Corbis

Lighthouse.jpg
file:/C:/Users/Public/Pictures/Sample Pictures/Lighthouse.jpg
Tom Alphin

Desert.jpg
file:/C:/Users/Public/Pictures/Sample Pictures/Desert.jpg
Corbis

Hydrangeas.jpg
file:/C:/Users/Public/Pictures/Sample Pictures/Hydrangeas.jpg
Amish Patel

Alerts - Architecture and mechanism

The alerts are stored in a local Apache Cassandra. You can create, edit or delete them from the *alerts.html* page which communicates with the *Alerts* servlet. This servlet creates a connection towards the database with the info contained in the *alerts.properties* file.

> ⚠ For the moment, the alerts cannot be used on plain text files, as Tika cannot extract any metadata like the modification date from them.

An alert is composed of seven fields:

- an *_id* created by the Cassandra,
- a *keyword* which will be used to make the request on the solr core,
- the *core* on which the query will be made, default : FileShare,
- a *frequency*: Hourly, Daily or Weekly. Frequency changes the time lapse between two requests but also the recency of the documents returned.
- a *mail* to which the results (if any) of the query will be sent,
- a *subject* which describes the subject of the email
- a *user*: the one who created the alert in the UI. It is only used for now to allow for the monitoring of the alerts by the admin and the search expert.

The *alertsAdmin* UI and servlet are used to modify the *alerts.properties* file. It is used to:

- configure the initial delay of the execution of the alerts of a specific frequency,
- print the date of the previous execution
- print the date of the next planned executions.

- configure the mail address from which Datafari will send its emails.
- configure the host and port of the Cassandra and the name of the Database and of the collection where the alerts are.

The *AlertsManager* class is a Singleton and is called at each start/stop of the server by the *StartAlertsListener*.

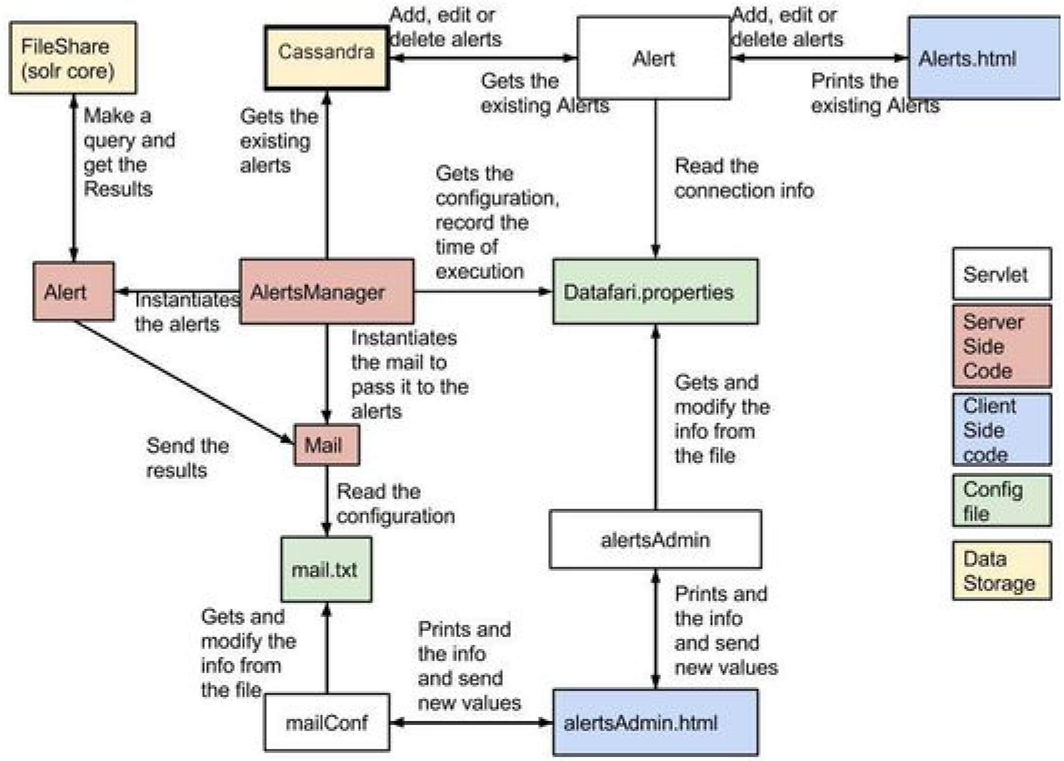
At each start it reads the *alerts.properties* file to fill its own fields, and check if the alerts were on or off at the end of the previous execution of Datafari. If it was on On, the alerts Manager schedules the task according to the dates in the file. However if those are invalid, it will calculate the delays according to the original execution schedule of the alerts.

For instance the hourly alerts are set to begin at midnight, if everything is going smoothly, the hourly alerts will run at 00:00 then at 01:00 and so on. Now let's say that the date was already past at the start of Datafari. Then the AlertsManager will plan the execution for the next expected hour.

This allows the alerts to be run regularly, even after one or more restart of the server.

In order to run an alert:

- the *AlertsManager* creates a connection with the Cassandra database at each execution
- the *AlertsManager* then retrieves all the alerts with the matching frequency.
- For each alert, the *AlertsManager* creates an Alert object, and delegates the execution to them.
- An object alert will use its attributes to make a solrQuery
- If the solrQuery contains results, it will use the email object passed at its creation to send a mail. To avoid re reading of the *alerts.properties* file for each alerts, this object is initialized in the alerts manager and passed as a parameter to the Alert constructor. If the configuration file is modified through the admin UI, Datafari will automatically restart the alerts, if it is manually modified, you will need to turnOff/turnOn the alerts in the alertAdmin UI to take into account the modifications.


FileShare (solr core)
Cassandra
Add, edit or delete alerts
Alert
Add, edit or delete alerts
Alerts.html
Gets the existing Alerts
Prints the existing Alerts
Make a query and get the Results
Gets the existing alerts
Gets the configuration, record the time of execution
Read the connection info
Alert
Instantiates the alerts
AlertsManager
Datafari.properties
Servlet
Server Side Code
Client Side code
Config file
Data Storage
Instantiates the mail to pass it to the alerts
Gets and modify the info from the file
Send the results
Mail
Read the configuration
alertsAdmin
mail.txt
Prints and the info and send new values
Gets and modify the info from the file
Prints and the info and send new values
mailConf
alertsAdmin.html

The schema is wrong concerning the Datafari.properties file and mail.txt file, both files are no more used by the AlertsManager and they have been replaced by one configuration file named alerts.properties located in Datafari_Home/tomcat/conf

More info on our code?
Please refer to our code in github, as our source code contains additional technical details avaiable as comments

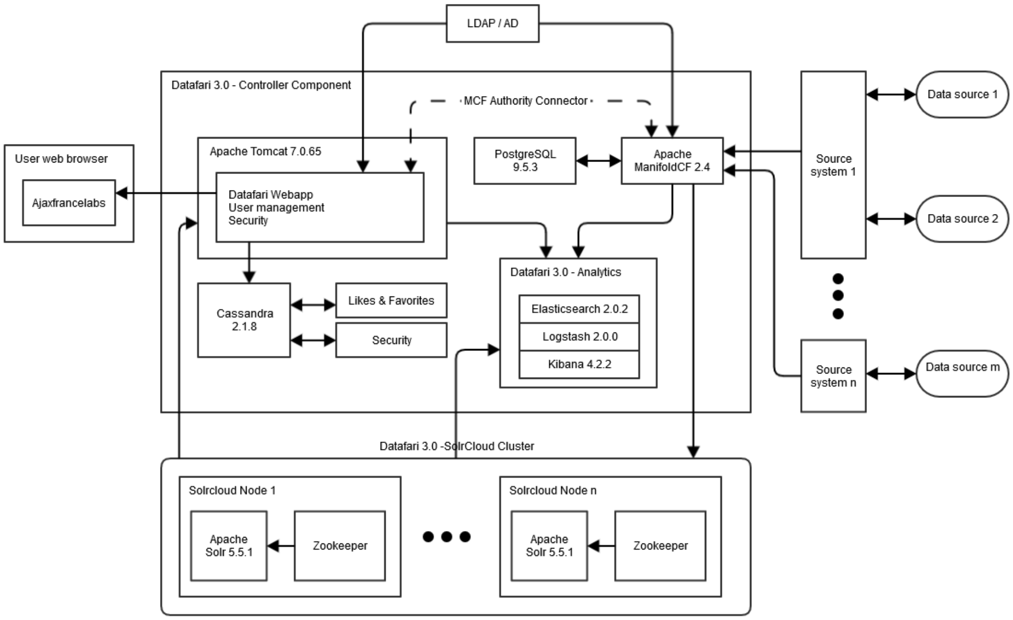
Alerts - technical doc (from 3.1)

The version 3.1.x of Datafari introduces a new way to create, consult and modify alerts.

The creation of an alert is now implemented by a widget embedded in the searchView. The widget javascript file is WebContent/js/AjaxFranceLabs/widgets/CreateAlert.widget.js and it is instantiated in WebContent/js/search.js and imported as a script file in WebContent/searchView.jsp.
A 'div' element with the id 'create_alert' has been added to the searchView.jsp file to be the container of the widget but is only added when an authenticated user is detected.
The widget adds a 'Create alert' button to the UI which displays, when the user click on it, a text field to enter an e-mail address and a 'select' element to choose the frequency of the alert. Once the user validate theses parameters, the widget sends them as a POST request, in addition with the search query parameters, to the "/admin/Alerts" servlet (code located in src/main/java/com/francelabs/datafari/servlets/admin/Alerts.java). Then the servlet insert everything in the "alerts" table of the Datafari database in Cassandra.

The management of the alerts is now implemented in the 'parameters' page and the javascript code is located in WebContent/js/parameters.js in the createAlertContent() method.
The goal of this javascript code is to provide a table displaying all the alerts created by the authenticated user and provide a 'modify' button for each of them to change the frequency (for now you cannot change the e-mail address), and a 'Delete' button.
Modify an alert will result in a POST request to the "admin/Alerts" servlet with the new frequency and all the other alert parameters as POST data. Therefore, the behavior is the same than the CreateAlert widget.
Delete an alert will result in a POST request to the "admin/Alerts" servlet with only the alert id as POST data. The servlet detects that only an id has been provide so it has to delete the corresponding entry in the database.

Architecture



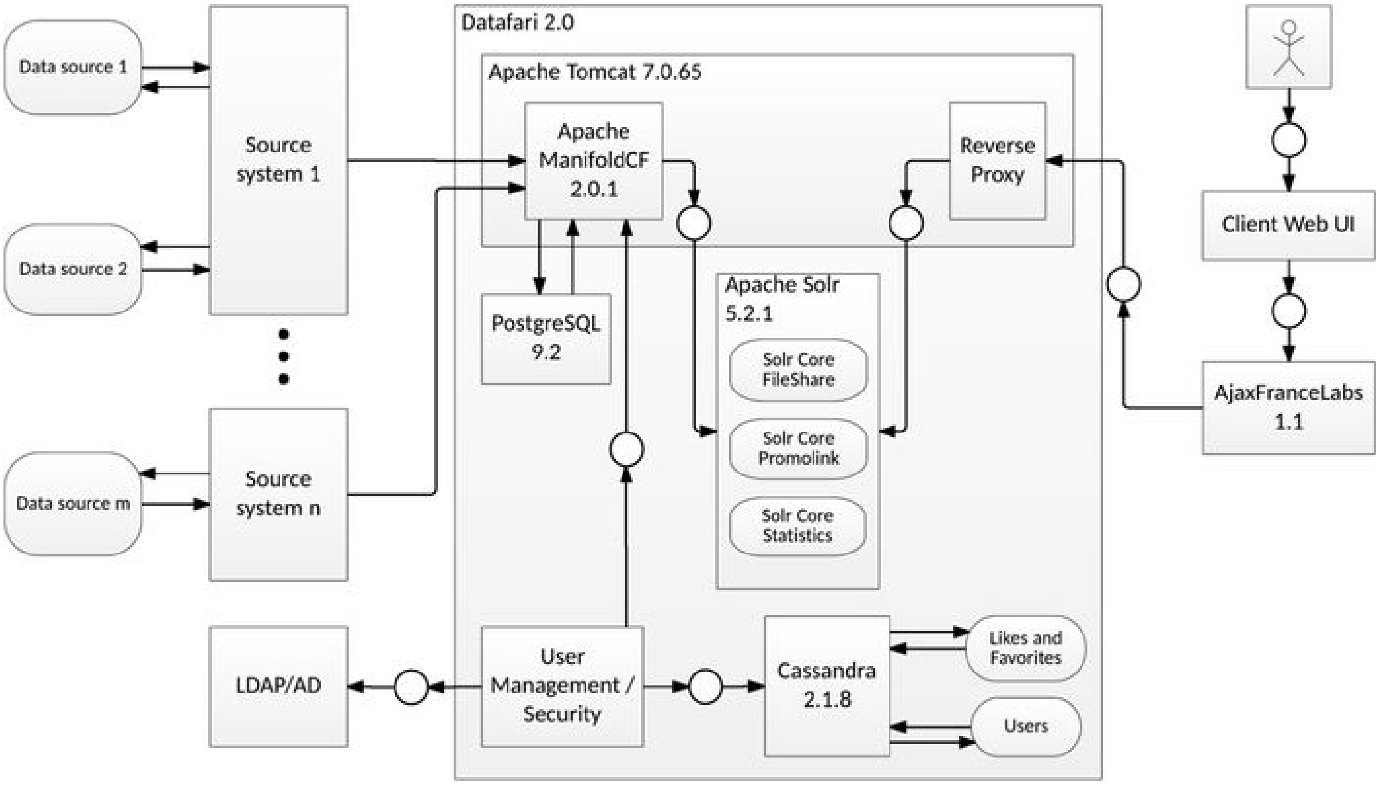
Valid for V2.0

Datafari uses a typical search engine architecture. It is based on the triptych crawling, indexing and search. The crawling part is using Apache ManifoldCF. There is another opensource connectors framework under Apache licence v2, the Google Connector Framework. Yet the latter is only being supported and developed by Google, so it appeared more reasonable to us to leverage ManifoldCF, proposed by the Apache foundation, and which benefits from the support of several committers from different entities.

The indexing and search parts both use Apache Solr. Again, there is another popular indexing and search engine available under Apache licence V2. It is Elasticsearch, but similarly to the Google Connector Framework, it is led by the eponym entity, and as such does not guarantee such a longevity as Apache Solr.

ManifoldCF, although being independent from Apache Solr, has the advantage of being conceived from the start as Solr’s connectors framework. It is thus conceptually “naturally” connected to Apache Solr.

The figure below illustrates the system architecture that we are using for Datafari. Its v2 is becoming rather large in terms of components, so this architecture is rather high level and intentionally avoids some connections and components for the sake of clarity.



.

Datafari 3.0

On Datafari 3.0 we chose to use directly SolrCloud and to isolate the different parts of our product that can be installed on different servers.

If your Solr index becomes very huge or if your nomber of users becomes too big, you can easily shard your index and/or replicate it with SolrCloud.

Custom Solr configuration

Datafari allows you to add your own Solr configuration to the existing one, without modifying the original configuration files. To do so, both solrconfig.xml and schema.xml files are including custom files that are organized by sections:

> Note that if for some reason you need to modify directly the solrconfig.xml file or the schema.xml file, you can do it but we strongly advise you not to do so. These files may change in future versions of Datafari and you will have to manually manage the conflict during the update process.

solrconfig.xml includes 4 custom files located in *{Datafari_Home}/solr/solr_home/FileShare/conf/customs_solrconfig*:

- **custom_libs.xml**: Used to add new jar libraries that Solr should load. For exemple, to use the ontology feature of Datafari, the Apache Jena lib directory will be added in this file
- **custom_request_handlers.xml**: Used to add new request handlers
- **custom_search_components.xml**: Used to add new searchComponents
- **custom_update_processors.xml**: Used to add new updateProcessors. The updateProcessors that you will specify in this file will be added to the 'datafari' updateRequestProcessorChain (refer to the solrconfig.xml)

schema.xml also includes 4 custom files located in *{Datafari_Home}/solr/solr_home/FileShare/conf/customs_schema*:

- **custom_copyFields.xml**: Used to add new copyFields to the FileShare core. An example on how to define custom copyField can be found in **custom_copyFields.xml.example**
- **custom_dynamicFields.xml**: Used to add new dynamicFields to the FileShareCore. An example on how to define custom dynamicField can be found in **custom_dynamicFields.xml.example**
- **custom_fields.xml**: Used to add new fields to the FileShare core. An example on how to define custom field can be found in **custom_fields.xml.example**
- **custom_fieldTypes.xml**: Used to add new fieldTypes to the FileShare core. An example on how to define custom field can be found in **custom_fieldTypes.xml.example**

These files are not loaded automatically in the schema. To do that, you should use the script *{Datafari_Home}/solr/solr_home/FileShare/conf/customs_schema/addCustomSchemaInfo.sh* to load the custom schemas information in Datafari through the schema API of Solr.

> Those 8 custom files contain by default an xml tag <to_remove/> This tag serves to pass the xml validity check of the file when Solr loads it, and avoid exceptions that will crash Solr. When you fill those files with your needs, you should remove or comment this tag and put it again when you remove everything from a file.

Deduplication : Technical note

The deduplication functionality is one of the functionalities that is simplified since Solr 1.4 and the above versions, as it proposes to enable it through its config files. Datafari used this functionality to implement it and make it available in the front-side.

- For the backend side, we have applied the wiki that is available here : https://cwiki.apache.org/confluence/display/solr/De-Duplication . The hash we generate is stored in a solr field labelled *signature*. By default, the hash is computed on the solr *content* field.
- we set the overwriteDupes parameter to false :

  ```
  <bool name="overwriteDupes">false</bool>
  ```

  in the *solrConfig.xml*. You can set it to true if you want the index to contain only one instance per set of duplicated documents. This is not our goal here (we want to let the search admin the possibility to remove duplicates after the fact).
- we have set the fields parameter of the *solrConfig.xml* to:

  ```
  <str name="fields">content,content_en,content_fr</str>
  ```

  so that the hash will take in consideration only the content of each document. You can change this parameter to put the fields you want, for instance the title so that the hash will take into consideration the title.
- The processors of the dedupe are in :

  ```
  <updateRequestProcessorChain name="datafari">
  ```

  We used also - for max precision - the MD5 Algorithm of solr for hashing. Still, you can always change it by changing the parameter*:*

  ```
  <str name="signatureClass"> solr.processor.MD5Signature</str>
  ```

  in *solrConfig.xml*.

For the front-end, we want to expose duplicates using a facet:

For this, we have created a new class called FacetDuplicates and which is located in */datafari/WebContent/js/AjaxFranceLabs/widgets/*. This class inherits from *TableWidget* and overloads the update method. This was achieved due to the fact that duplication is a facet and so returns only the hashes and that we wanted to return the names of a document from the duplicated documents. So what happens is that we send for every hash in the facet a get query. We have also set a mincount for the facet so we will show only duplicated file names in the facet which is not the case when the mincount is equal to 0 (which is the default configuration). You can find in this link a short doc for the parameter mincount : https://cwiki.apache.org/confluence/display/solr/Faceting. We also make the facet disappear if it doesn't contain a duplicated document : a simple $('#facet_signature').show/hide had done the trick.

ELK

> **SECURITY WARNING**
> In the community version, ELK is not secured. This means that anyone that knows the url to access the dashboards, will see everything. It is up to you to secure your ELK environment: some of your options can be to switch to the Datafari Enterprise edition, or to do it yourself, or to disable ELK.

> Feature only available from the v2.2 of Datafari

> **ELK configuration notes**
> By default, aside from Kibana, Elasticsearch and Logstash are automatically configured by Datafari on the first start, to fit with your installation directory and be ready to run.
> If for some reasons (like an architecture decision), you move one or more of these components to a different place than the installation directory of Datafari, you must modify the ELK environment parameters located in [DATAFARI_HOME]/elk/scripts/set-elk-env.sh and maybe also the [DATAFARI_HOME]/elk/scripts/start-elk.sh and [DATAFARI_HOME]/elk/scripts/stop-elk.sh scripts.
>
> Furthermore, if you want to manage by yourself the instance of ELK you just need to set the ELKactivation property to "true" in [DATAFARI_HOME]/tomcat/conf/datafari.properties and enter the correct Kibana URI in the ELK configuration page of the admin UI

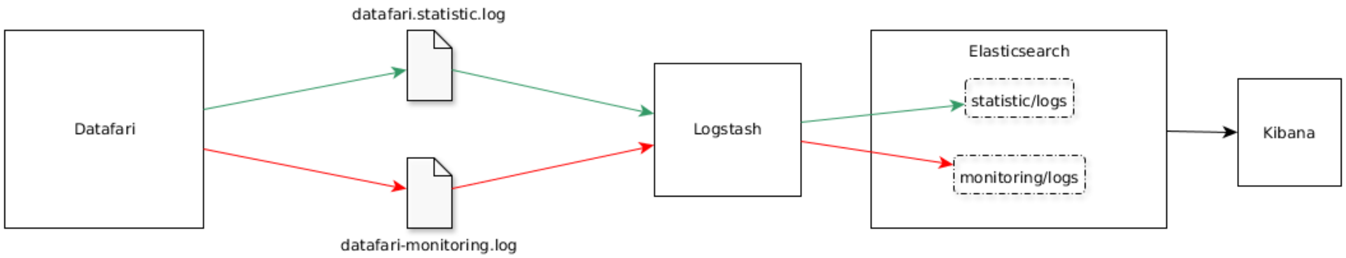
The 2.2 version of Datafari comes with an Elasticsearch Logstash Kibana layout in order to bring customizable monitoring and analytic views on Datafari.

> In order to use the ELK layout, you need to configure it first !

Two new kind of logs are generated each in separate files, in order to be exploited in Kibana:

- Statistics logs
- Core Monitoring logs

The diagram below shows the flow between Datafari and Kibana:



As Datafari writes lines in the log files, Logstash, in near real time, catches each new line and insert it in the "statistic/logs" or the "monitoring/logs" index of Elasticsearch, according to the log line comes from the "datafari.statistic.log" file, or the "datafari-monitoring.log".

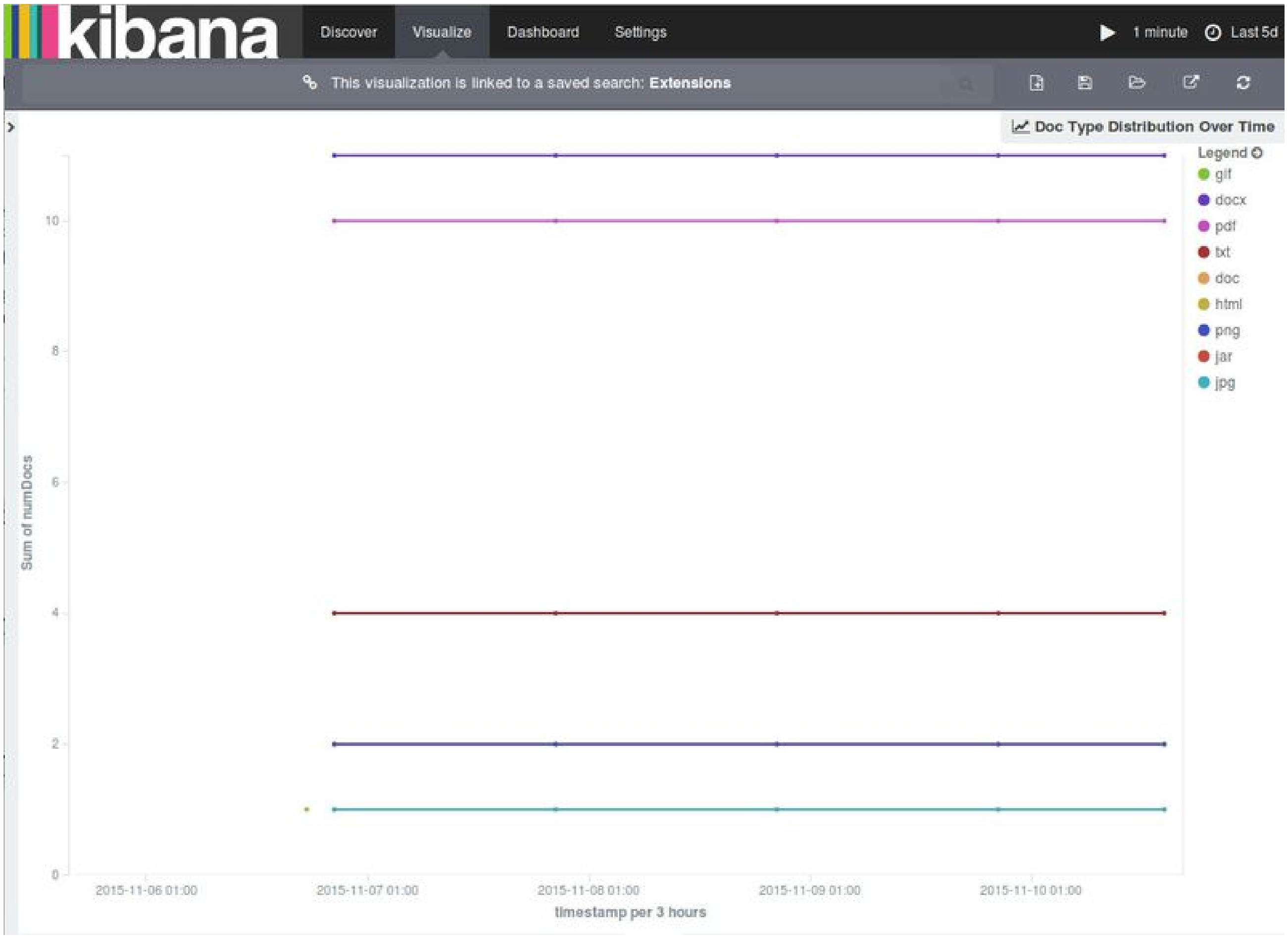
The "logstash.conf" file which describe the path of the input log files and how to process the lines and insert them into Elasticsearch can be found in the Logstash main repository. Refer to the Logstash documentation for a deep understanding on how it works.
You will also find the templates used for each Elastisearch index in the "templates" directory under the Logstash main directory. The files are called "datafari-statistic-template.json" and "datafari-monitoring-template.json".

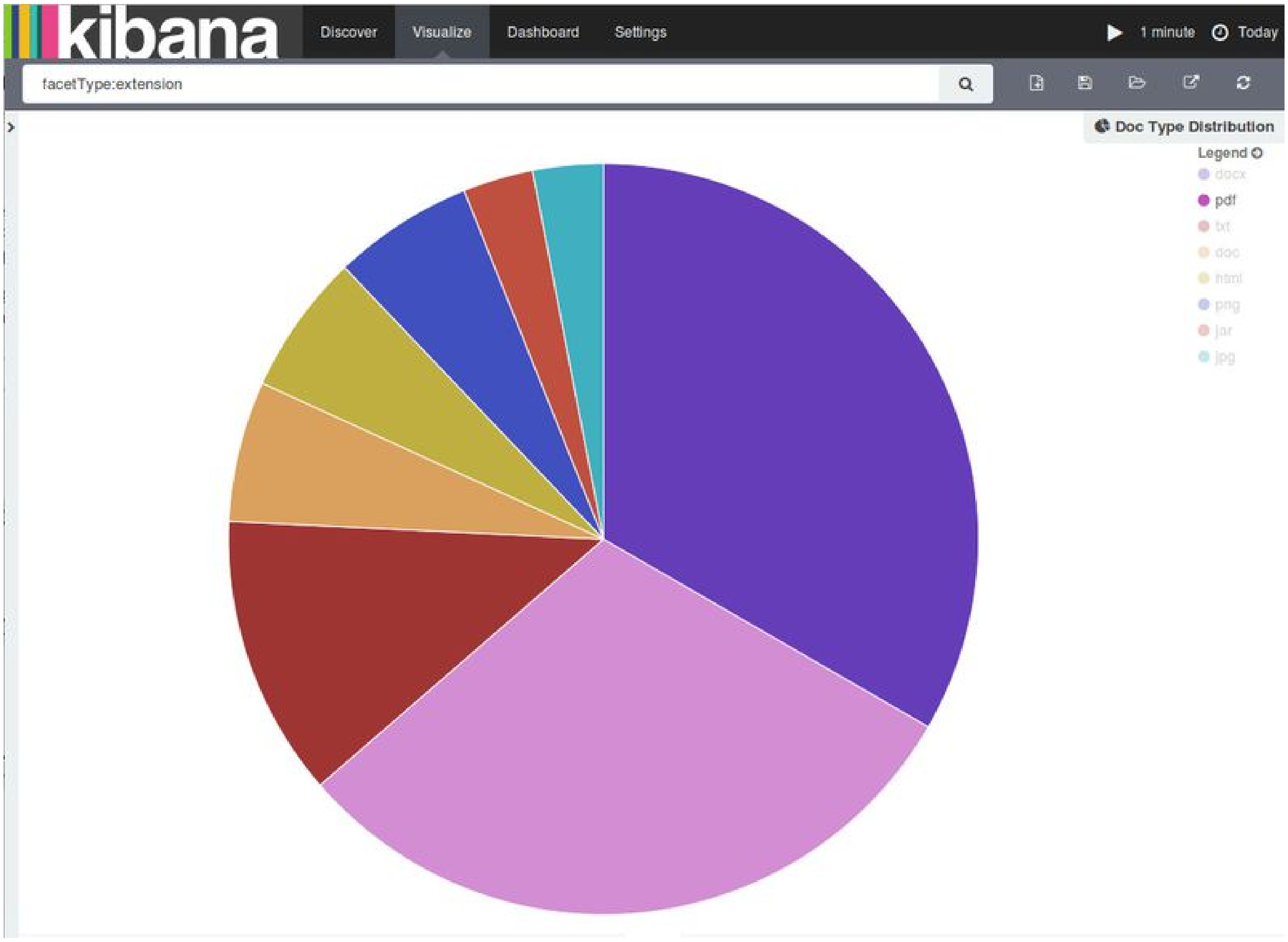
Let's describe how the flows works for each kind of log:

1. **Statistics logs**
   Thoses logs give informations about the queries and actions performed in Datafari by users. For each query or action, a log line is written in the *datafari.statistic.log* file, Logstash detects the new line, applies some filters (defined in the *logstash.conf* file), and inserts it as a document in the statistic/logs index of Elasticsearch.
   As described in the Statistics logs page, an id is generated for each query (the actions related to a query, like clicking on a result, keep the id of the query) by Datafari. This id is used as the document id in Elasticsearch. By doing this, one has the guarantee, in Elasticsearch and then Kibana, to not have duplicates data for the same query/action.
   So, if several lines in the log file are related to the same query/action, they will be inserted in the same document in Elasticsearch, which consequently means that the document in Elasticsearch will correspond to the most recent log line. However, this does not mean that informations are lost, because a log line contains the history of the query and his related actions.
   Thanks to this flow, you can see what kind of charts you can create on the Usage Statistics page.

2. **Core Monitoring logs**
   Those logs give informations on the content of the main Solr core of Datafari at a fixed rate. The default rate is once per hour and currently can only be changed by modifying it in the code.
   So, at each iteration, Datafari performs some queries on Solr, format the results and writes several logs lines in the *datafari-monitoring.log file*. In near real time, Logstash detects the new lines, applies some filters (defined in the *logstash.conf* file), and inserts them as documents in the *monitoring/logs* index of Elasticsearch.
   As described in the Core Monitoring logs page, a log line contains an id which is based on the facet value, the facet field and the time event unity. This id is used as the document id in Elasticsearch.
   The time event, which is "daily" by default, correspond to the unity that will be used to visualize data in Kibana. For example, with the default "daily" unity, you will have one Elasticsearch document by facet value and field, by day. If you set the time event unity to "hourly" you will have one Elasticsearch document by facet value and field by hour.
   This means that as long as a new time event has not started, for each new log line related to a couple of facet value and field, only one document will be created/updated. When a new time event starts, a new document is created/updated for the same couple of facet value and field.

   Concretely, a daily time event unit allows such Kibana visualizations:





As you can see on the time based line chart on the left, which represents the document type distribution over the time, we only have one dot per doc type per day, so, for the last 5 days, 5 values for each facet value of the face field "extension".
On the global doc type distribution pie chart on the right, one have the global distribution of document types in the Solr core. This chart is not based on time unlike the other one, so, to be able to have a correct distribution view, one need to set the time frame of Kibana to the time event unity evoked before : the current day
Notice that, as our monitoring log iterations are hourly (by default), our data are updated each hour without having more than one dot per day in the time base line chart.

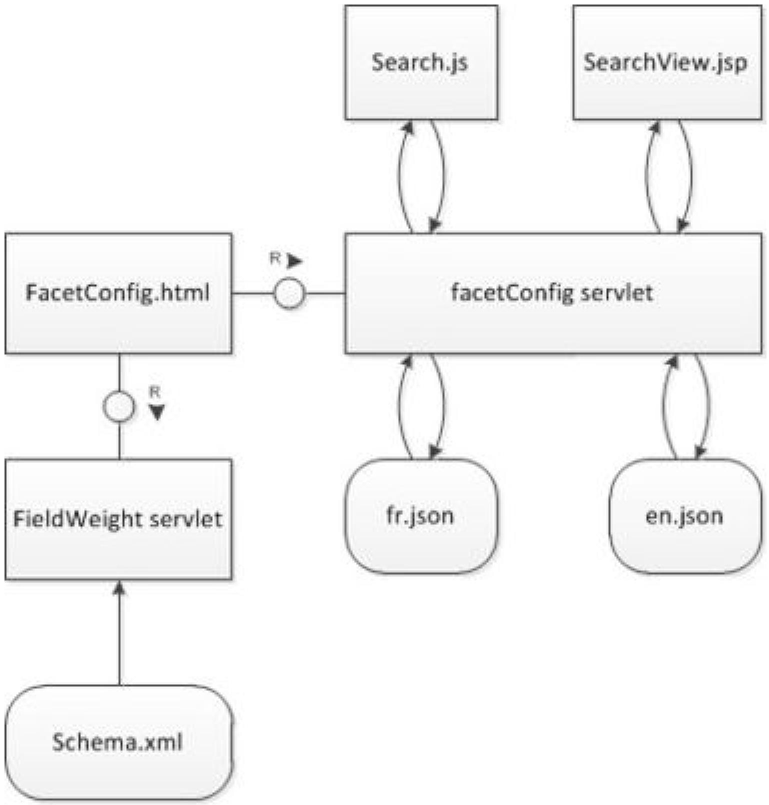
Now you have a better understanding on the time event unity that you can change in the code and his effect on Elasticsearch and the visualization of data in Kibana. Up to you to set you own time event unity and adapt your visualizations in consequence.
You can find more example of time based charts and "normal" charts you can create thanks to this flow in the Content Analysis Over Time and Content Analysis pages.

Faceting

FacetConfig allows you to create custom facets, to delete them, or to modify their printing order.

The UI interacts with the fieldWeight and the facetconfig servlet:

- fieldWeight reads the schema.xml,
- facetconfig reads and writes the internationalisation file, the searchview.jsp and the search.js



At creation:

- a request is made towards FieldWeight servlet to get the list of fields,
- another request is made towards the facetConfig servlet to get all the existing servlets, removing them from the selection.
- the submit button sends a post request to facet config, with all the facets. Those are differenciated between themselves by their parameters. Both types of facets are added pretty much the same way, except in the modification of the search.js.

Note that adding a facet modifies four files : the searchview.jsp, the search.js, then en and fr.json. You need to modify or add a json file after you have changed or added another language.

For query facets, the construction of the String added to the js is different, it includes every query that is filled, and if there is no query passed as parameters then it ignores the request.

Manually adding facets to the Datafari UI

In this tutorial, we will see how to add a facet in the Datafari UI. The facet here will be a facet field, that means that the source of the faceting are the different values of the Solr field.

- Indexation
  - Add a metadata value in a MCF job
  - Field already added to the Datafari schema
- Configure the Datafari User interface

Indexation

We distinguih two cases here :

- you want to add some metadata from you ManifoldCF job
- or you already modified your schema and you indexed a custom Solr field

Remark : in the tutorial we write the absolute path for Debian version wich starts by /opt/datafari but it is the same for Windows paths : just replace the beginning of the path by your local installation of Datafari

Add a metadata value in a MCF job

In our example, let's say that we want to have a facet with the source of the ManifoldCF job. Here I want to index 2 websites with ManifoldCF : francelabs.com and datafari.com. I want to propose to the users to have a facet with the origin of the document : francelabs.com or datafari.com.

Let's do this !

- Add the custom field to the Solr schema

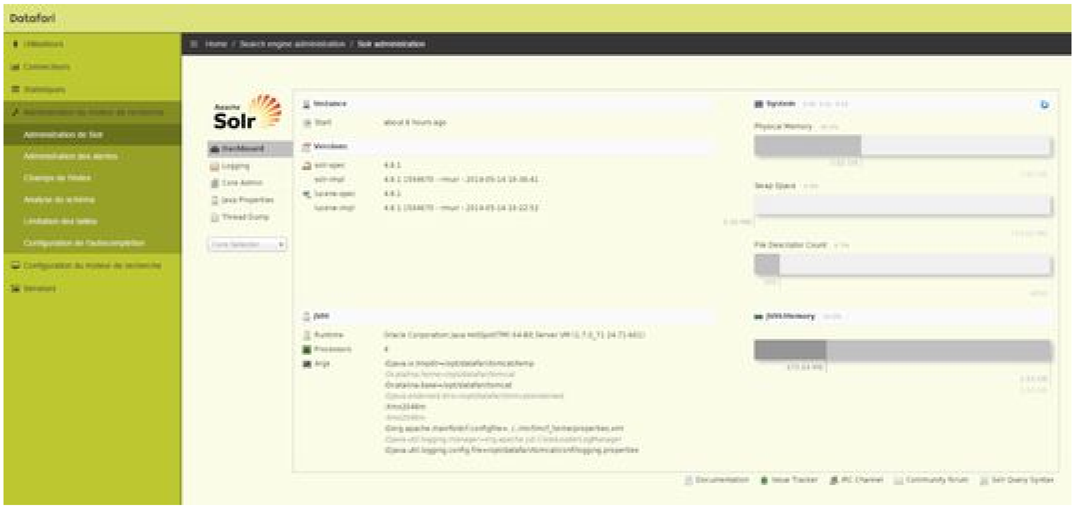
Edit custom_fields.xml located in /opt/datafari/solr/solr_home/FileShare/conf/customs_schema/custom_fields.xml and add your new field. It needs to be at least indexed (not necessarily stored) and not tokenized so if I want to add the field named job, the configuration will be :

```
 <field name="job" type="string" indexed="true" stored="true"
multiValued="false"/>
```

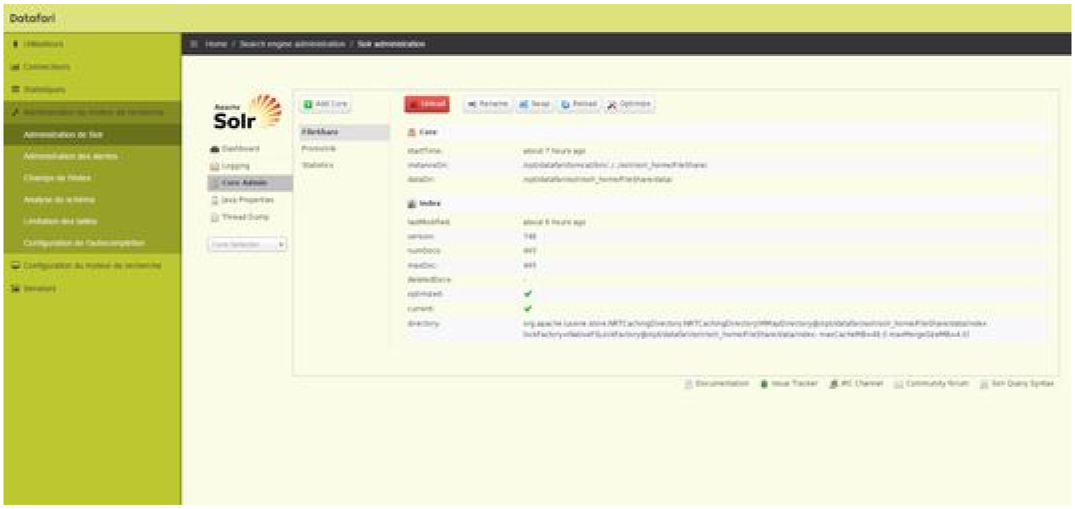
Don't forget to save your changes into the file.

- Reload the FileShare core for Solr to load your new schema

Go to the Datafari Admin User Interface (UI) then click on Search Engine Administration then on Solr Administration



Then click on Core Admin on the main window, check if FileShare is highlighted then click on the Reload button

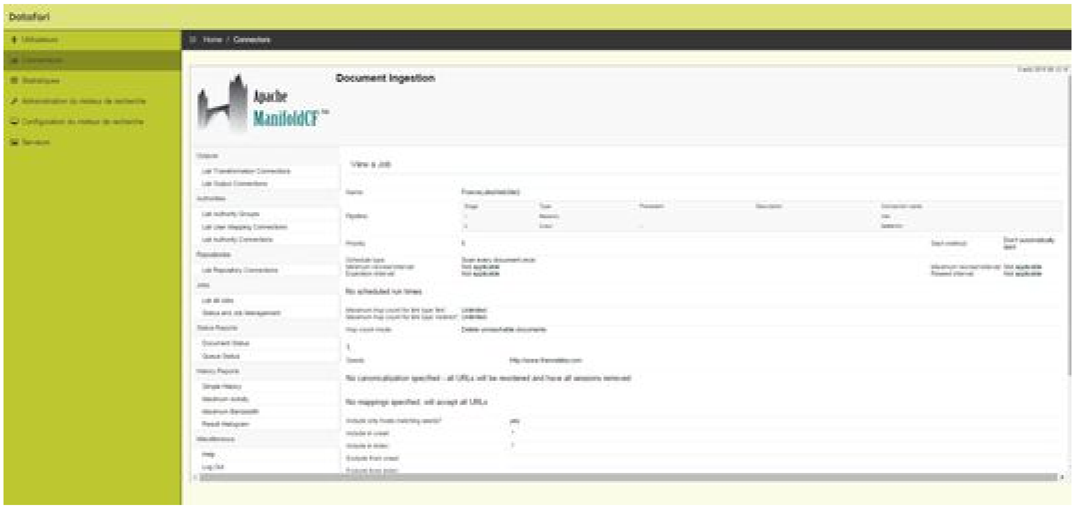


The button is now green, it means that the new configuration is correctly loaded into Solr.

- So now we can configure ManifoldCF. First thing is obviously to configure the repository connection and the job configuration.

I invite you to go to this section of the wiki if you are not familiar with this step : Crawling

Here we add a Web repository connection and then added a FranceLabswebsite job in which we configured the seeds tab with the url of the France Labs website.



Document Ingestion

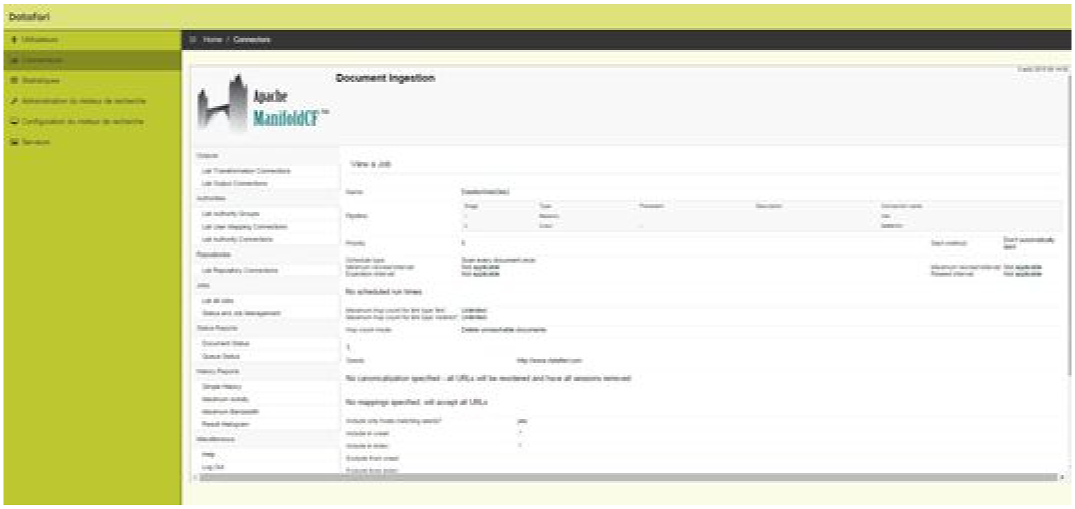
View a Job

No scheduled run times

No canonicalization specified - all URLs will be reordered and have all sessions removed

No mappings specified, will accept all URLs

We did the same thing for Datafari website (seed: datafari.com) :



Document Ingestion

View a Job

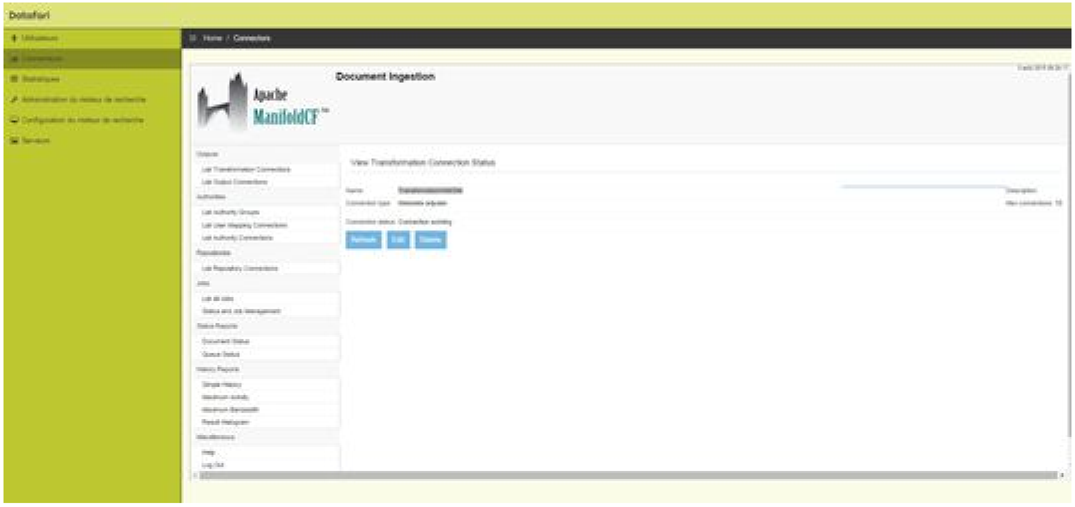
No scheduled run times

No canonicalization specified - all URLs will be reordered and have all sessions removed

No mappings specified, will accept all URLs

- So for now we have our 2 ManifoldCF (MCF) jobs well configured. We need to indicate to ManifoldCF to add a particular metadata for each job to distinguish the documents. In order to to that, we need to configure a Transformations connector.
  Click on the right menu on List transformations connections in the Output sections.

Then add a name and click on the Type tab. Select Metadata adjuster on the dropdown list then click save.



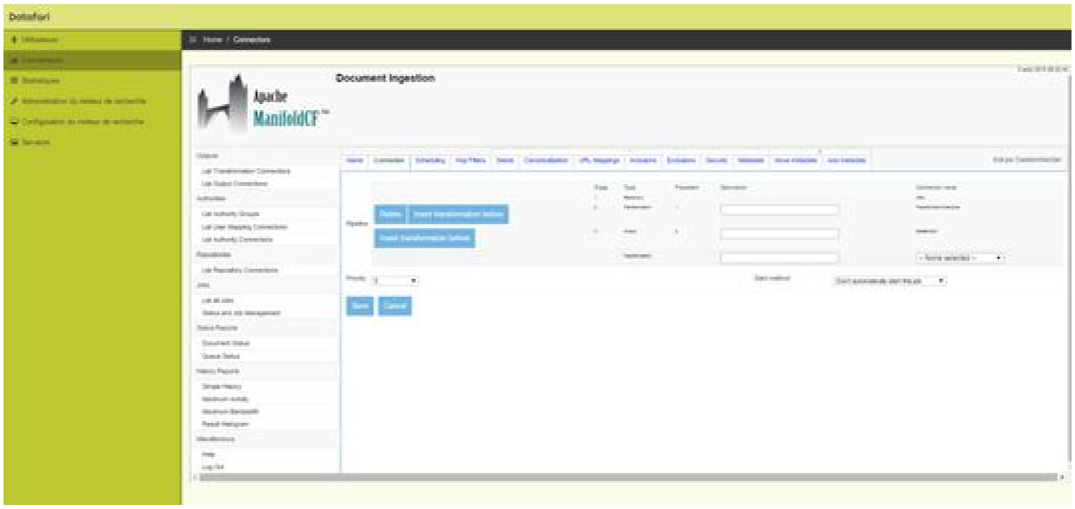
Document Ingestion

View Transformation Connection Status

- Now go back to the job configuration. Click on List all jobs on the Jobs section. Then click on the edit link for the "FranceLabsWebsite" job.

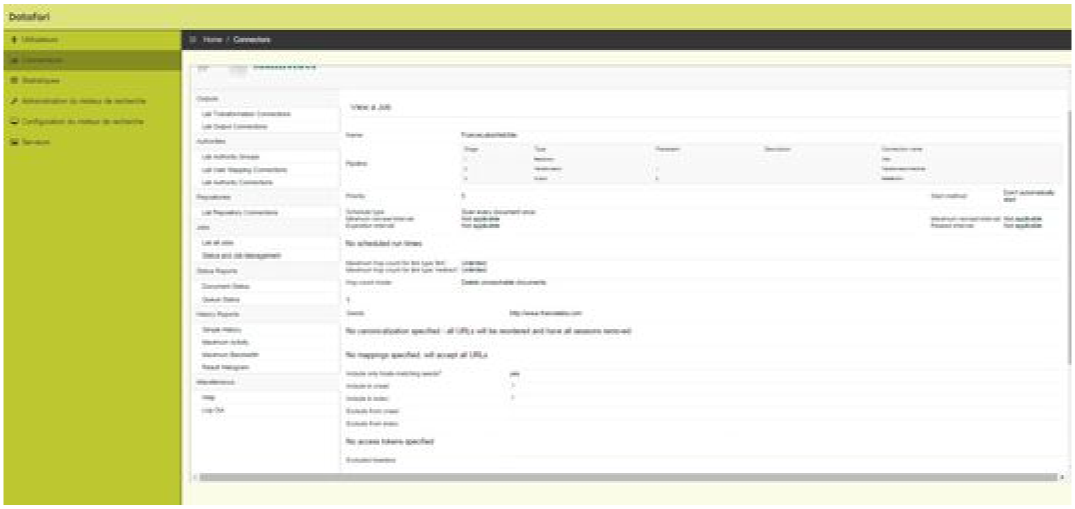
Go to the Connections tab and click on the dropdown select list in front of the Transformation line then select the new transformer freshly created. Then click on the button named "Insert transformation before".

So you should have the following configuration :



- New tabs appeared called Metadata, Move metadata and Add metadata. Click on Add metadata.
  Into parameter name you have to write the same name that the field you configured above so in our case it is named job. For the parameter value, I wrote FranceLabs.
  Then click on save.

The global configuration of the FranceLabs job is this one now :



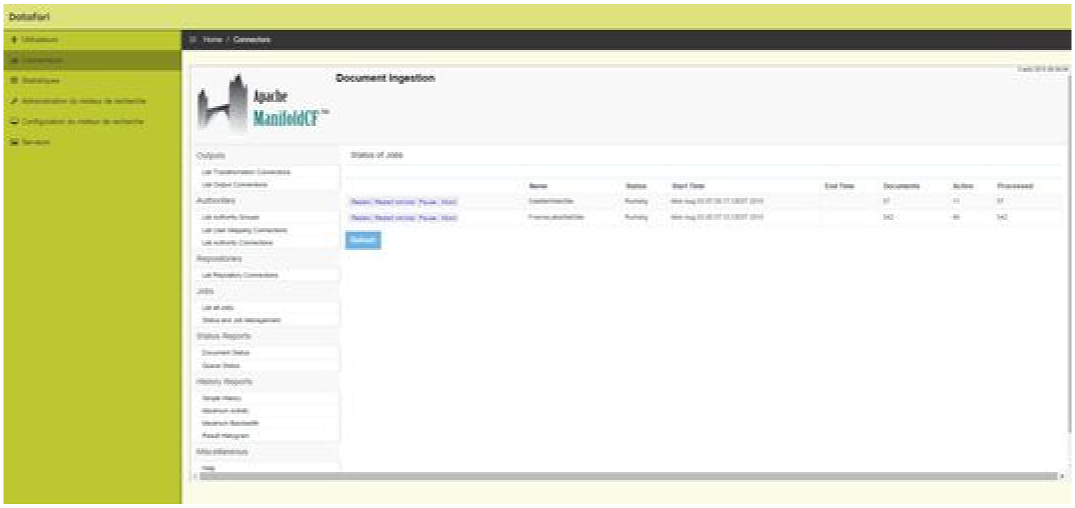
No scheduled run times

No mappings specified, will accept all URLs

No access tokens specified

We do the same thing for the Datafari job, the parameter name is job too and the value is Datafari.

- We can now launch the crawl job for the two jobs. Go to the Jobs section and click on "Status and Jobs management". Than click on Start for the two jobs.

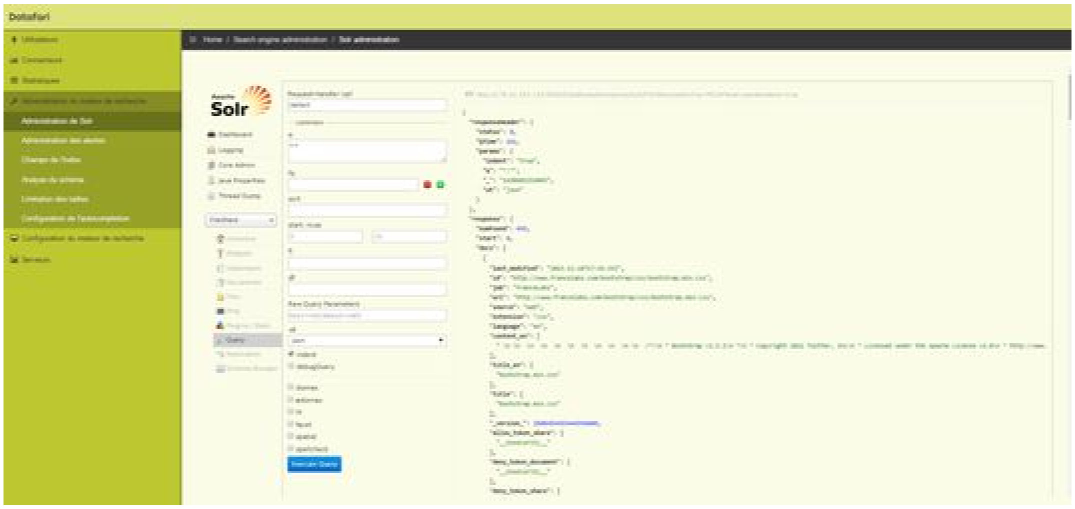


We have now to wait a little for some documents to be indexed.

- We are going to check if all is OK in the Solr Administration interface. Click on Search engine administration then on Solr administration in the right menu.

In the main window, select now FIleShare in the dropdown select list then click on Query in the tabs list below.

Then click on the blue button Execute Query : it will launch a default Solr query : we search anything on all the Solr corpus. We should obtain :



So we notice that in the fields list, we have the field job with the value FranceLabs. It is exactly what we expected : we have a new field with a different value fo each job configured in MCF.

We can now configure the Datafari User Interface !

Field already added to the Datafari schema

If you have already modified your indexation code and added your metadata, you have to respect these requirements for the Solr field :

```
<field name="job" type="string" indexed="true" stored="true" />
```

The field needs to be indexed and not tokenized. You can now configure the Datafari User Interface.

Configure the Datafari User interface

Our field is correctly present in our Solr schema and the values are present for each document. We just now have to configure the faceting in the Datafari UI.

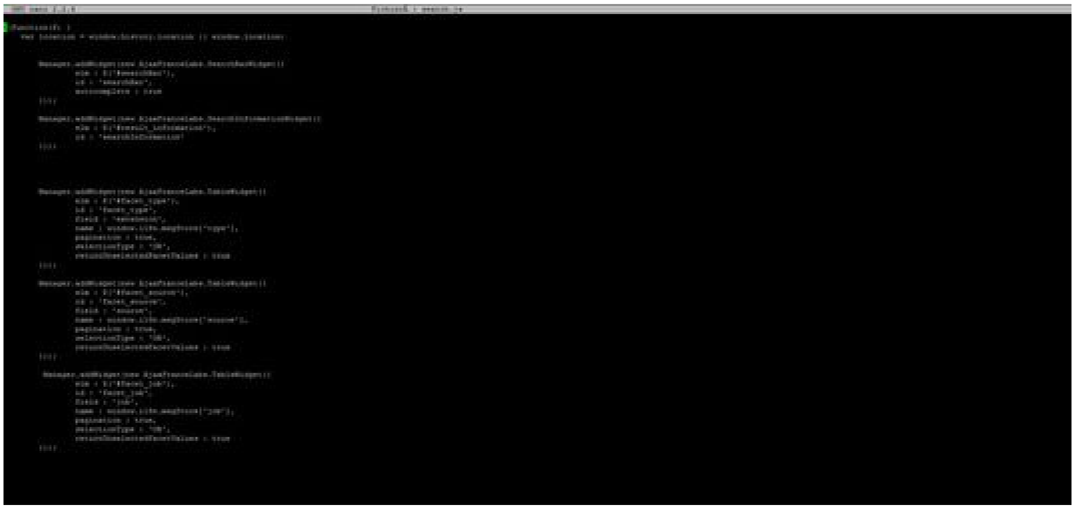
We need to modify the search.js file in order to add the TableFacet widget, we also have to add the facet display in the searchView.jsp and finally add the label of the facet in the i18n files : en.json and fr.json.

- First, edit the file search.js into /opt/datafari/tomcat/webapps/Datafari/js, add the widget in the code :

```
Manager.addWidget(new AjaxFranceLabs.TableWidget({
elm : $('#facet_job'),
id : 'facet_job',
field : 'job',
name : window.i18n.msgStore['job'],
pagination : true,
selectionType : 'OR',
returnUnselectedFacetValues : true
}));
```

We need to have an unique identifier in elm and id, we choose to call it facet_job, in the field parameter we indicate job, for the name we put job.

The file seems like that :

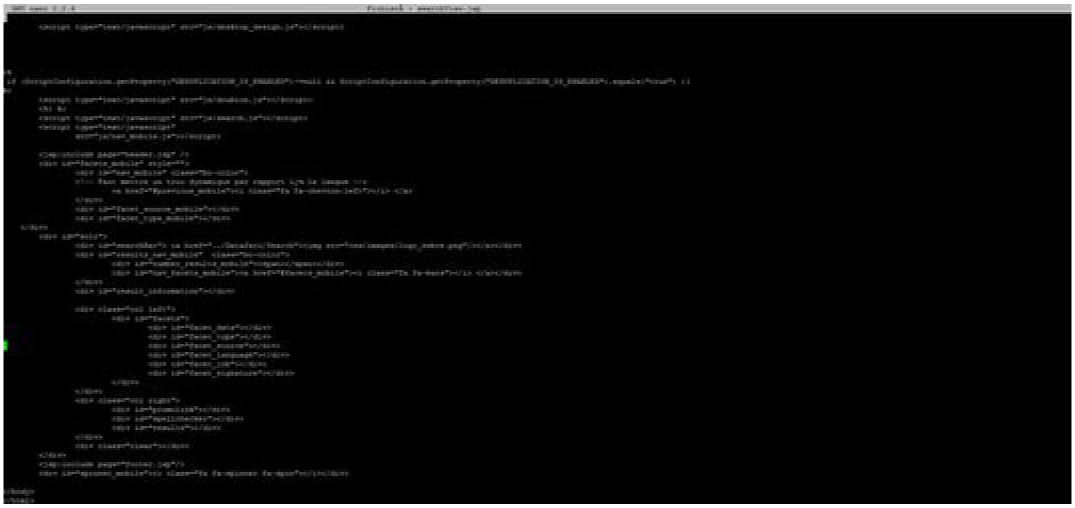


- Edit now the file searchView.jsp located in /opt/datafari/tomcat/webapps/Datafari

We add the facet element to display on the page.We add it on the div section called col left :

```
 <div id="facet_job"></div>
```

The order is important, by default we have the facets in this order : by date, type and source. If I want to the job facet just after the source facet, the file seems like that :



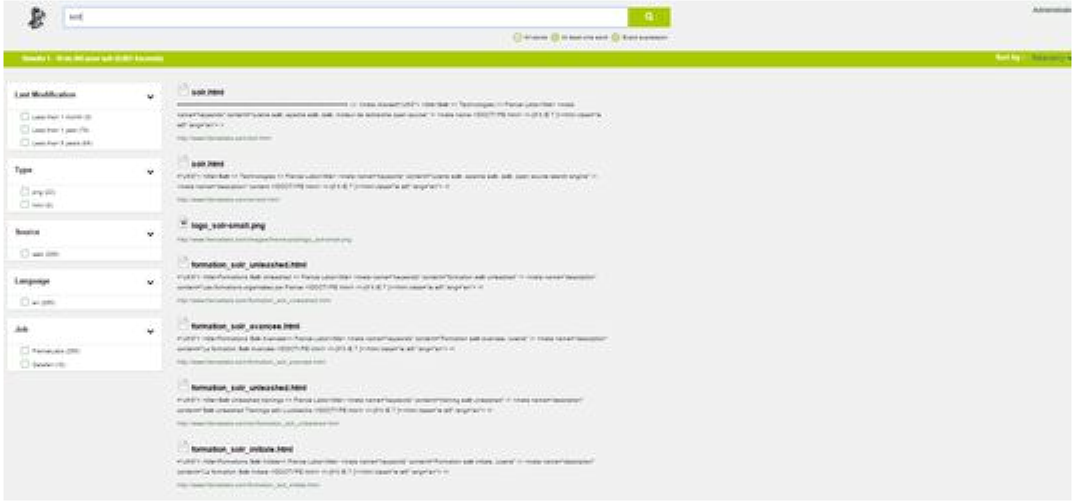
- Ok now the final step is to localize the label name of our new facet. Let's edit the two files en.json and fr.json in /opt/datafari/tomcat/webapps/Datafari/js/AjaxFranceLabs/locale :

We add a new line for the parameter 'job'. Here we add the same value for both the languages : Job. So the line to add is :

```
"job" : "Job",
```

Screenshot of the fr.json file :

- The configuration is now over. We can check if our new facet is correctly displayed in Datafari :


Last Modification
Type
Source
Language
Job
solr.html
solr.html
logo_solr-small.png
formation_solr_unleashed.html
formation_solr_avancee.html
formation_solr_unleashed.html
formation_solr_initiale.html

It seems to be the case. We can now filter the results by the ManifoldCF job : documents from Datafari website or France Labs website !

Fieldweights - Architecture and mechanism

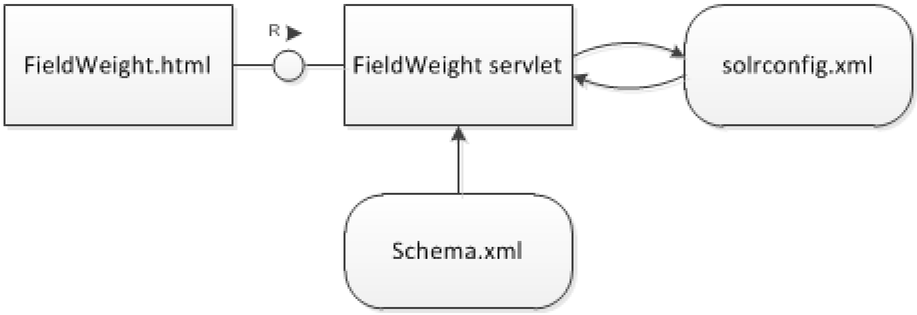
The fields weight are stored in the *solrconfig.xml* file of the FileShare core.

The admin UI proposed in Datafari communicates with the *FieldWeight* servlet, which in turn:

- reads the schema and solrconfig files,
- modifes the solrconfig.

When you go on the UI, it immediately sends a request towards the servlet to get the fields that are appended to the select.

Selecting a field will acquire a semaphore to prevent people from reading or writing a value that you are about to change.



> Note that there are two types of query with their own weight for each fields, so there is one semaphore per type, since you will not have to modify the same line in the file.

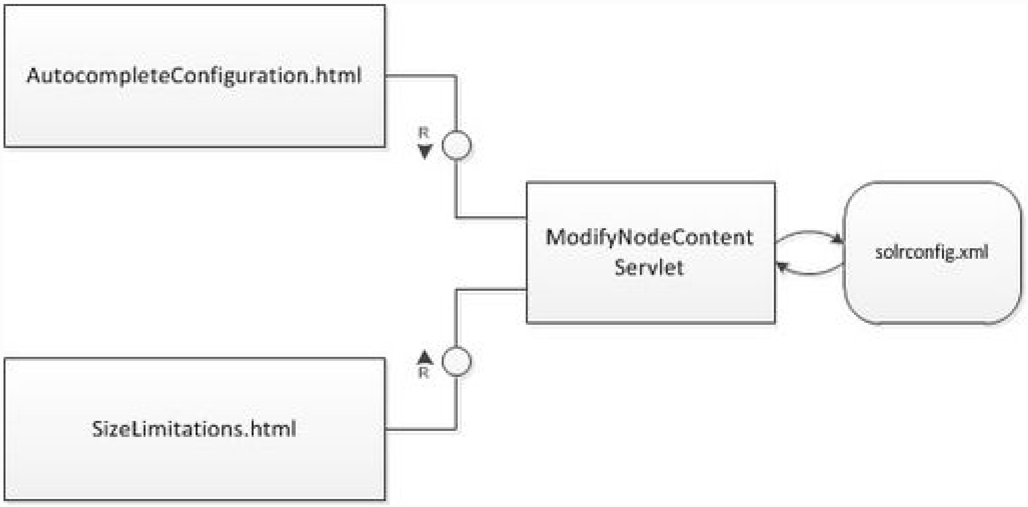
Once you selected a file, you will get its current weight (0 if not weighted). Confirming your modifications will send the request and release the semaphore. Due to the particular structure of the lines involved, this functionnality has its dedicated servlet, but if you want to modify the content of a node, you can use the ModifyNodeContent servlet.

> **ModifyNodeContent**
> It is a servlet that opens solrconfig.xml and looks for a node with a specified attribute. For instance, assuming there is a node with parameter "name" equal to "threshold", and we want to set the node value to 0.01:
>
> This is done by applying the following parameters:

- `<float name="threshold">0.005</float>`

  This line will then become :
- `<float name="threshold">0.01</float>`

This servlet is currently used by the size Limitation UI, and by the autocomplete configuration UI. This allows to prevent two or more users from modifying the same value. This servlet creates a semaphore on the doGet. The semaphore is based on the value of the attribute: in our previous example, “threshold”, someone can modify the size limitation and someone else can modify the threshold of the autocomplete.



Likes and Favorites

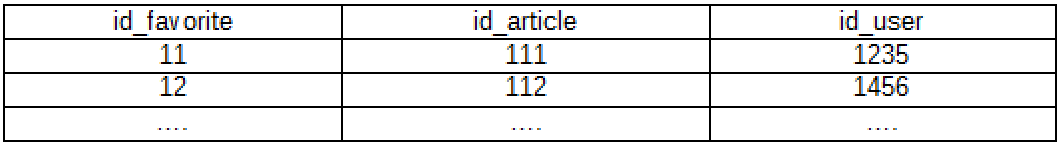
Problematic :

Starting with Datafari 2.0, users have the possibility to store pointers to documents shown in results list.

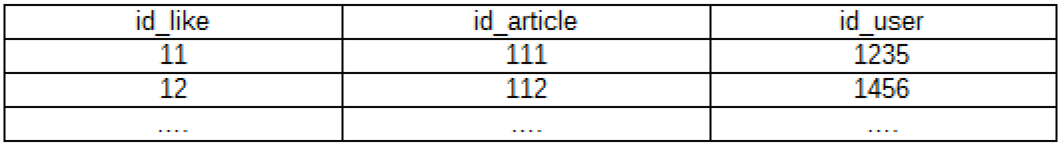
In the following document, “likes” corresponds to the functionality where a connected user can like or unlike a document, and “favorites” corresponds to the functionality where a connected user can save a pointer to a document he found in the results list.

Technically wise, we use Apache Cassandra for saving all the likes and favorites and Solr for saving the counters of likes of each document.

Using Cassandra, we cannot use standard relational db schemas, like the following:



id_fav orite	id_article	id_user
11	111	1235
12	112	1456
....	....	....

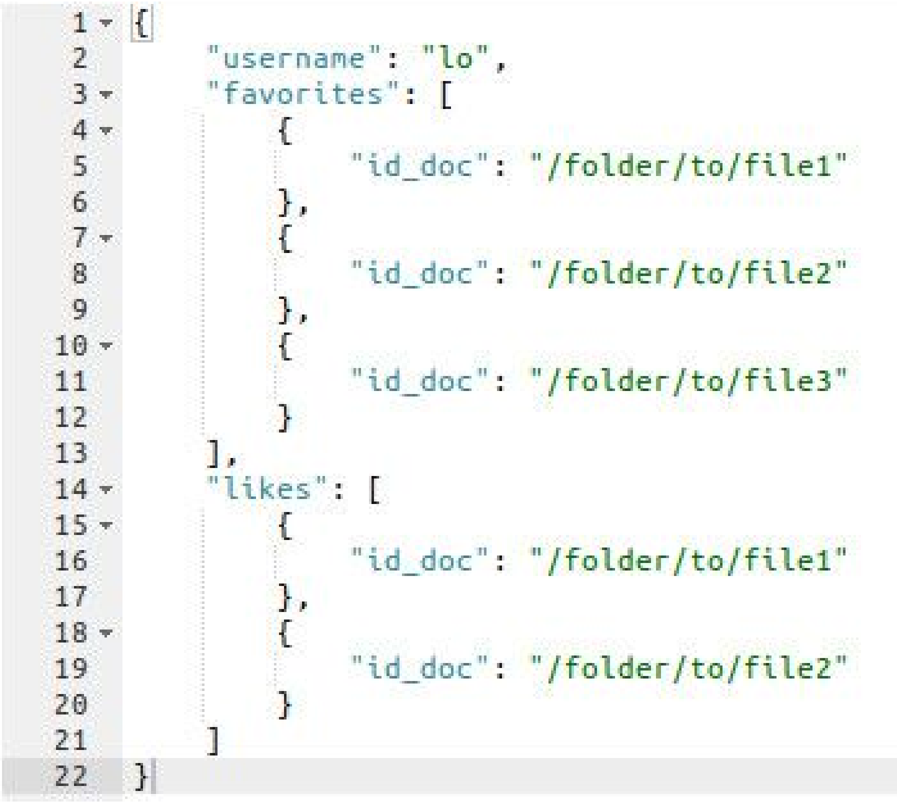


id_like	id_article	id_user
11	111	1235
12	112	1456
....	....	....

Modeling with Nosql in mind:

For our nosql modeling, we took our inspiration from this article http://docs.mongodb.org/manual/tutorial/aggregation-with-user-preference-data/ .

Applied to our problematic, we get the following :



```
{
    "username": "lo",
    "favorites": [
        {
            "id_doc": "/folder/to/file1"
        },
        {
            "id_doc": "/folder/to/file2"
        },
        {
            "id_doc": "/folder/to/file3"
        }
    ],
    "likes": [
        {
            "id_doc": "/folder/to/file1"
        },
        {
            "id_doc": "/folder/to/file2"
        }
    ]
}
```

That means we have dedicated to likes and favorites (which is distinct from the collection dedicated to user authentication and authorization).

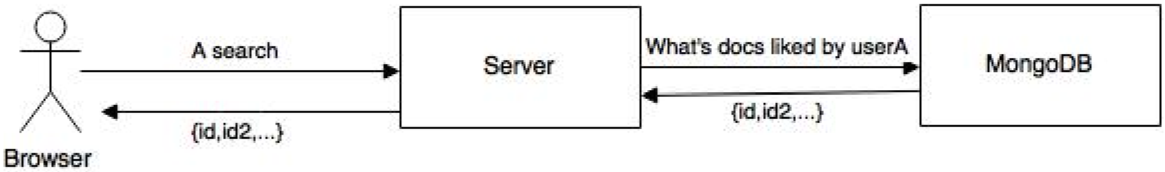
The request :

The aim is to get for the connected user the list of documents he liked out of the ones which are currently displayed. After the standard search query, we send only one http request which contains only the username . In return, we get all the documents in the index that are liked, as well as the list of favorites.

On the browser side, we go through this documents list and pick the ones that match the currently displayed documents to identify the favorites and the liked.

In order to display the total number of likes per documents, we use Solr and its External File Fields mechanism (https://cwiki.apache.org/confluence/display/solr/Working+with+External+Files+and+Processes).

Every time a connected user like or unlikes a document, we increment or decrement the counter of the document in the file located in SOLR_HOME/FileShare/data/externalk_nbLikes.



The Implementation :

There are two parts: Frontend and backend. For the frontend, we added a widget that builds the inteface and makes all the Ajax Requests. For the backend, we created two classes: Like and Favorite. We created the corresponding servlets that use these two classes and send back the result.

1. Frontend :

   The widget is called LikesAndFavoritesWidget. This widget inherits from ResultWidget and is only included in SearchView.jsp if the functionality of likes was activated by the admin.

This inheritance is important because we need to execute first the code of the ResultWidget and then the LikesAndFavoriteWidget.

Once the buildWidget is executed, we execute the beforeRequest which will add the fl nbLikes:field(nbLikes) in the Solr query so we can retrieve as part of the results the number of likes per document.

Then the afterRequest will fire. The afterRequest retrieves the documents liked and saved by the user from the server (ajax request) and stores it in window.globalVariableLikes and window.globalVariableFavorites. Then we parse the results retrieved from Solr and put the correct values in html by checking if the doc belong in the window.globalVariableLikes and window.globalVariableFavorites.

Next, we put a listener to the “Like” button and a listener for saving a favorite. Depending on which button is clicked, we send a request to add/delete the favorite or the like from the server and from the lists saved in the global variable window of javascript.

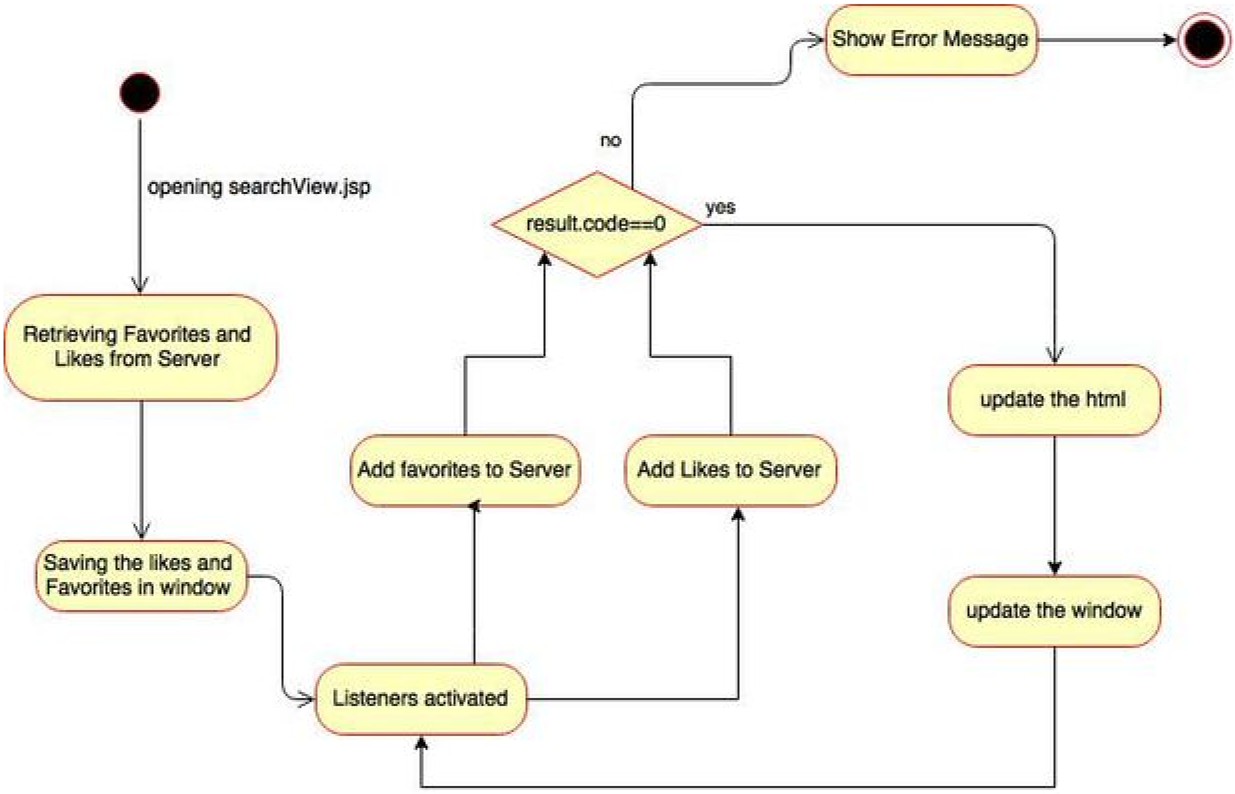


Illustration 1: State Diagram of AfterRequest of LikesAndFavoritesWidget

2. Backend :

We use two major classes which are Like And Favorite. They provide the method that will get, delete and add a Favorite or a Like by querying the Cassandra database. Whenever a change is needed to likes and favorites, we use directly these classes.

Like.class and Favorite.class belong to the package com.francelabs.datafari.user which also contains *UserConstants.class* where we put all the constant Strings used in this package.

We also use the following servlets:

- AddFavorite: add a document to favorites,
- AddLikes: add document to likes,
- DeleteFavorite: delete a document from favorites,
- Unlike: delete a document from the likes,
- GetLikesAndFavorites: gets all the likes and the favorites of a user,,, .

AddLike, AddFavorite, DeleteFavorite and Unlike are very similar. Therefore we will detail only one of them, the others can be derived from it. This is the sequence diagram involving the AddLike servlet:

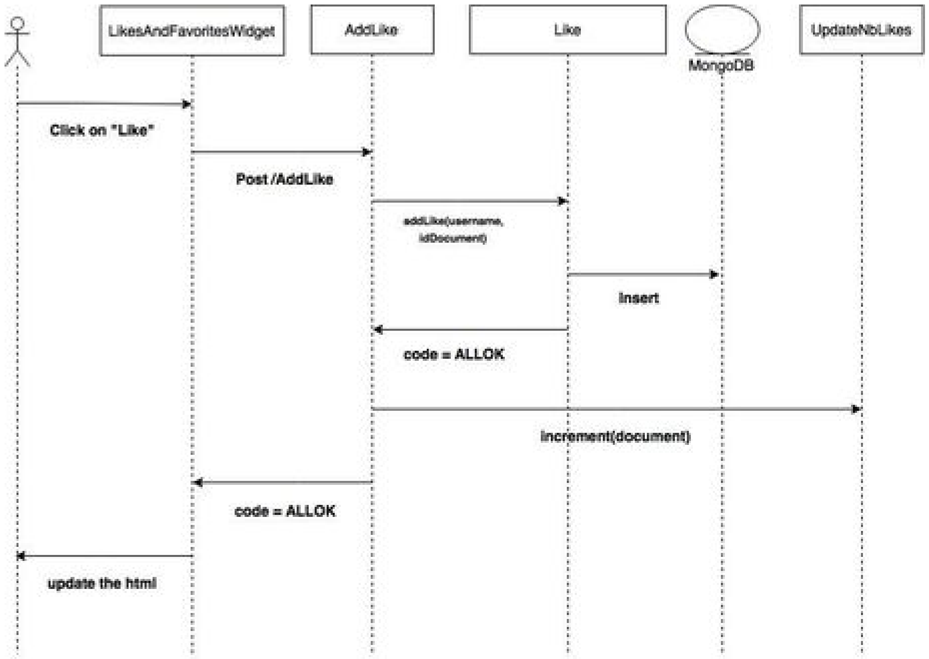


Illustration 2: Sequence Diagram of AddLike for a normal Behaviour

Here is the sequence diagram that involves the LikesAndFavoritesWidget to retrieve the likes and favorites of a user :

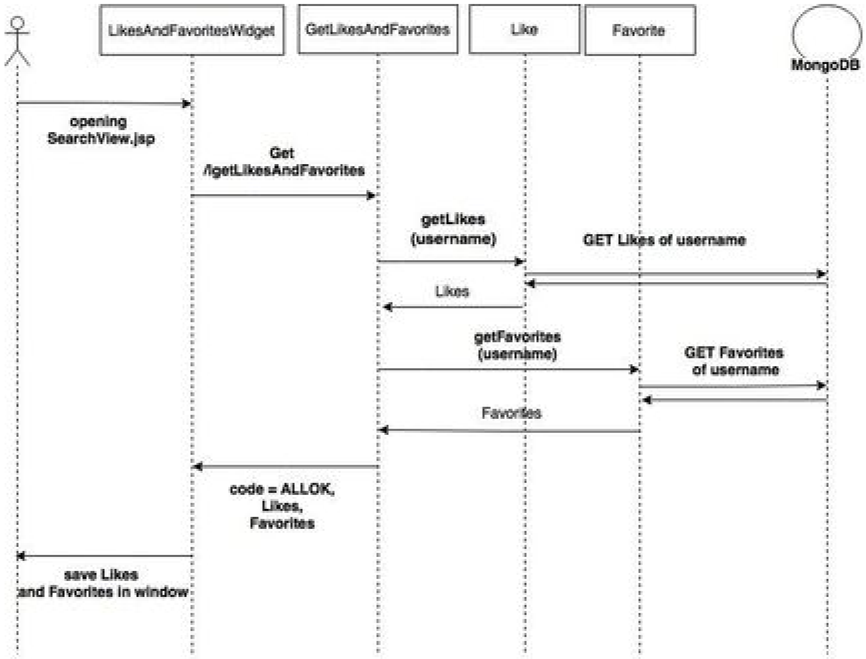


Illustration 3: Sequence Diagram of getting the likes and favorites from server

Logging

Datafari 3.0 comes with a revised way of logging. All the logs generated by Datafari and the other components (Manifoldcf, Solr, ELK) have been formated in order to be exploited by ELK.

Here is the new format :

```
[level] [timestamp] [thread name] - [process name]|[component
name]|[package name]|[message]
```

It is implemented through a log4j properties file for each Datafari component. The corresponding log4j pattern is the following:

```
%5p %d{ISO8601} (%t) - [process name]|[component name]|%c{3}|%m%n
```

> To understand the specific log4j patterns used, refer to the official doc: https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

For example, the following log4j conversion pattern has been defined in the log4j.properties file of Tomcat for the Datafari appender:

```
log4j.appender.datafari.layout.ConversionPattern=%5p %d{ISO8601} (%t) -
Tomcat|Datafari|%c{3}|%m%n
```

This configuration will generate log lines that look like this:

```
INFO 2016-07-25 16:38:08,374 (localhost-startStop-1) -
Tomcat|Datafari|datafari.startup.UserManagementLauncher|UserManagement
info
 INFO 2016-07-25 16:38:08,851 (localhost-startStop-1) -
Tomcat|Datafari|service.db.DBContextListerner|Cassandra client
initialized successfully
 INFO 2016-07-25 16:38:08,936 (localhost-startStop-1) -
Tomcat|Datafari|datafari.alerts.AlertsManager|Alert config file
successfully read
 INFO 2016-07-25 16:38:08,945 (localhost-startStop-1) -
Tomcat|Datafari|datafari.alerts.AlertsManager|Alert scheduler started
```

With this kind of format, Datafari can push every generated log line to Elasticsearch thanks to Logstash, and Kibana can be used to visualize and exploit them with efficiency.

ELK logs exploitation

> Starting with Datafari 3.1

When ELK is activated in Datafari, every log file generated by Datafari and its components is pushed to Elasticsearch thanks to Logstash. The configuration file of Logstash is the following: DATAFARI_HOME/elk/logstash/logstash-datafari.conf and the template used to create the Elasticsearch logs indices is DATAFARI_HOME/elk/logstash/templates/datafari-logs-template.json

If you open the logstash-datafari.conf file, you will notice that the configuration also includes the statistics and monitoring logs. It is because logstash can only manage one config file by process and one of the best practice is to limit the number of logstash processes.
This is the reason why you will also notice that the inputs are "marked" with a "type" which can be 'log' or 'exploit' and is used to apply a different parsing and insertion strategy for the inputs.

The other thing to notice in the logstash conf file is that the logs are inserted in a daily index and the logs with a "DEBUG" level are ignored. These lines are used to specify that the logs must be pushed to the index corresponding to the log date (variable %{index_date} which is created earlier in the conf file):

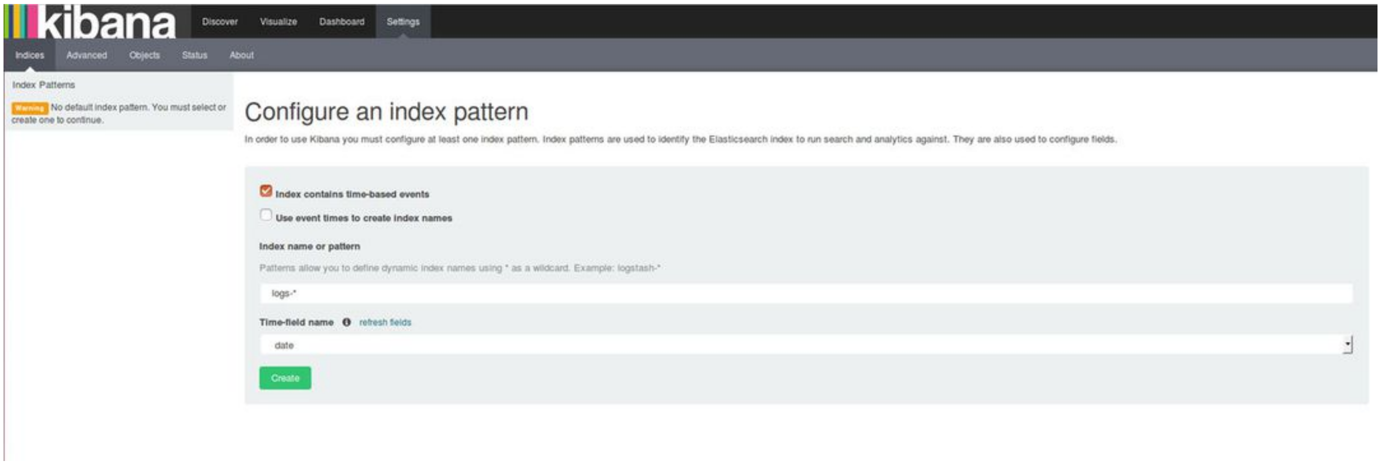
```
if [type] == "log" {
 if [level] != "DEBUG" {
  elasticsearch {
   hosts => ["localhost:9200"]
   index => "logs-%{index_date}"
   template => "./templates/datafari-logs-template.json"
   template_name => "datafari-logs"
   template_overwrite => true
   flush_size => 100
   idle_flush_time => 1
  }
 }
}
```

For example, the following log line:

```
INFO 2016-07-25 16:38:08,936 (localhost-startStop-1) -
Tomcat|Datafari|datafari.alerts.AlertsManager|Alert config file
successfully read
```

will be inserted in the Elasticsearch index named "logs-2016.07.25". This behavior has been implemented to facilitate the purge of Elasticsearch when necessary. All you need to do is to delete the indices that you consider too old and all the logs corresponding to the index date will be deleted. This is a quick way, with a low resource consumption, to purge Elasticsearch and it is recommended by the Elasticsearch best practices.

Once the logs are inserted in Elasticsearch, you will be able to consult them and create dashboards with Kibana which make it a powerful tool for monitoring. To create dashboards or visualizations for the logs, simply connect to Kibana either by the direct url http://[HOSTNAME]:5601 or through the admin UI of Datafari.
Go to the indices settings, click on the 'Add New' button, and add the new pattern 'logs-*'



No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☑ Index contains time-based events

☐ Use event times to create index names

Index name or pattern

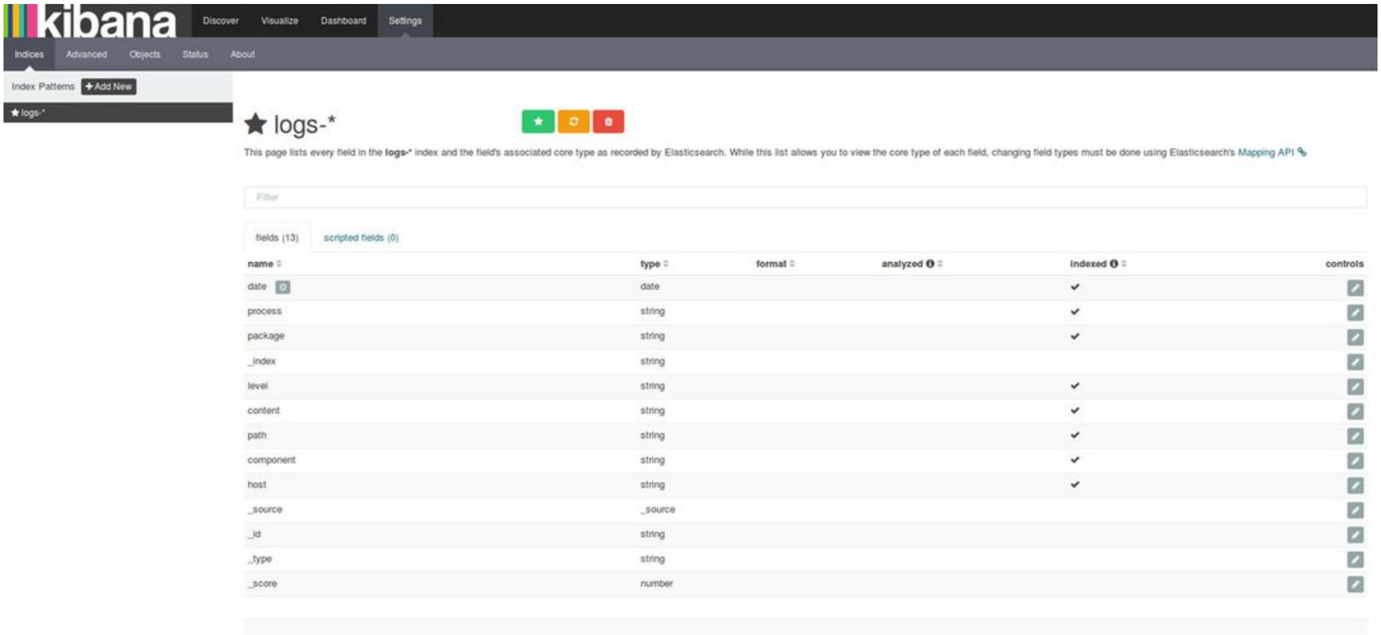
Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logs-*

Time-field name refresh fields

date

Check that Kibana detects the field 'date' as in the above screenshot and click on 'Create'. You should obtain the following result:



★ logs-*

This page lists every field in the **logs-*** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

name	type	format	analyzed	indexed	controls
date	date			✓	✎
process	string			✓	✎
package	string			✓	✎
_index	string				✎
level	string			✓	✎
content	string			✓	✎
path	string			✓	✎
component	string			✓	✎
host	string			✓	✎
_source	_source				✎
_id	string				✎
_type	string				✎
_score	number				✎

Now you can create visualizations and dashboards, but you can already some visualizations and dashboards made by France Labs by importing the file datafari-logs-kibana.json located in DATAFARI_HOME/elk/save. To import them, go to Setting => Objects in Kibana, click on the import button then select the json file.

Save search - technical doc

Datafari 3.2 introduces a new feature which allows you to save a search. This feature is divided in four parts:

- **The UI**: a new div element has been added to the file WebContent/searchView.jsp. The id of this div element is 'save_search' and the div is only added when an authenticated user is detected to prevent unsupported access to the feature.
- **The widget**: the javascript of the widget that adds the search button and implements its behavior is located in WebContent/js/AjaxFranceLabs/widgets/SaveSearch.widget.js and is instanciated in the file WebContent/js/search.js. The widget javascript file needs also to be imported in the file WebContent/searchView.jsp. The widget function is to save all the parameters of the query performed by Solr, thanks to the manager.store object.
- **The servlet**: Once the user clicks on the "Save search" button, he needs to enter a name and validate. When the 'Validate' button is pushed, the widgets sends the chosen search name and the search infos (the query performed by Solr) to the servlet "/saveSearch". The servlet retrieve the query parameters, the name entered by the user and the username, and insert everything in the Cassandra database. The servlet java code is located in src/main/java/com/francelabs/datafari/servlets/SaveSearch.java
- **The database table**: The servlet insert the saved search data into the datatable "search" of the Datafari database of Cassandra. This table is created thanks to the script datafari-cassandra/conf/dev-env/tables and is composed of three fields:
    - username: the user name
    - name: the search name entered by the user
    - request: the Solr query parameters corresponding to the saved search

Security

This section details the way Datafari manages security.

CustomCombinedRealm - Architecture and mechanism

In Datafari, we propose a user management following the standard Role Based Access Control mechanism.

We use two steps:

- the authentication step, to ensure a user is who he claims to be (using credentials validation),
- the authorization step, to ensure the authenticated user is allowed to see a certain part of Datafari (be it admin pages or secured documents in the search results list).

To manage that propery, we rely on the Tomcat Realm mechanisms. Still, there was no existing realm to satisfy our needs, which are to do the authentication either on a remote LDAP/AD or on our Cassandra, and to do the authorization on our Cassandra (and Solr for the search part, but this is another story). This is why we have created CustomCombinedRealm and a user data model in our Cassandra.

CustomCombinedRealm is a class that belongs to the package com.francelabs.datafari.realm**.**

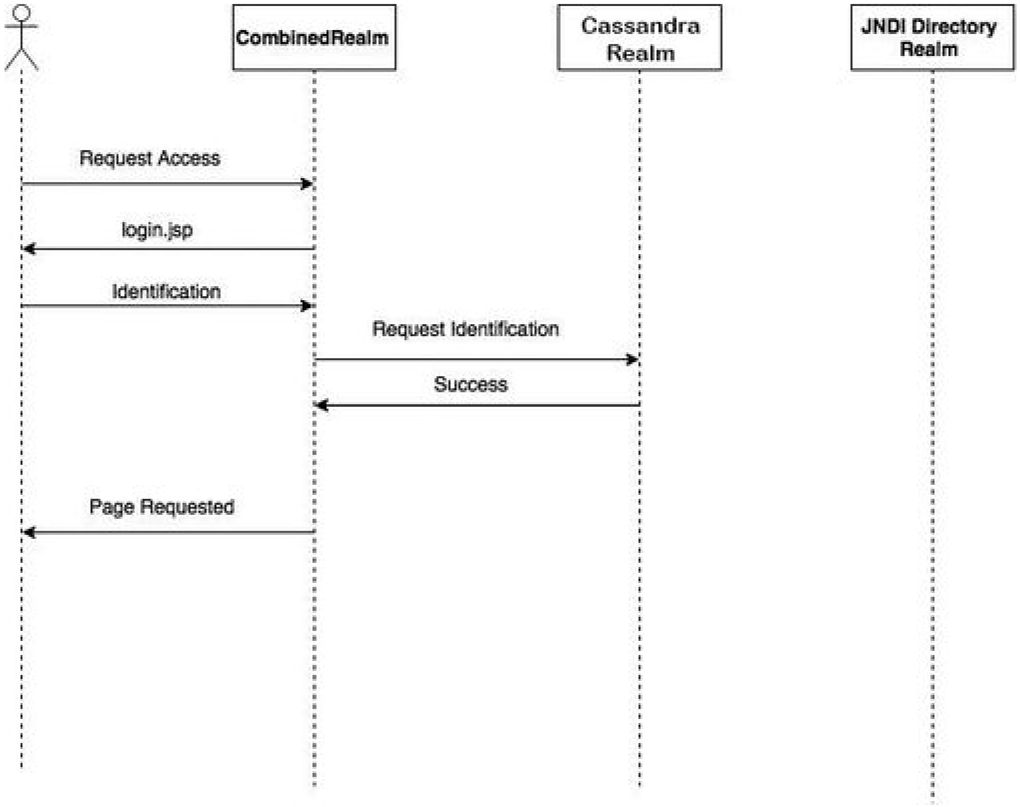
This class inherits from *CombinedRealm* and differs from its super class by collecting the roles.

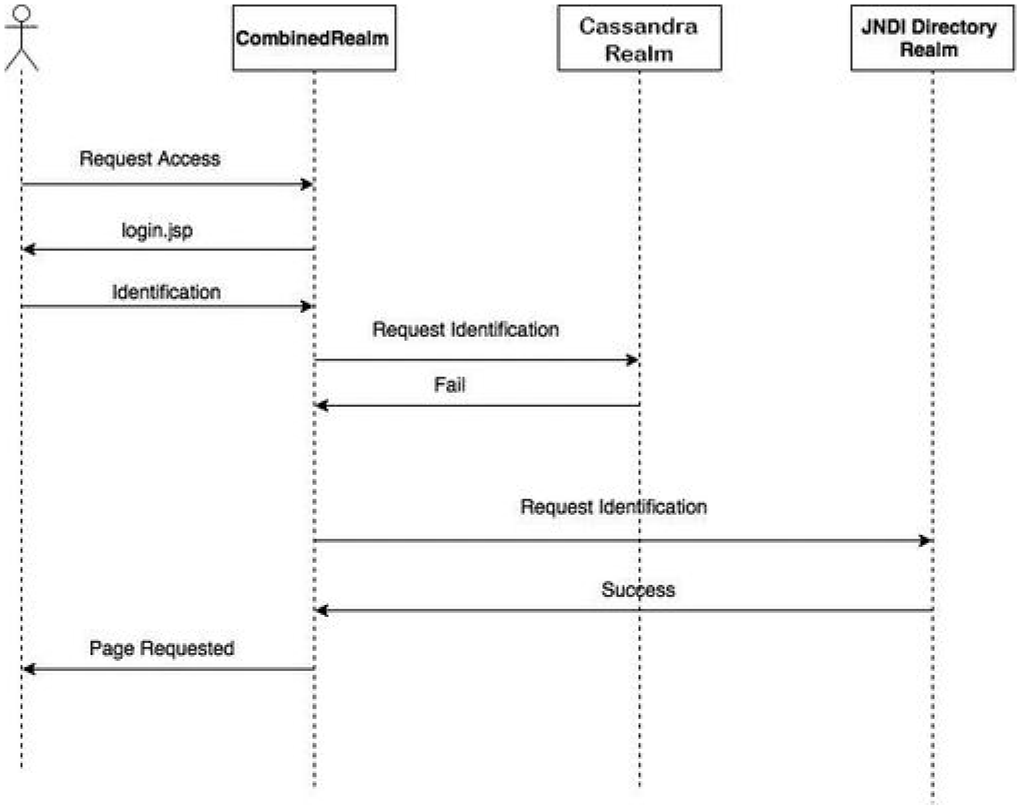
We developed this class to retrieve the roles of a user from our Cassandra database, and the user authentication either from AD/LDAP or from Cassandra. That means that even if we use LDAP for authentication, we will get the corresponding roles from Cassandra and not from LDAP.

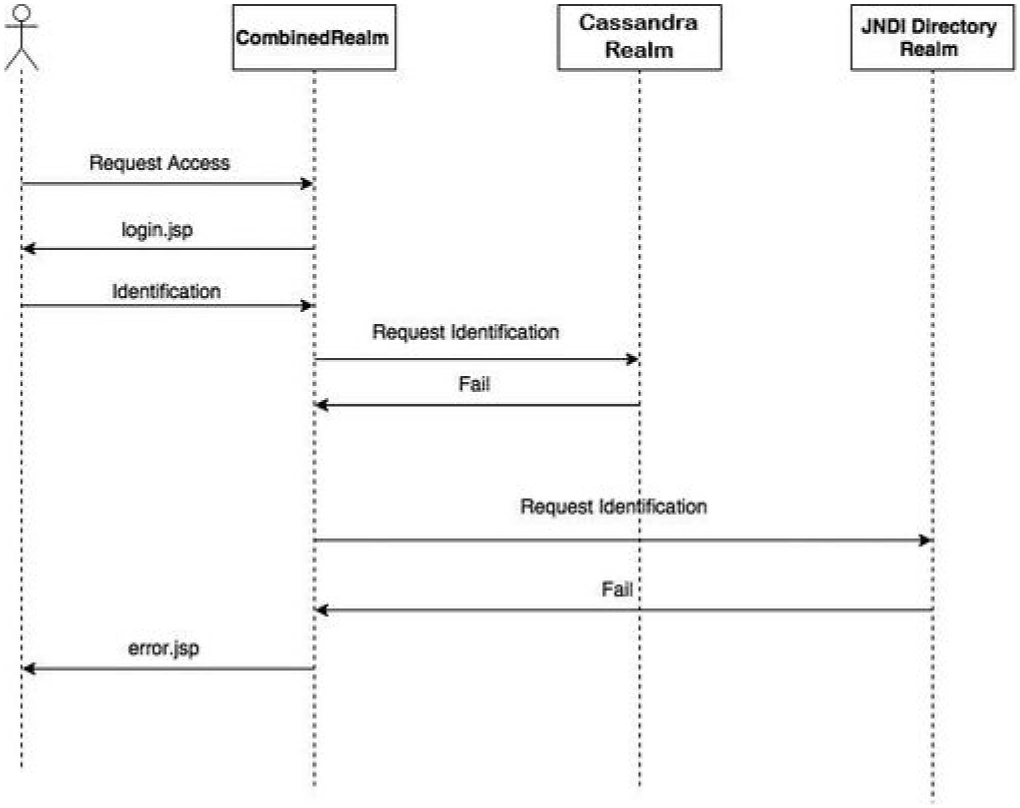
The process is as follows:

- we start by requesting an authentication in Cassandra.
- In case of success, we get the corresponding roles from it but if it fails we use LDAP for authentication.
- If the authentication succeeds, we get then the roles from Cassandra and if it fails we return an error to the user.

You can see below the illustration of these steps.

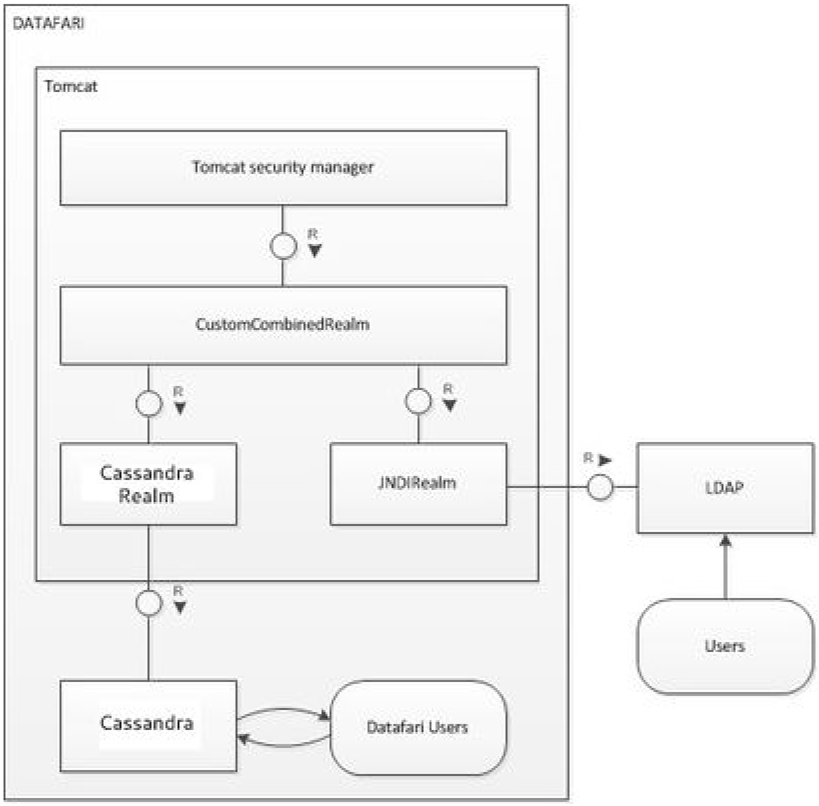






As shown above, CustomCombinedRealm communicates with the two Realm : ***CassandraRealm*** and the ***JNDIRealm*** **(AD/Ldap Realm). These classes will communicate respectively to there databases and check if the authentication succed and return the response to the CustomCombinedRealm.**

Here is the overall architecture to understand the way it works:



Understanding Security Realms

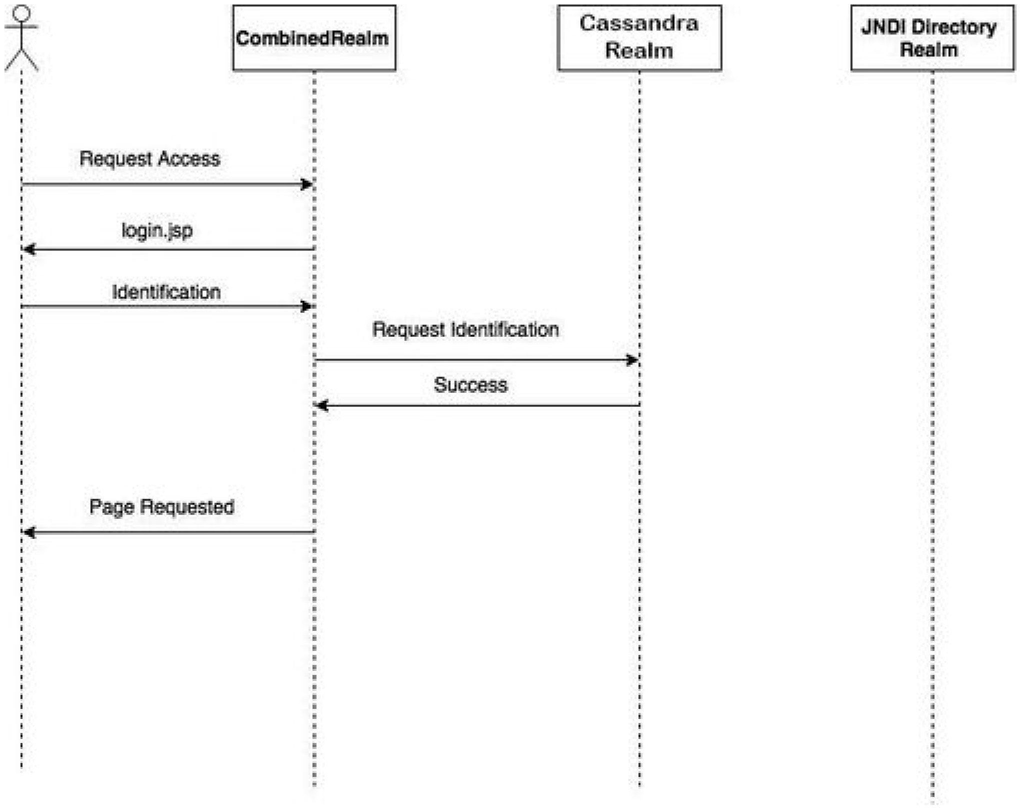
This document explains how user management is done in Datafari, following the RBAC (Role Based Access Control) model.

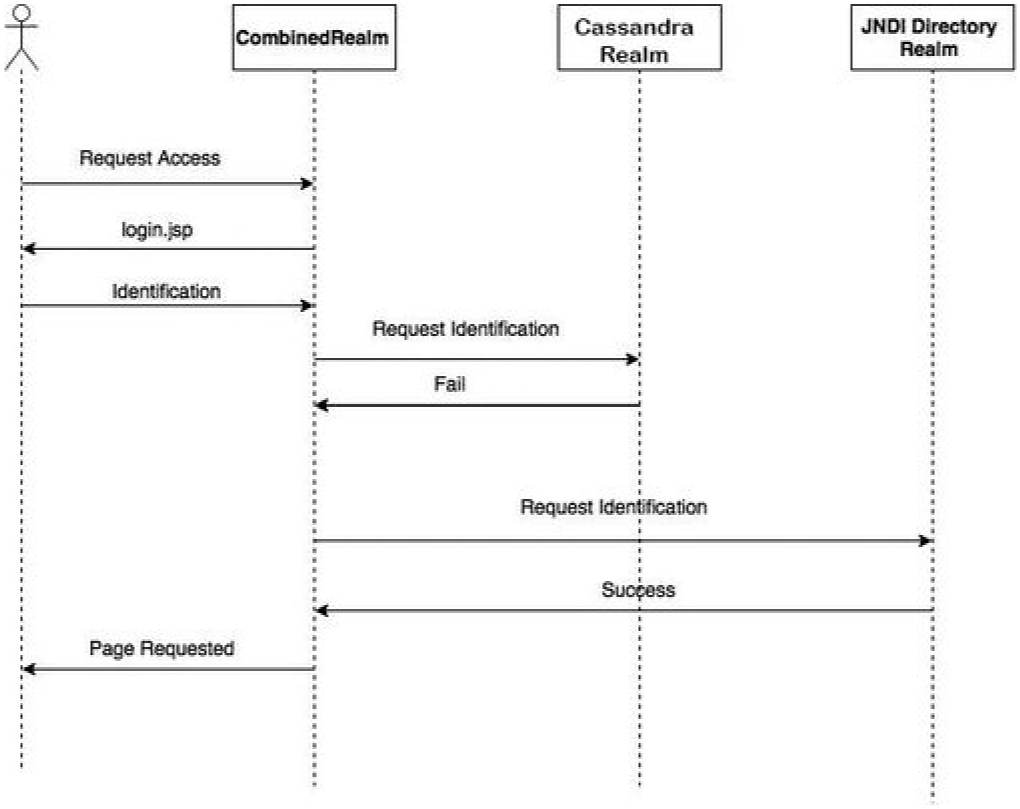
Datafari uses three Tomcat Realms for the authentication. This functionality is used to recognize the user and give him a personalized environment.

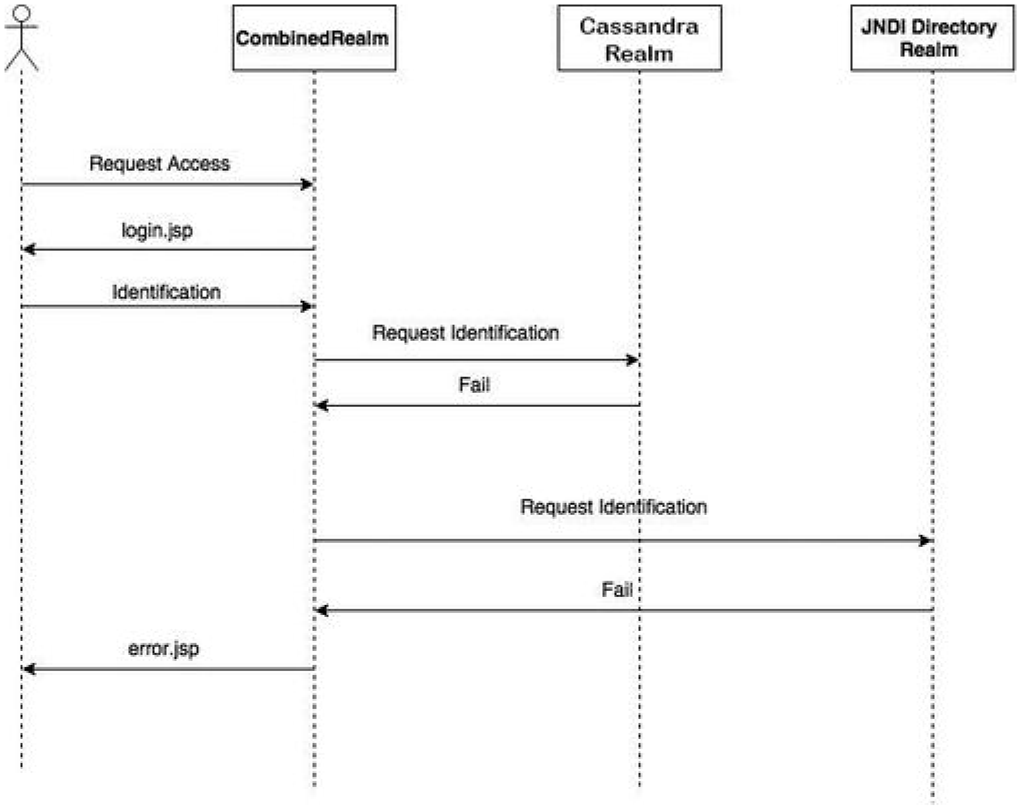
Also, every user has a specific role which will allow the user to have access to specific areas of Datafari and to prevent others from having this ability.

This functionality is of no use for persons who install Datafari on their personal computers.

The steps for authentication is illustrated in the three pictures below.


CombinedRealm
Cassandra Realm
JNDI Directory Realm
Request Access
login.jsp
Identification
Request Identification
Success
Page Requested


CombinedRealm
Cassandra Realm
JNDI Directory Realm
Request Access
login.jsp
Identification
Request Identification
Fail
Request Identification
Success
Page Requested



As shown above, when a user wants to authenticate, Datafari first calls *CombinedRealm*.

CombinedRealms triggers *CassandraRealm*, which will check the authentication using the database of Cassandra. If it fails, *CombinedRealm* triggers *JNDI DirectoryRealm* that will check the authentifcation using the AD/LDAP you have configured. These steps are done using the user credentials provided.

It is important to note that AD/LDAP is not required for Datafari to work : You can use only Cassandra without changing any setting.

Since Datafari uses 3 realms, they need to be configured. *CustomCombinedRealm* and *GenericCassandraRealm* should not require you to modify their configuration, as they are embedded in the Datafari package. Still, in case you want to customise them, it is feasible. The realm that requires configuration is the *JNDIRealm*, for the remote AD/LDAP connection. These configurations are done in *context.xml*, which is in *WebContent/META-INF* :

```
<?xml version="1.0" encoding="UTF-8"?>

<Context>

 <Realm
  className="com.francelabs.realm.CustomCombinedRealm">
  <Realm
     authDB="db-containing"
     authCollection="users"
     authUserField="username"
     authPasswordField="password"
     authRoleField="role"
     className="com.francelabs.realm.GenericMongoRealm"
     defaultDbHost="localhost"
     defaultDbPass=""
     defaultDbUser=""
     defaultRole="user"
     digest="SHA-256"/>
  <Realm
     className="org.apache.catalina.realm.JNDIRealm"
     connectionURL="ldap://ldap.forumsys.com:389"
     userPattern="uid={0},dc=example,dc=com"
     connectionName="cn=read-only-admin,dc=example,dc=com"
     connectionPassword="password"/>
 </Realm>
</Context>
```

Regarding the last Realm above, which relates to the AD/LDAP, its configuration depends on your AD/LDAP system. To configure it properly, please follow the Tomcat 7 JNDIRrealm howto page (url: https://tomcat.apache.org/tomcat-7.0-doc/realm-howto.html)

Regarding the Cassandra realm configuration: by default, the Cassandra database requires no username and password. Still, you should definitely add a password and a username for the connection your Cassandra. Once this is done, fill in these information in *defaultDbUser* and *defaultDbPass*.

Semantics

In this section, you will find information on the "semantics" capabilities of Datafari. We put semantics in brackets, because some of you may consider that certain functionalities are not real semantics. But since we noticed some of our competitors don't hesitate to flag as semantic the simple use of a dictionary to handle synonyms, we've decided to create this section. So you will find here details about the management of synonyms and stopwords, promolinks, and ontologies.

Link an ontology

Datafari gives you the opportunity to link an ontology to it, in order to enrich your documents with the ontology data and then use these additional infos in the search phase.

To do so, the first step is to add the OntologyUpdateProcessor as a custom UpdateProcessor in the file *{Datafari_Home}/solr/solr_home/FileShare/conf/customs_solrconfig/custom_update_processors.xml*

The OntologyUpdateProcessor will index, for each document inserted in Solr, the labels of the corresponding node into the specified ontology but also its parents and children labels and URIs.
It has several parameters that can be set:

- **enabled** : boolean value to enable/disable the OntologyUpdateProcessor
- **annotationField** : *[REQUIRED]* the field in the input document that contains the annotation URI used as the reference in the ontology.

> Note that this field must be added to the document by yourself: Datafari does not provide any plugin or other mechanism to do it. You may want to modify the DatafariUpdateProcessor or make your own UpdateProcessor to do so.

- **ontologyURI** : *[REQUIRED]* the location of the ontology, such as http://francelabs.com/ontology/owl.rdf or file:///ontology/owl.rdf

> Datafari only supports OWL ontology format for now, so the ontologyURI parameter has to reference a OWL formated ontology

- **labelField** : the field in your schema that should be used for the annotation's label(s). Default: *ontology_labels*
- **childField** : the field to be used for child document references. These are direct (ie. single-step) relationships *down* the hierarchy. Default: *ontology_children*
- **parentField** : the field to be used for parent document references. These are direct relationships *up* the hierarchy. Default: *ontology_parents*
- **childLabel** : the field to be used for child documents labels. Default: *ontology_children_labels*
- **parentLabel** : the field to be used for parent documents labels. Default: *ontology_parents_labels*
- **useLanguages** : (boolean) should language distinction for label fields be used ? If this is set to *true*, the processor will create additional label fields for each language found, using the language as a suffix, for instance *ontology_parents_labels*_fr, *ontology_parents_labels_en*
  The field format is [Original_Label_Field]_[Language]
  This is useful if you want to have, in addition to the original label fields which will contain all values, specific label fields containing values of the concerned language. But remember that the more the Ontology contains languages, the more it will create label fields.
  Default: *false*
- **includeIndirect** : (boolean) should indirect parent/child relationships also be indexed ? If this is set to *true*, *all* ancestor and descendant labels and URIs will also be added to the document. Default: *false*.
- **descendantsField** : the field to be used for the full set of descendant references. These are direct AND indirect relationships *down* the hierarchy. Default: *ontology_descendants*
- **ancestorsField** : the field to be used for the full set of ancestor references. These are direct AND indirect relationships *up* the hierarchy. Default: *ontology_ancestors*
- **descendantsLabel** : the field to be used for descendant documents labels. Default: *ontology_descendants_labels*
- **ancestorsLabel** : the field to be used for ancestor documents labels. Default: *ontology_ancestors_labels*

> If you decide not to use the default values for the fields, DO NOT FORGET to add them in the appropriate custom schema file(s) which are located in {Datafari_Home}/solr/solr_home/FileShare/conf/customs_schema
> Otherwise none of the ontology features will work !

To implement the OntologyUpdateProcessor, follow these steps:

- open the '*custom_libs.xml*' file in *{Datafari_installation_folder}/solr/solr_home/FileShare/conf/customs_solrconfig/*
- add the following line (replace the '<to_remove_tag>') :

```
<lib dir="./lib/jena"/>
```

> Datafari uses Apache Jena to load an ontology

- open the '*custom_update_processors.xml*' file in the same location
- add the following lines(still replace the '<to_remove_tag>'):

```
<processor
class="com.francelabs.datafari.updateprocessor.OntologyUpdateProcessorFactory">
 <bool name="enabled">true</bool>
 <str name="annotationField">ontology_annotation</str>

 <!-- Location of the ontology -->
 <str name="ontologyURI">file:///owl.rdf</str>
</processor>
```

> Of course you have to replace the values by yours and also add the optional parameters described up above with what you want

- Restart Datafari
- Crawl (again) your documents to add to them the new fields generated by the OntologyUpdateProcessor

Then, you can configure Datafari to use the new Ontology infos on the search requests :

- open the 'datafari.properties' file in {Datafari_installation_folder}/tomcat/conf/
- Modify those lines to set the ontologyEnabled property to *true* and the other lines to match with your ontology configuration:

```
ontologyEnabled=true
ontologyLanguageSelection=true
ontologyNodeLabels=ontology_labels
ontologyChildrenLabels=ontology_children_labels
ontologyParentsLabels=ontology_parents_labels
```

ontologyEnabled : (boolean) Enable or disable the ontology use
ontologyLanguageSelection : (boolean) should use the Datafari selected language for the ontology labels ? If *true*, be sure that you used the **OntologyUpdateProcessor** with the parameter **userLanguages** set to *true* during the crawl of the documents by MCF
ontologyNodeLabels : the field in your schema that contains the annotation's label(s)
ontologyChildrenLabels : the field in your schema that contains the child documents labels
ontologyParentsLabels : the field in your schema that contains the parent documents labels

Your Datafari is now ready to use the ontology additional fields during search queries thanks to the OntologySuggestion.widget

Promolinks - architecture and mechanism

Promolinks are indexed in a solr Core. The configuration is located in $DATAFARI_HOME/solr/solr_home/Promolinks/conf

A Promolink contains five fields :

- a keyword (mandatory field) which is unique since it is also the id of the PromoLink,
- a title (mandatory field),
- a content (mandatory field),
- a starting date (before this date the PromoLink will not be displayed) (optional),
- an ending date (after this date the capsule will not be printed) (optional).

You can see all the existing Promolinks in the Admin UI. You can also add or delete or edit a Promolink on this page. This UI communicates with the admin servlet which communicates with the solr core.

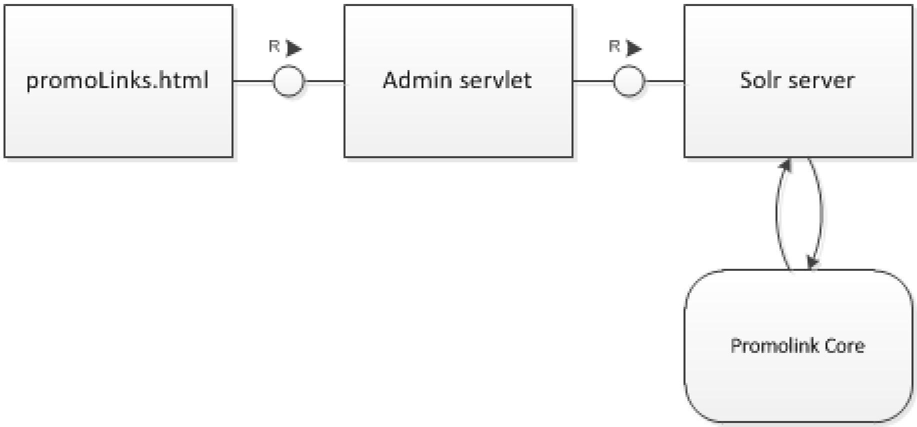
When you make a search on Datafari, once SearchProxy's doGet, two queries are sent:

- One is sent to FileShare to retrieve the documents
- Another is sent to the PromoLink core.
    - If a promolink is found, there will be a check of the dates (if those are filled).
    - If dates are not filled, the promolink will be considered as valid and it will be put in the promolinkSearchComponent key of the json object returned at the end.
    - Then the promolink widget will print it at the beginning of the result list.

The Solr fieldtype chosen for the field keywork is named capsuleType :

```
<fieldType name="capsuleType" class="solr.TextField"
positionIncrementGap="100">
<analyzer>
<tokenizer class="solr.WhitespaceTokenizerFactory"/>
<filter class="solr.LowerCaseFilterFactory" />
<filter class="solr.ASCIIFoldingFilterFactory"/>
</analyzer>
</fieldType>
```

Basically it splits the token at each blank character, it puts the string in lowercases and removes the accentuated characters. So it supports multiple words.



Synonyms and stopwords

Those two functionnalities are very similar since they use the corresponding Solr basic functionnalities that rely on a text file.

Unless it has been modified, those text files are located in the conf folder of the FileShare core.

The Admin UI allows the user to see and modify the files.

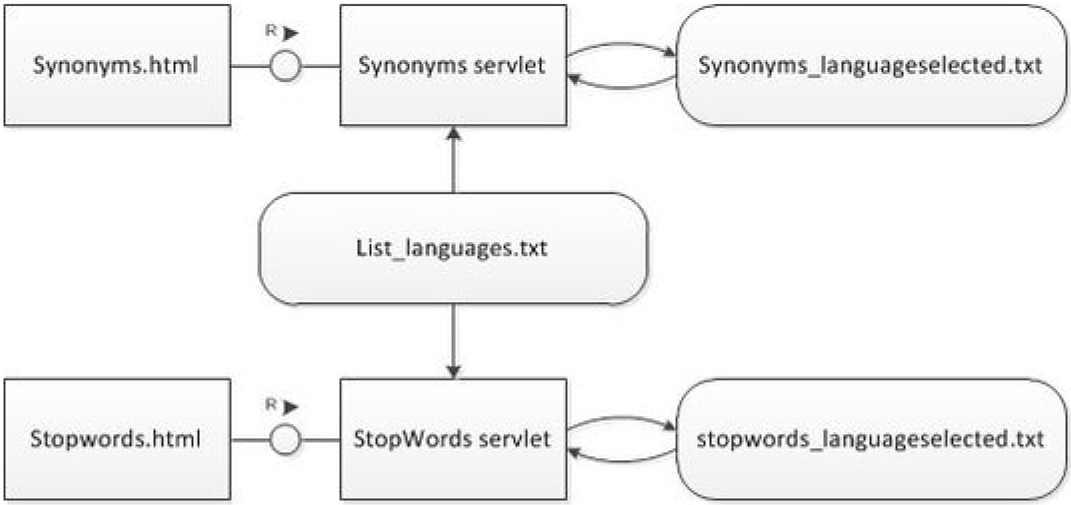
The syntax for the synonyms is as follows :

```
i-pod, i pod => ipod,
sea biscuit, sea biscit => seabiscuit
```

The syntax for the stopwords file is the following one :

```
a
an
to
from
```

To prevent two people from modifying the same text file at the same moment, the servlets dedicated to these functionnalities use a mutex semaphore. There is also a list_language.txt that you have to fill, which defines where to apply the semaphores.



You can see the effect of the stopword filter in the solr admin UI analysis.

Setup Development Environments

This section explains how to set up your development environment for the different versions of Datafari.

Setup development environnement with Eclipse (Linux)

DEPECRATED
This page is deprecated as of Datafari 2.0

Clone repository

git clone https://github.com/francelabs/datafari.git

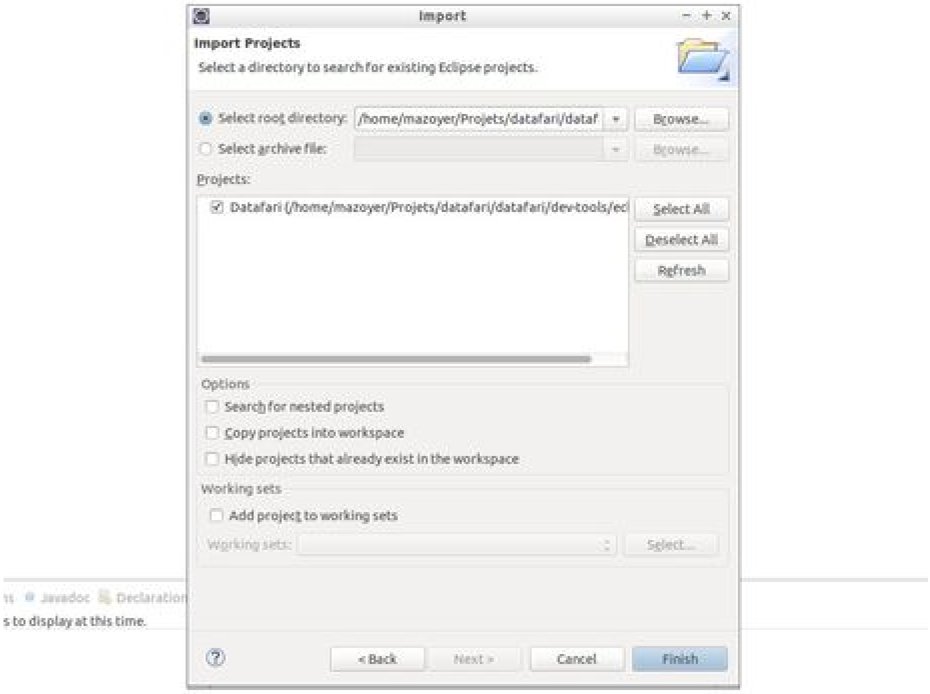
Open the project in Eclipse

Open a terminal. Navigate to the folder datafari. We now call it DATAFARI_SRC.

Run an : "ant eclipse" in DATAFARI_SRC.

In eclipse, do an « import existing project into workspace ».

Select the DATAFARI_SRC folder and click Finish.



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /home/mazoyer/Projets/datafari/dataf

○ Select archive file:

Projects:

☑ Datafari (/home/mazoyer/Projets/datafari/datafari/dev-tools/ecl

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

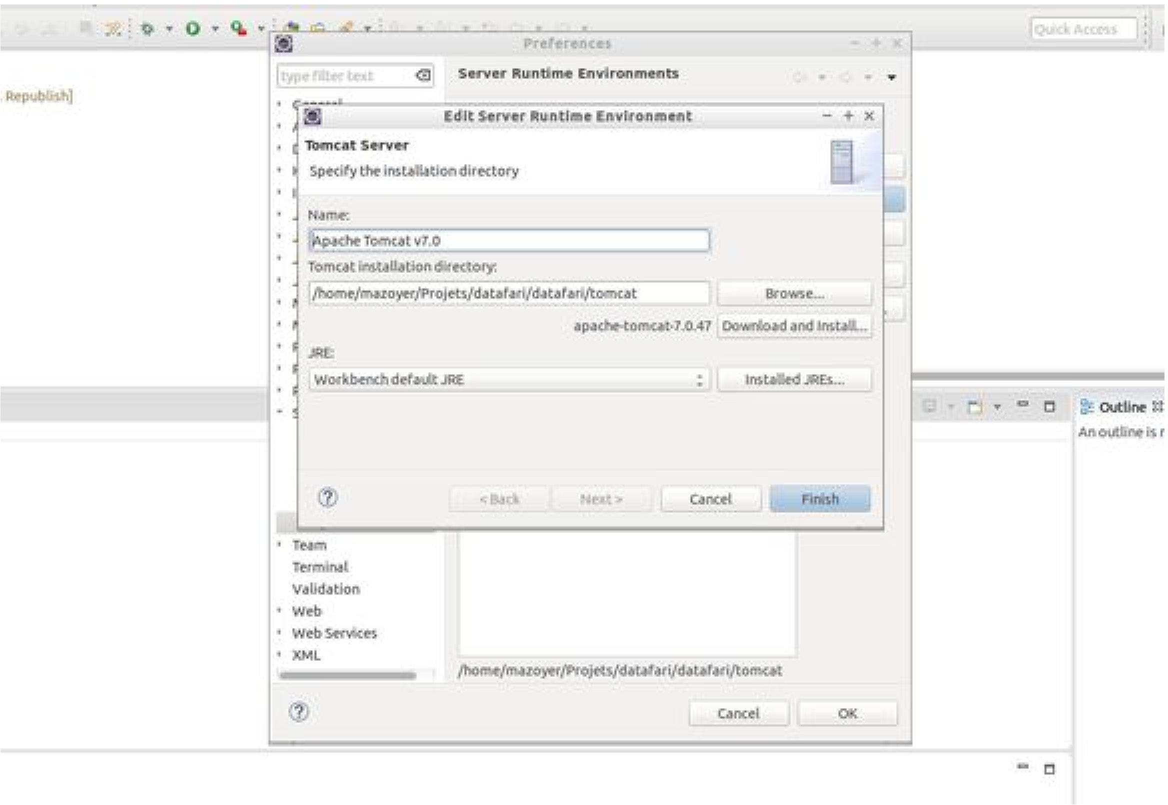
s to display at this time.

Setup the Tomcat server in Eclipse

Go to Window => Preferences => Server => Runtime Environments => Add…

and select Apache Tomcat V. 7.0

Select DATAFARI_SRC/tomcat for the Tomcat Installation directory and Finish.



Edit Server Runtime Environment

Tomcat Server

Specify the installation directory

Name: Apache Tomcat v7.0

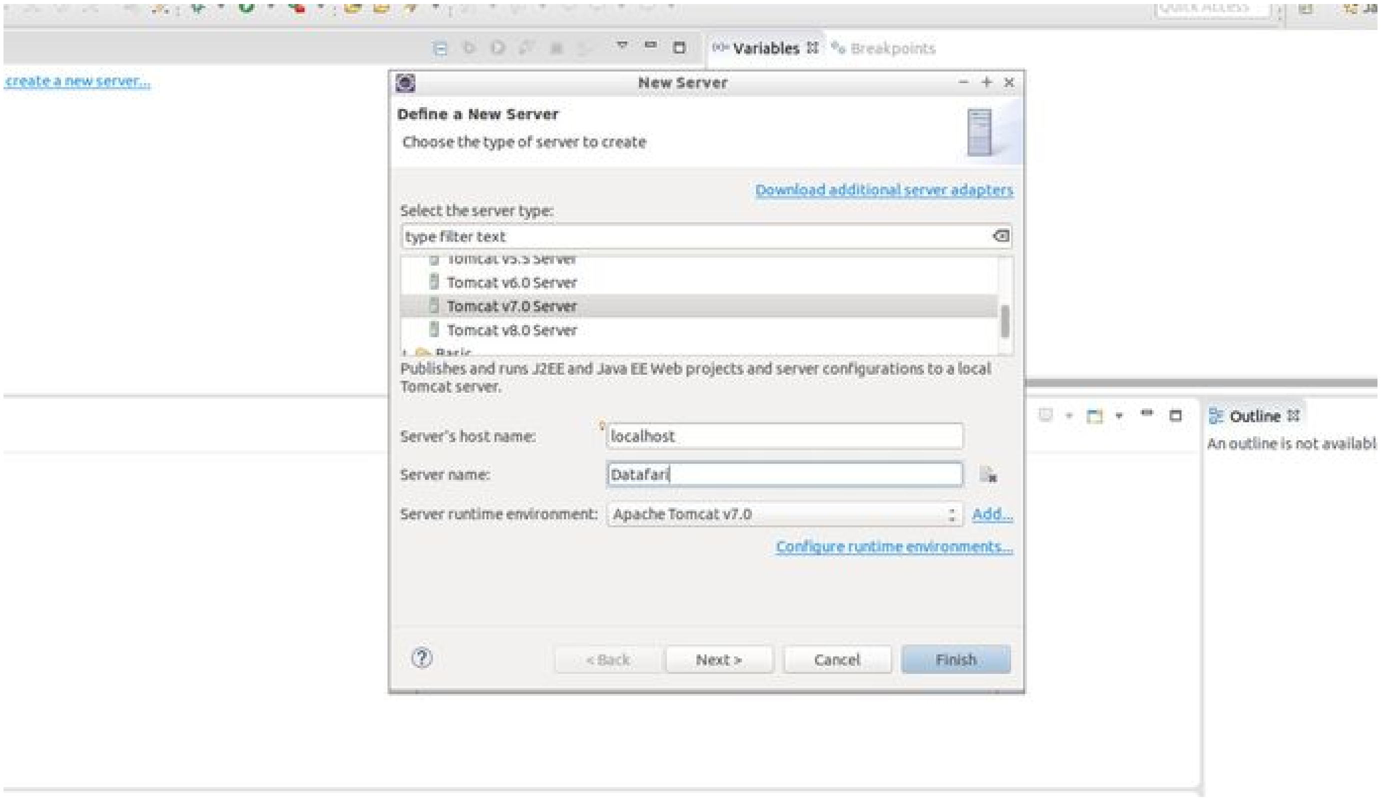
Tomcat installation directory: /home/mazoyer/Projets/datafari/datafari/tomcat

apache-tomcat-7.0.47

JRE: Workbench default JRE

/home/mazoyer/Projets/datafari/datafari/tomcat

In a server windows (for example in Debug Perspectives), click on "add a server", select a "tomcat 7.0 server" and set the server name to "Datafari":



New Server

Define a New Server

Choose the type of server to create

Select the server type: Tomcat v7.0 Server

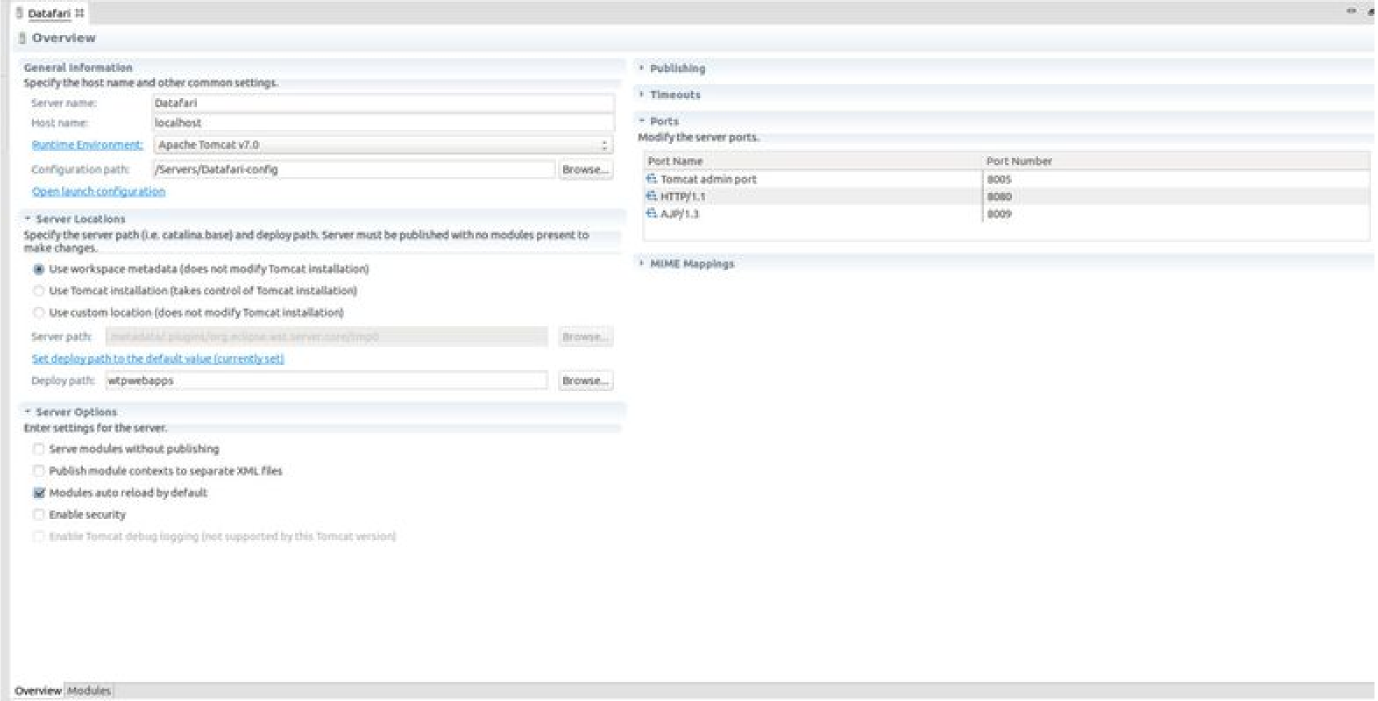
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Datafari

Server runtime environment: Apache Tomcat v7.0

Double click on Datafari to change its configuration :



Specify the host name and other common settings.

Server name: Datafari

Host name: localhost

Runtime Environment: Apache Tomcat v7.0

Configuration path: /Servers/Datafari-config

Open launch configuration

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

● Use workspace metadata (does not modify Tomcat installation)

○ Use Tomcat installation (takes control of Tomcat installation)

○ Use custom location (does not modify Tomcat installation)

Server path: .metadata/.plugins/org.eclipse.wst.server.core/tmp0

Set deploy path to the default value (currently set)

Deploy path: wtpwebapps

Enter settings for the server.

□ Serve modules without publishing

□ Publish module contexts to separate XML files

☑ Modules auto reload by default

□ Enable security

□ Enable Tomcat debug logging (not supported by this Tomcat version)

Modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	8080
AJP/1.3	8009

Click on « Use tomcat installation » and on « Open launch configuration ».

In argument, add this to VM arguments :

```
-Dsolr.solr.home="DATAFARI_SRC/solr/solr_home" -Dorg.apache.manifoldcf.configfile="DATAFARI_SRC/mcf/mcf_home/propertiesDev.xml"
```

And replace the parameter catalina.home by the path of your DATAFARI_SRC/tomcat :

-Dcatalina.home="DATAFARI_SRC/tomcat"

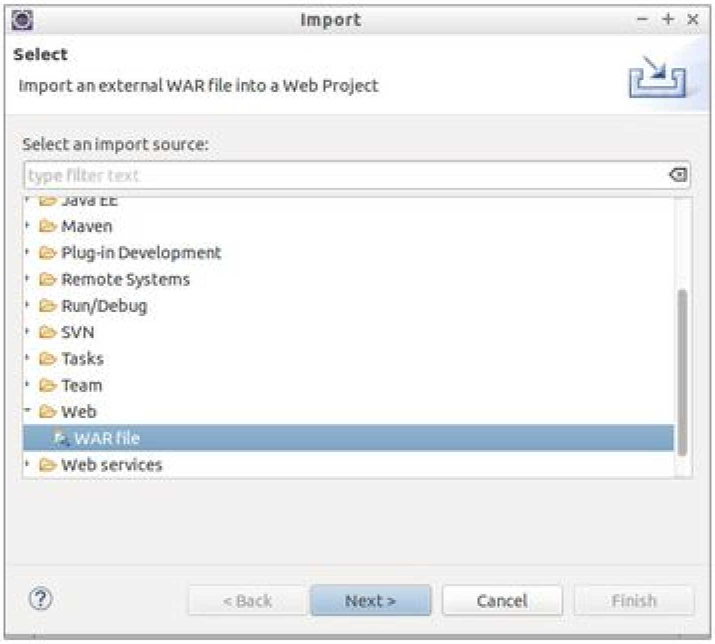
Finally, if you want to access to the manager page of Tomcat and the Banana page, you have to copy/paste the webapps into the webapps folder of your Eclipse workspace :

cp -R DATAFARI_SRC/tomcat/webapps/ROOT *your-eclipse-workspace/*.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/

cp -R DATAFARI_SRC/tomcat/webapps/manager *your-eclipse-workspace/*.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/

cp -R DATAFARI_SRC/tomcat/webapps/banana.war *your-eclipse-workspace/*.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/

In java EE perspective, select File Import and select Web War File :



Select

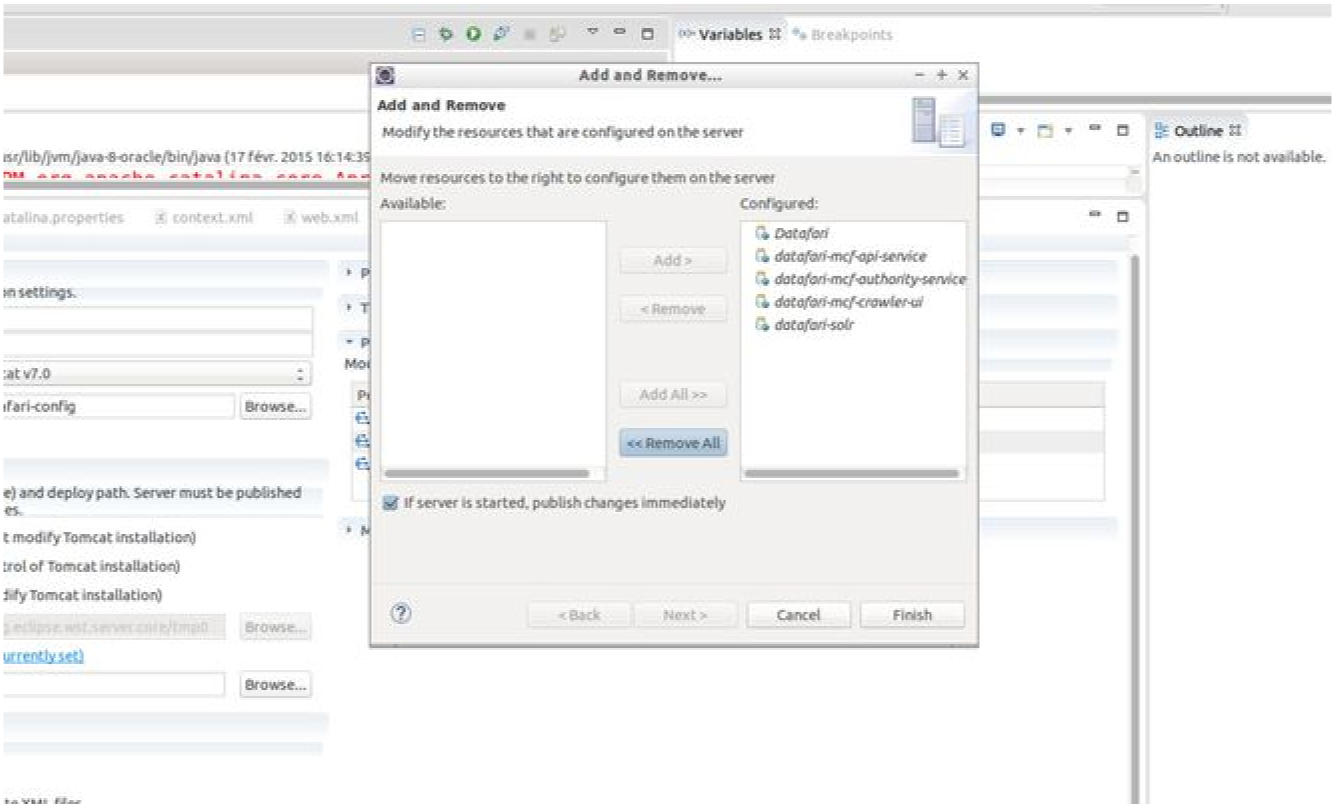
Import an external WAR file into a Web Project

Select an import source:

Browse to DATAFARI_SRC/solr/war/solr.war and set name of web project to datafari-solr.

Do the same for DATAFARI_SRC/mcf/war/mcf-api-service.war with the name datafari-mcf-api-service, for DATAFARI_SRC/mcf/war/mcf-crawler-ui.war (web project name : datafari-mcf-crawler-ui) and for DATAFARI_SRC/mcf/war/mcf-authority-service.war (web project name : datafari-mcf-authority-service).

Go back to the server window. Right click on Datafari, and do an Add and Remove… Add all the web app application (Datafari, datafari-solr, datafari-mcf-crawler-ui, datafari-mcf-api-service, datafari-mcf-authority-service).



Add and Remove

Modify the resources that are configured on the server

Move resources to the right to configure them on the server

Available:

Configured:

- Datafari
- datafari-mcf-api-service
- datafari-mcf-authority-service
- datafari-mcf-crawler-ui
- datafari-solr

☑ If server is started, publish changes immediately

Go to the « Servers » « Datafari Config » directory in the « Package explorer » of eclipse. Change the Datafari admin password in the tomcat-users.xml file. Copy DATAFARI_SRC/tomcat/conf/datafari.properties in the Datafari Config directory. You can now run the Datafari tomcat server

Run database and manifold agents

Open a terminal and set JAVA_HOME environement variable and navigate to DATAFARI_SRC/mcf/mcf_home.

Run bash start-database-dev.sh.

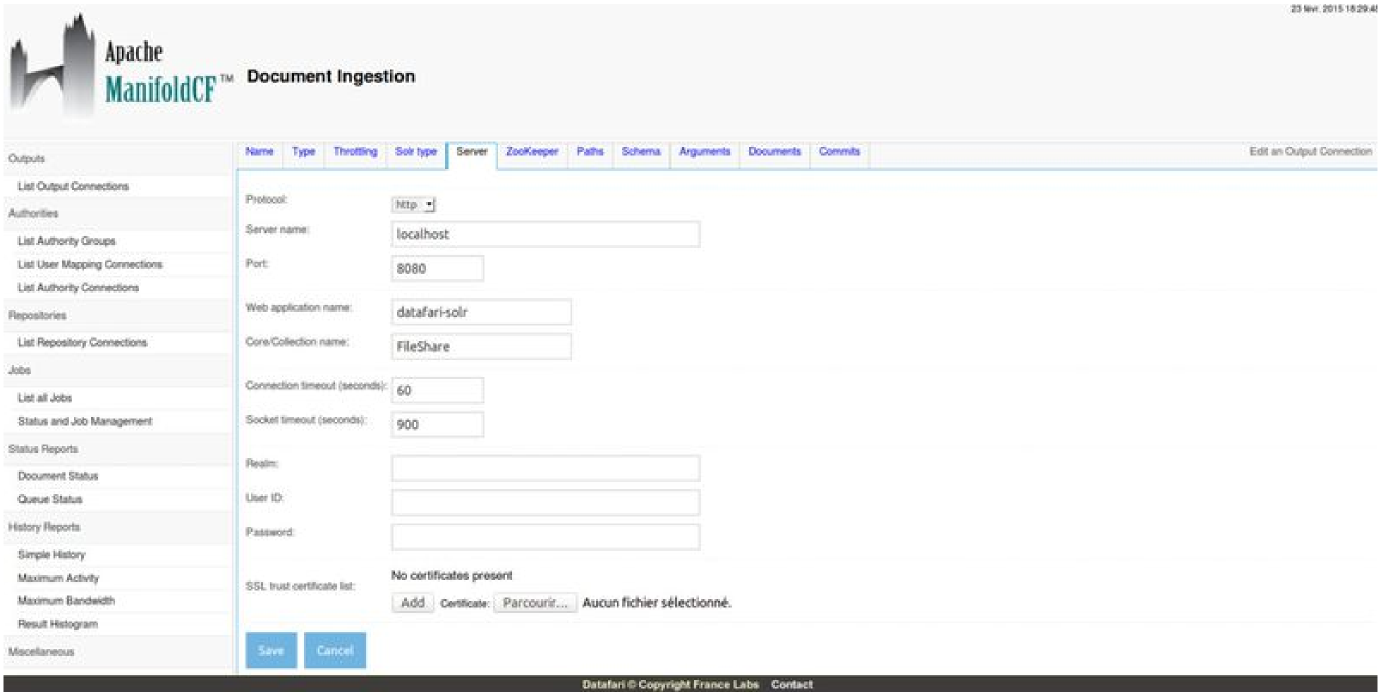
In another terminal, run bash initialize-dev.sh and then start-agent-dev.sh.

Keep the two terminals opened.

Configure connectors

Run Datafari tomcat. Go to the Datafari URL (http://localhost:8080/Datafari). Login in administrator. Go to Crawler tab-> Manifold CF Admin.

Log in and click on « List Output Connections » and create a new « Output Connection ». Set the name to « DatafariSolr ». Set Port to « 8080 », set web application name to « datafari-solr » and « Core/Collection name » to FileShare. You can now save:



Protocol: http

Server name: localhost

Port: 8080

Web application name: datafari-solr

Core/Collection name: FileShare

Connection timeout (seconds): 60

Socket timeout (seconds): 900

Realm:

User ID:

Password:

SSL trust certificate list: No certificates present

Add Certificate: Parcourir... Aucun fichier sélectionné.

Your development environnement is now ready.

Setup development environment with Eclipse (Linux) for Datafari 2.0

DEPRECATED

Deprecated : this page is for versions of Datafari older than 2.1, the one for 2.x is with Maven environment : Set up development environment with Eclipse (Linux) for Datafari 2.x - Maven

Pre-requisites

- Have a JDK installed on your PC (suggested version Java 7)
- Have an IDE installed on your PC (in the guide we use Eclipse) and be able to lunch it as root
- Have git installed on your PC

Clone repository

Open a terminal and navigate to the folder where you want to have Datafari source code to be checkouted (usually your workspace folder).

Perform a

```
git clone https://github.com/francelabs/datafari.git
```

to checkout the code. The root directory name is datafari.

Open the project in Eclipse

Navigate to the folder datafari. We now call it **DATAFARI_SRC**.

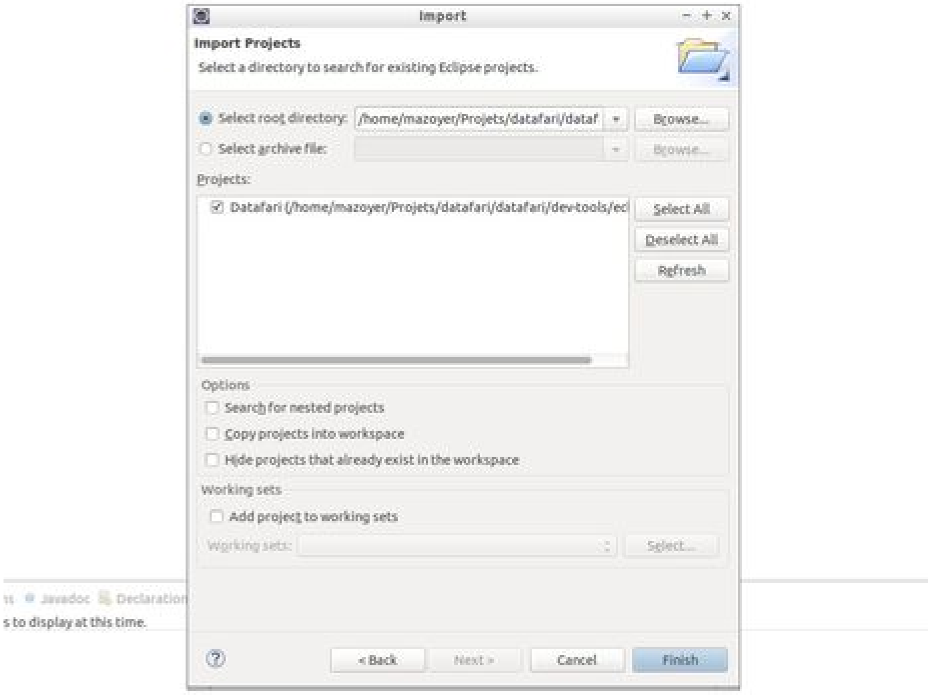
Run an : "ant eclipse" in DATAFARI_SRC.

ℹ If you get an error related to tools.jar not found, please check your JAVA_HOME environment variable.

⚠ Run Eclipse as root ⚠

In eclipse, go to File -> Import... , expand General and do an « Import existing projects into workspace ».

Select the DATAFARI_SRC folder as root directory and click Finish.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /home/mazoyer/Projets/datafari/dataf

○ Select archive file:

Projects:

☑ Datafari (/home/mazoyer/Projets/datafari/datafari/dev-tools/ecl

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

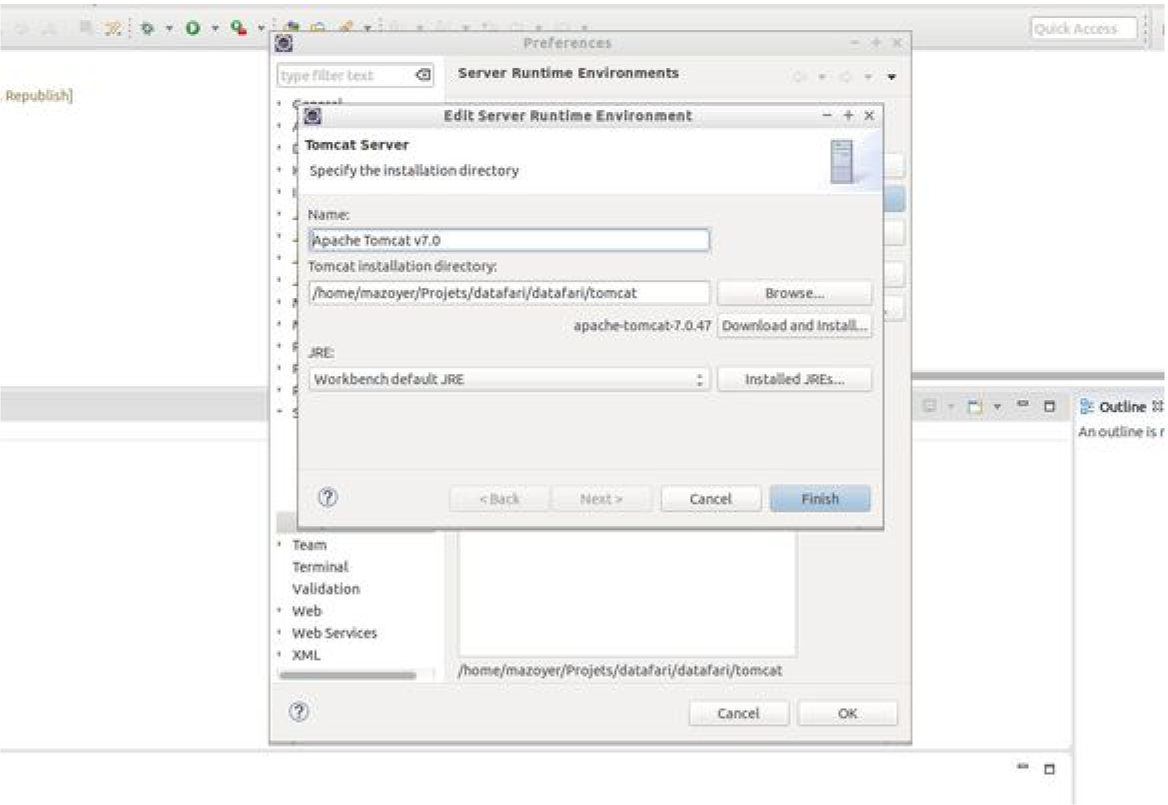
Working sets:

Setup the Tomcat server in Eclipse

Go to Window -> Preferences -> Server -> Runtime Environments -> Add…

and select Apache Tomcat V. 7.0

Select DATAFARI_SRC/tomcat for the Tomcat Installation directory and Finish.



Tomcat Server

Specify the installation directory

Name: Apache Tomcat v7.0

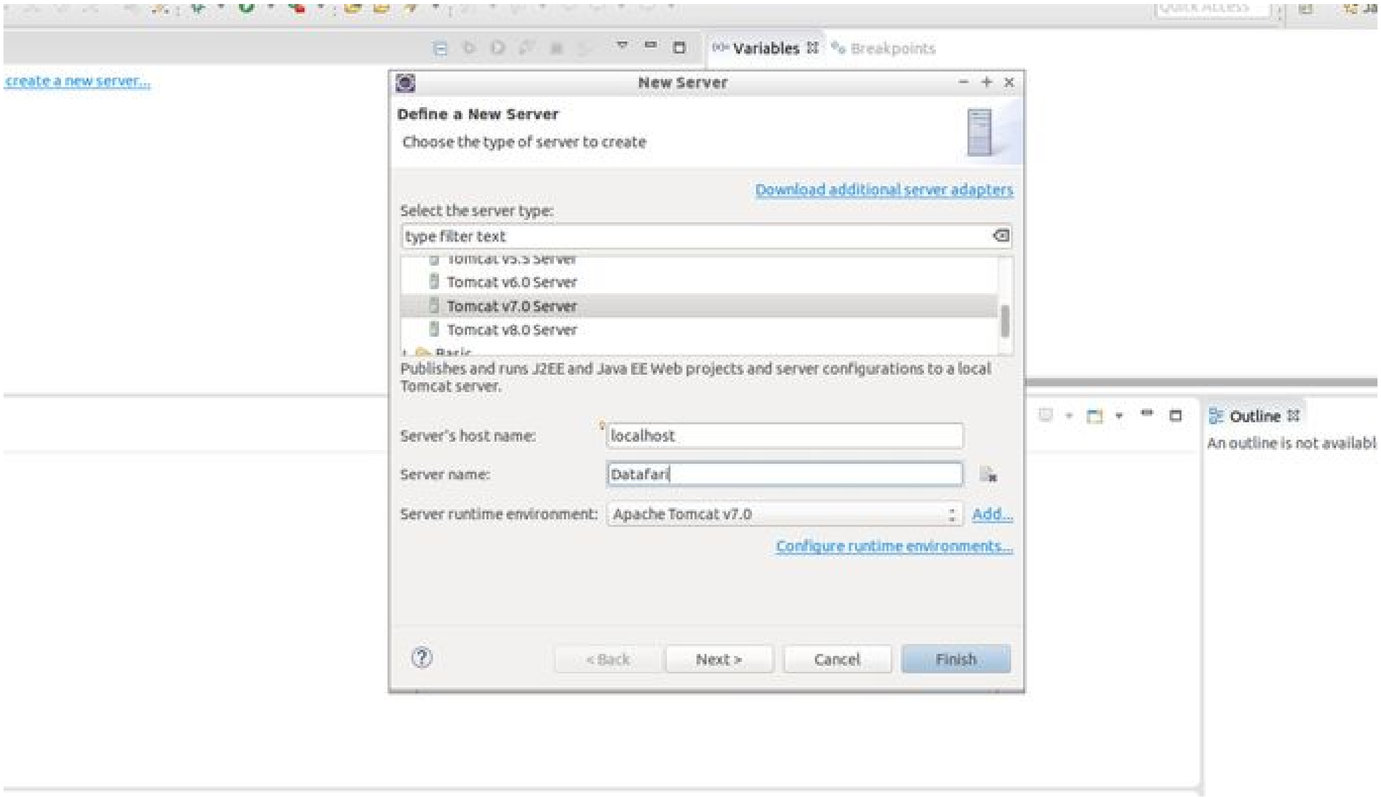
Tomcat installation directory: /home/mazoyer/Projets/datafari/datafari/tomcat

apache-tomcat-7.0.47

JRE: Workbench default JRE

/home/mazoyer/Projets/datafari/datafari/tomcat

In a Servers window (for example in Debug Perspectives), click on "add a server", select a "tomcat 7.0 server" and set the server name to "Datafari":



Define a New Server

Choose the type of server to create

Select the server type: type filter text

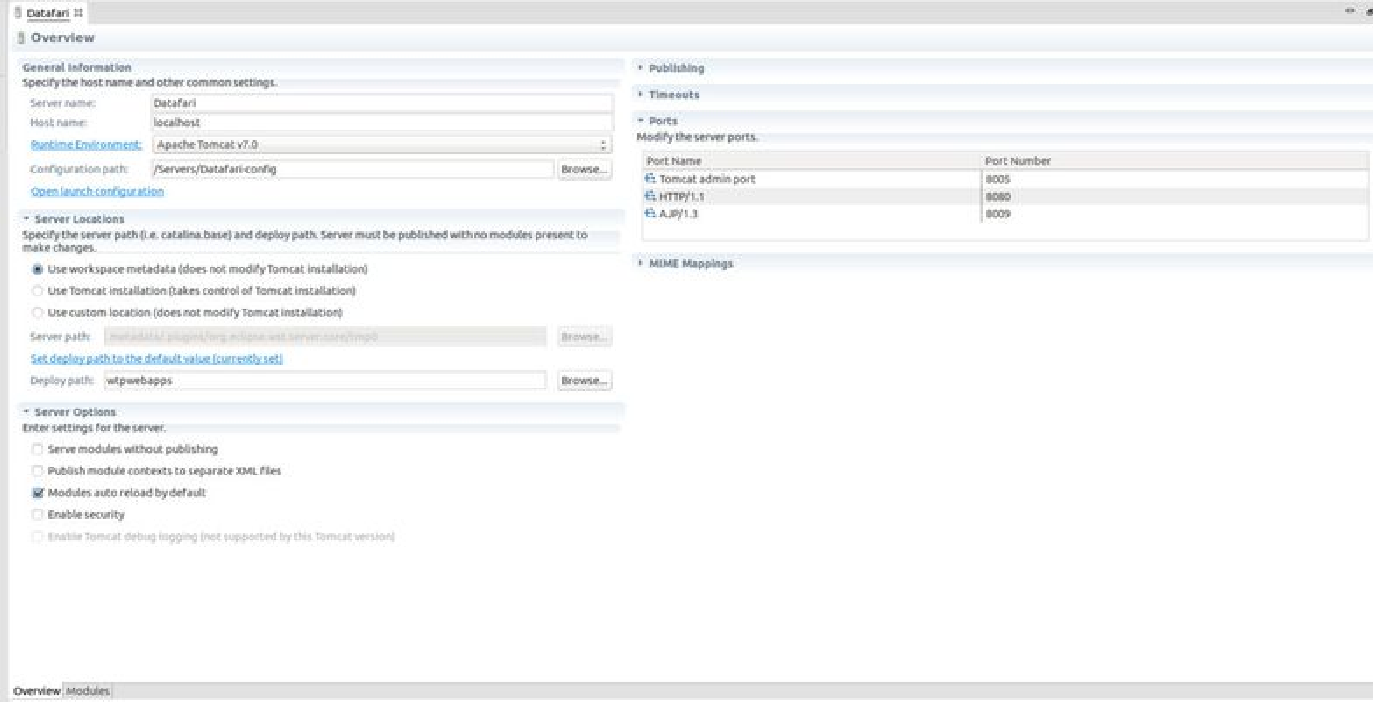
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Datafari

Server runtime environment: Apache Tomcat v7.0

Double click on Datafari to change its configuration :



General Information
Specify the host name and other common settings.

Server name: Datafari
Host name: localhost
Runtime Environment: Apache Tomcat v7.0
Configuration path: /Servers/Datafari-config Browse...
Open launch configuration

Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

● Use workspace metadata (does not modify Tomcat installation)
○ Use Tomcat installation (takes control of Tomcat installation)
○ Use custom location (does not modify Tomcat installation)

Server path: .metadata/.plugins/org.eclipse.wst.server.core/tmp0 Browse...
Set deploy path to the default value (currently set)
Deploy path: wtpwebapps Browse...

Server Options
Enter settings for the server.

□ Serve modules without publishing
□ Publish module contexts to separate XML files
☑ Modules auto reload by default
□ Enable security
□ Enable Tomcat debug logging (not supported by this Tomcat version)

Publishing

Timeouts

Ports
Modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	8080
AJP/1.3	8009

MIME Mappings

In Server Locations section: select « Use Tomcat installation ».

In General Information section, click on « Open launch configuration ».

In Arguments tab, add this to VM arguments :

```
-Dorg.apache.manifoldcf.configfile="DATAFARI_SRC/mcf/mcf_home/propertiesDev.xml"
-Dsolr.solr.home="DATAFARI_SRC/solr/solr_home/"
```

[Optional] And replace the parameter catalina.home by the path of your DATAFARI_SRC/tomcat :

```
-Dcatalina.home="DATAFARI_SRC/tomcat"
```

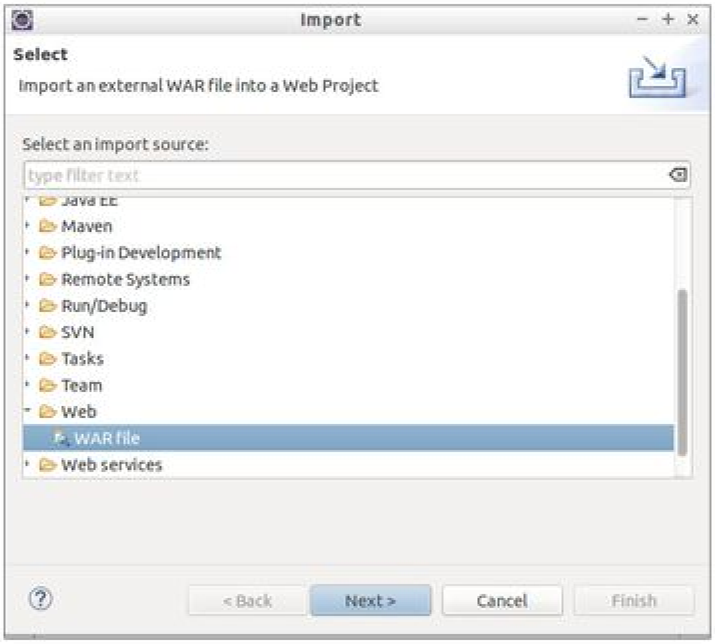
Finally, if you want to access to the manager page of Tomcat and the Banana page, you have to copy/paste the webapps into the webapps folder of your Eclipse workspace :

```
cp -R DATAFARI_SRC/tomcat/webapps/ROOT
your-eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/
```

```
cp -R DATAFARI_SRC/tomcat/webapps/manager
your-eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/
```

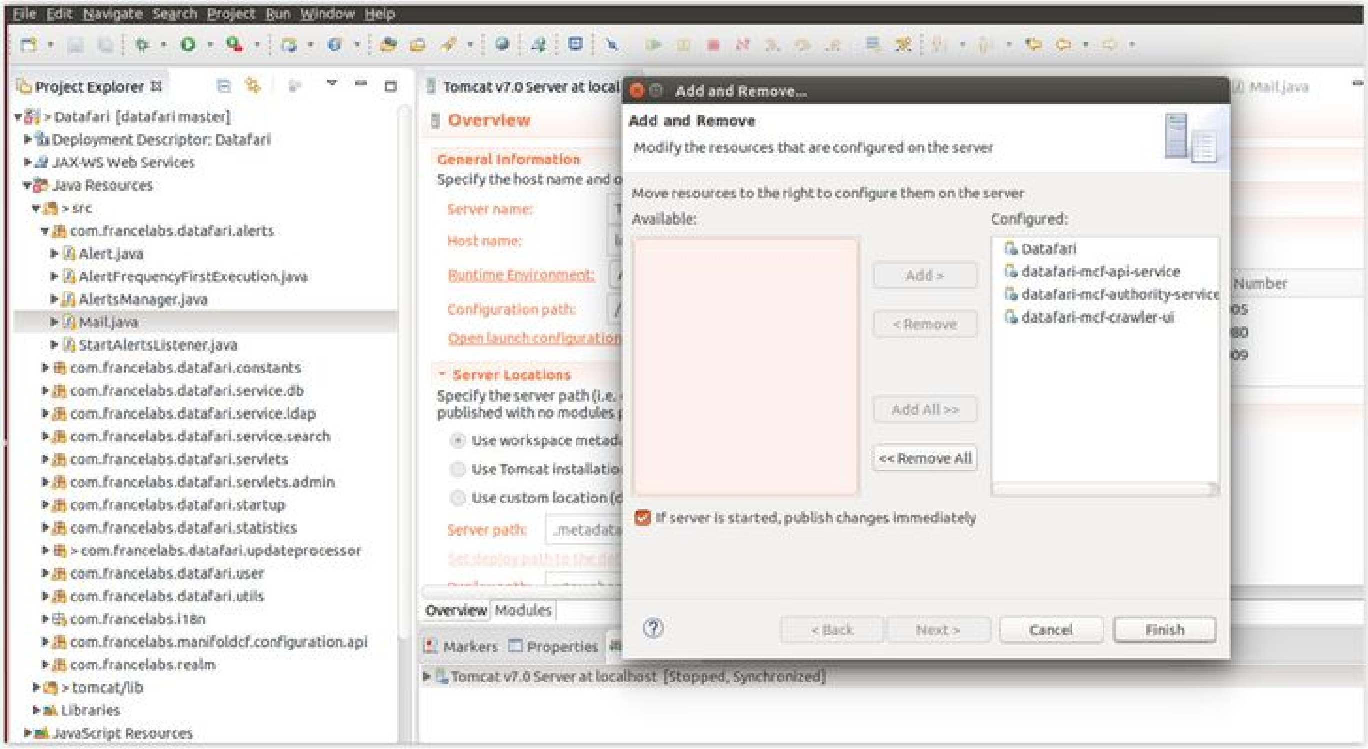
```
cp -R DATAFARI_SRC/tomcat/webapps/banana.war
your-eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/webapps/
```

In Java EE perspective, select File -> Import and select Web -> War File :



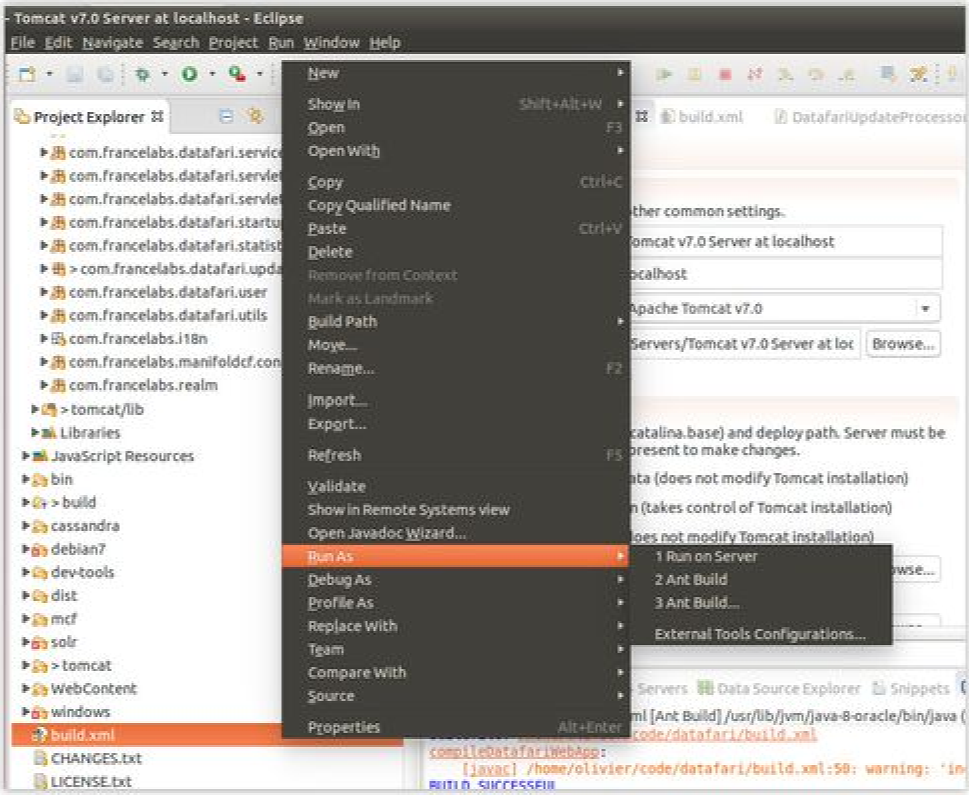
Browse to DATAFARI_SRC/mcf/war/mcf-api-service.war and set name of web project with the name datafari-mcf-api-service, for DATAFARI_SRC/mcf/war/mcf-crawler-ui.war (web project name : datafari-mcf-crawler-ui) and for DATAFARI_SRC/mcf/war/mcf-authority-service.war (web project name : datafari-mcf-authority-service).

Go back to the Servers window. Right click on Datafari, and do an Add and Remove… Add all the web app applications (Datafari, datafari-mcf-crawler-ui, datafari-mcf-api-service, datafari-mcf-authority-service).


File Edit Navigate Search Project Run Window Help
Project Explorer
> Datafari [datafari master]
Deployment Descriptor: Datafari
JAX-WS Web Services
Java Resources
> src
com.francelabs.datafari.alerts
Alert.java
AlertFrequencyFirstExecution.java
AlertsManager.java
Mail.java
StartAlertsListener.java
com.francelabs.datafari.constants
com.francelabs.datafari.service.db
com.francelabs.datafari.service.ldap
com.francelabs.datafari.service.search
com.francelabs.datafari.servlets
com.francelabs.datafari.servlets.admin
com.francelabs.datafari.startup
com.francelabs.datafari.statistics
> com.francelabs.datafari.updateprocessor
com.francelabs.datafari.user
com.francelabs.datafari.utils
com.francelabs.i18n
com.francelabs.manifoldcf.configuration.api
com.francelabs.realm
> tomcat/lib
Libraries
JavaScript Resources
Tomcat v7.0 Server at local
Overview
General Information
Specify the host name and o
Server name:
Host name:
Runtime Environment:
Configuration path:
Open launch configuration
Server Locations
Specify the server path (i.e. published with no modules
Use workspace metad
Use Tomcat installatio
Use custom location (d
Server path:
.metadata
Overview Modules
Markers Properties
Tomcat v7.0 Server at localhost [Stopped, Synchronized]
Mail.java
Number
Add and Remove...
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
Datafari
datafari-mcf-api-service
datafari-mcf-authority-service
datafari-mcf-crawler-ui
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Cancel
Finish

Open either Package explorer or the Navigator view: Go to Servers Datafari-config directory.
Change the Datafari admin password in the tomcat-users.xml file (look for @PASSWORD@).
Copy DATAFARI_SRC/tomcat/conf/datafari.properties in the Datafari-config directory.

Run Ant on DATAFARI_SRC/build.xml in order to build the required libraries (Right click on build.xml of datafari web project, then select Run As -> Ant build) :


Tomcat v7.0 Server at localhost - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
com.francelabs.datafari.servic
com.francelabs.datafari.servlet
com.francelabs.datafari.servlet
com.francelabs.datafari.startu
com.francelabs.datafari.statist
> com.francelabs.datafari.upda
com.francelabs.datafari.user
com.francelabs.datafari.utils
com.francelabs.i18n
com.francelabs.manifoldcf.con
com.francelabs.realm
> tomcat/lib
Libraries
JavaScript Resources
bin
> build
cassandra
debian7
dev-tools
dist
mcf
solr
> tomcat
WebContent
windows
build.xml
CHANGES.txt
LICENSE.txt
New
Show In Shift+Alt+W
Open F3
Open With
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete
Remove from Context
Mark as Landmark
Build Path
Move...
Rename... F2
Import...
Export...
Refresh F5
Validate
Show in Remote Systems view
Open Javadoc Wizard...
Run As
Debug As
Profile As
Replace With
Team
Compare With
Source
Properties Alt+Enter
1 Run on Server
2 Ant Build
3 Ant Build...
External Tools Configurations...
build.xml
DatafariUpdateProcessor
ther common settings.
omcat v7.0 Server at localhost
ocalhost
Apache Tomcat v7.0
Servers/Tomcat v7.0 Server at loc
Browse...
catalina.base) and deploy path. Server must be present to make changes.
ata (does not modify Tomcat installation)
n (takes control of Tomcat installation)
loes not modify Tomcat installation)
wse...
Servers
Data Source Explorer
Snippets

```
ml [Ant Build] /usr/lib/jvm/java-8-oracle/bin/java (
code/datafari/build.xml
compileDatafariWebApp:
    [javac] /home/olivier/code/datafari/build.xml:50: warning: 'in
BUILD SUCCESSFUL
```

Run Solr and Cassandra

Start Datafari: Open a terminal and navigate to DATAFARI_SRC/dev-tools/script/debian.

You need to be root to run the command.

```
bash start-datafari-devmode.sh
```

After that, you will need to initialize Cassandra. To do it, go to DATAFARI_SRC/dev-tools/script/debian.

You also need to be root to run the command.

```
bash configure-cassandra-dev.sh
```

The command ends without any feedback.

The Cassandra keyspace and the tables are created. A new user 'admin' is created too. The credentials to login to Datafari admin are : admin/admin

Check if the folder DATAFARI_SRC/pid is created and if the Solr and Cassandra PID files are present. If not create it manually and affect enough rights for the folder. For example :

```
mkdir DATAFARI_SRC/pid
```

```
sudo chmod -R 777 DATAFARI_SRC/pid
```

Run database and ManifoldCF agents

[Optional] Set JAVA_HOME environment variable pointing to your installation of Java.

Open a terminal and navigate to DATAFARI_SRC/mcf/mcf_home.

(you may need to change the permissions on the MCF folder if you are not root)

Run

```
bash start-database-dev.sh.
```

In another terminal, run (ONLY ONCE, the first time you set-up Datafari dev environment)

```
bash initialize-dev.sh
```

and then

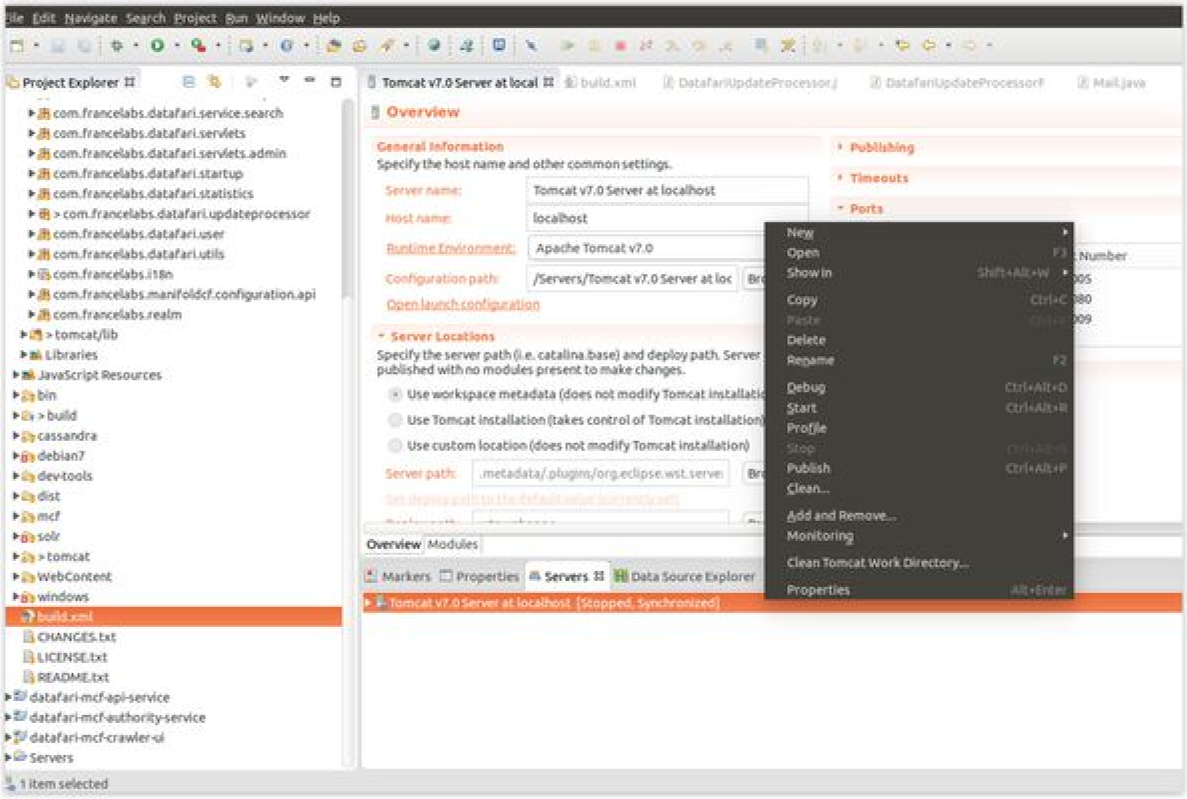
```
bash start-agents-dev.sh.
```

Keep the two terminals open.

Run Tomcat

Go to Eclipse and start the Tomcat server (from Servers view)

ℹ If you get FileNotFoundExceptions on the Datafari and/or Solr log files, please run Eclipse as root.


File Edit Navigate Search Project Run Window Help
Project Explorer
com.francelabs.datafari.service.search
com.francelabs.datafari.servlets
com.francelabs.datafari.servlets.admin
com.francelabs.datafari.startup
com.francelabs.datafari.statistics
> com.francelabs.datafari.updateprocessor
com.francelabs.datafari.user
com.francelabs.datafari.utils
com.francelabs.i18n
com.francelabs.manifoldcf.configuration.api
com.francelabs.realm
> tomcat/lib
Libraries
JavaScript Resources
bin
> build
cassandra
debian7
dev-tools
dist
mcf
solr
> tomcat
WebContent
windows
build.xml
CHANGES.txt
LICENSE.txt
README.txt
datafari-mcf-api-service
datafari-mcf-authority-service
datafari-mcf-crawler-ui
Servers
1 item selected
Tomcat v7.0 Server at local
build.xml
DatafariUpdateProcessor.j
DatafariUpdateProcessorF
Mail.java
Overview
General Information
Specify the host name and other common settings.
Server name: Tomcat v7.0 Server at localhost
Host name: localhost
Runtime Environment: Apache Tomcat v7.0
Configuration path: /Servers/Tomcat v7.0 Server at loc
Open launch configuration
Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server published with no modules present to make changes.
Use workspace metadata (does not modify Tomcat installation)
Use Tomcat installation (takes control of Tomcat installation)
Use custom location (does not modify Tomcat installation)
Server path: .metadata/.plugins/org.eclipse.wst.server
Publishing
Timeouts
Ports
Number
Overview Modules
Markers Properties Servers Data Source Explorer
Tomcat v7.0 Server at localhost [Stopped, Synchronized]
New
Open F3
Show In Shift+Alt+W
Copy Ctrl+C
Paste
Delete
Rename F2
Debug Ctrl+Alt+D
Start Ctrl+Alt+R
Profile
Stop
Publish Ctrl+Alt+P
Clean...
Add and Remove...
Monitoring
Clean Tomcat Work Directory...
Properties Alt+Enter

Go to the Datafari URL (http://localhost:8080/Datafari)

To login into Datafari, the credentials are : admin/admin

Configure connectors

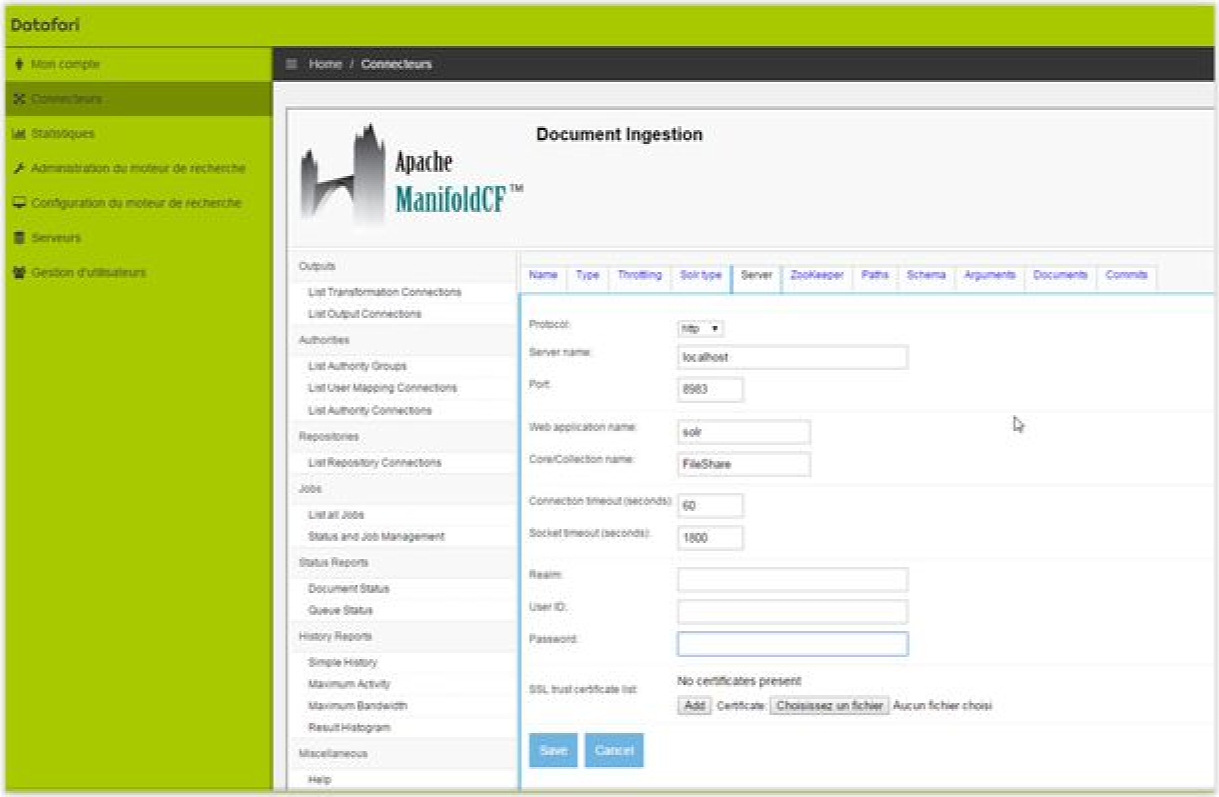
Run Datafari tomcat. Go to the Datafari URL (http://localhost:8080/Datafari). Login in administrator (admin/admin).

In the left menu, click on Connectors; the Apache ManifoldCF login page shows up. Enter admin/admin to login (click on Login button, as the Enter key may not work).

Click on « List Output Connections » of Output section, and add a new « Output Connection ».

In Name tab, set the name to « DatafariSolr ».
In Type tab, set the connection type to Solr and then click on Continue button.
In the Server tab, set port to « 8983 », set web application name to « solr » and « Core/Collection name » to FileShare.

You can now save:



Protocol: http
Server name: localhost
Port: 8983
Web application name: solr
Core/Collection name: FileShare
Connection timeout (seconds): 60
Socket timeout (seconds): 1800
Realm:
User ID:
Password:
SSL trust certificate list: No certificates present
Certificate: Aucun fichier choisi

Your development environment is now ready.

For more information about ManifoldCF configuration, please have a look here: Crawling

Setup development environment with Eclipse (Linux) for Datafari 2.x - Maven

Pre-requisites

- Have a JDK installed on your PC (suggested version Java 7)
- Have an IDE installed on your PC (in the guide we use Eclipse)
- Have git installed on your PC
- Have Maven installed on your PC (at least version 3.0.5)
- Have Ant installed on your PC
- Have an Internet connection available (No offline installation permitted)

Wondering what are the possible Datafari build flows ? Have a look here Distributions HOW-TO

Clone repository

Open a terminal and navigate to the folder where you want to have Datafari source code to be checkouted (usually your workspace folder).

Perform a

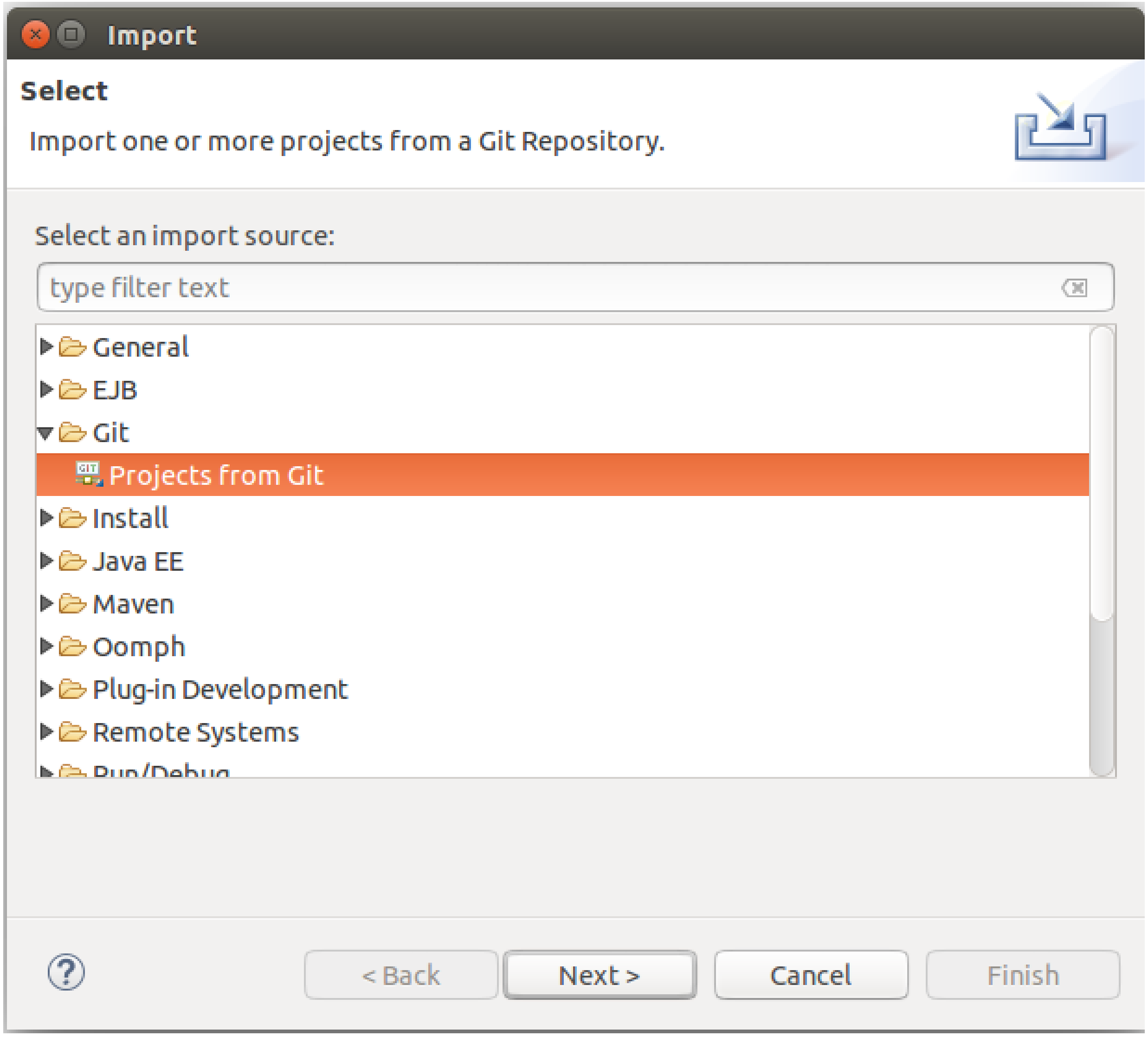
```
git clone https://github.com/francelabs/datafari.git -b maven datafari
```

to checkout the code. The root directory name must be "datafari": we now call it **DATAFARI_SRC**.

Open the project in Eclipse

Start Eclipse by calling the executable from the workspace folder (the one directly containing DATAFARI_SRC).

In eclipse, go to File -> Import... , expand Git and do a « Projects from Git ».



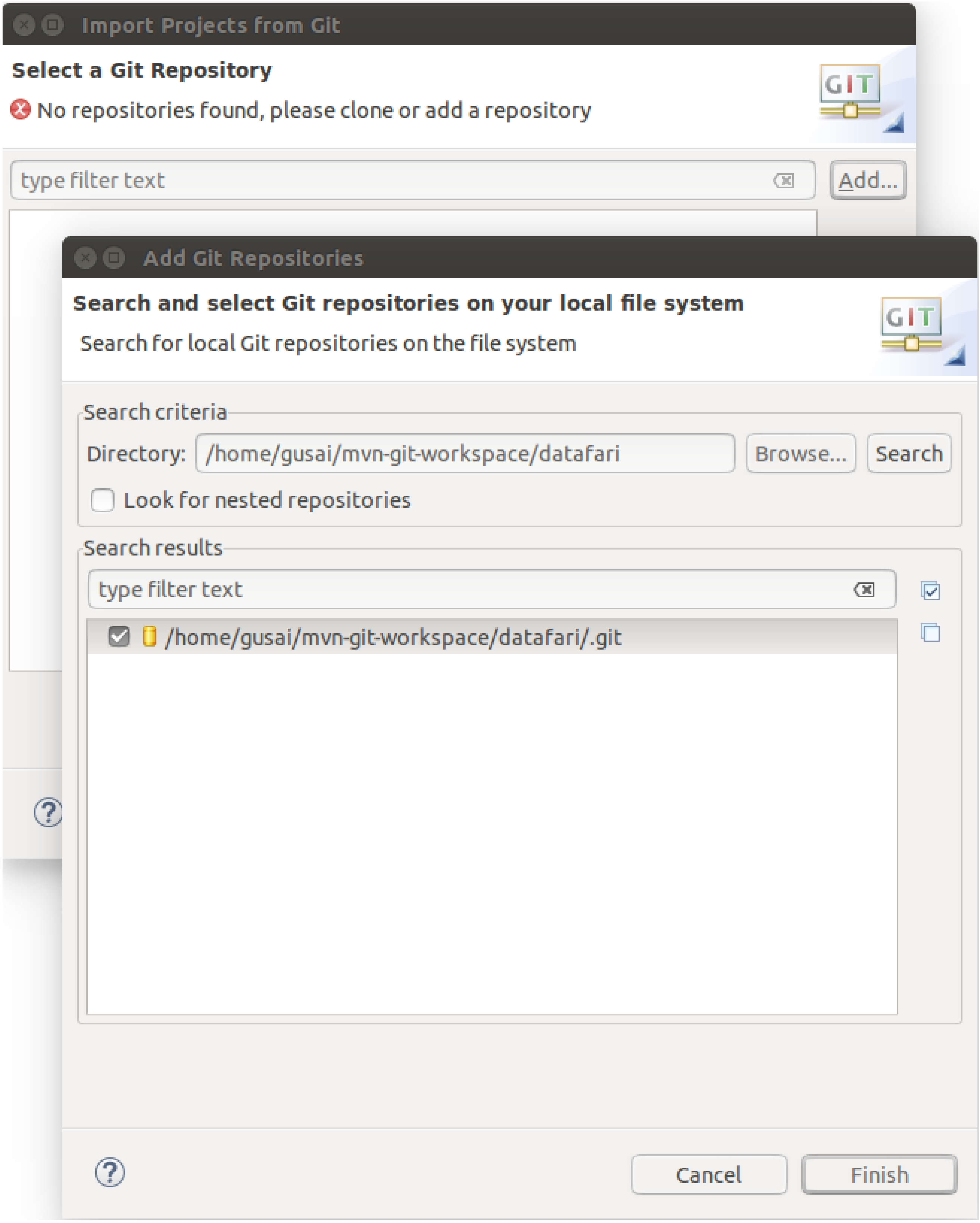
Import

Select

Import one or more projects from a Git Repository.

Select an import source:

Select "Existing local repository" and add the repo we have just cloned (by means of browse pop-up).



Select a Git Repository

No repositories found, please clone or add a repository

Search and select Git repositories on your local file system

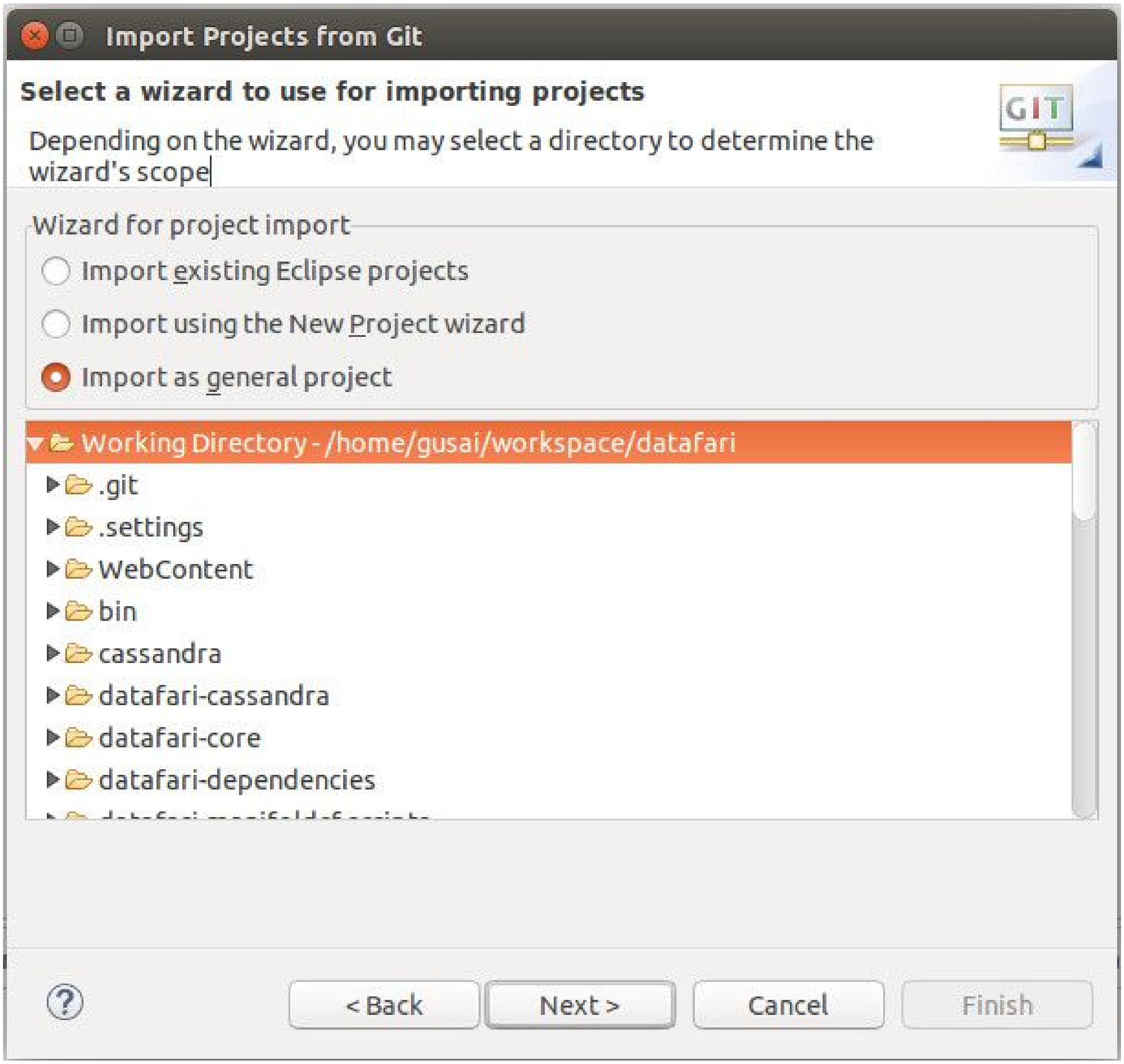
Search for local Git repositories on the file system

Directory: /home/gusai/mvn-git-workspace/datafari

☐ Look for nested repositories

☑ /home/gusai/mvn-git-workspace/datafari/.git

Use "Import as general project" as Wizard. (Don't worry, the classpath and Eclipse settings will be set-up by Ant script).


Import Projects from Git

Select a wizard to use for importing projects

Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

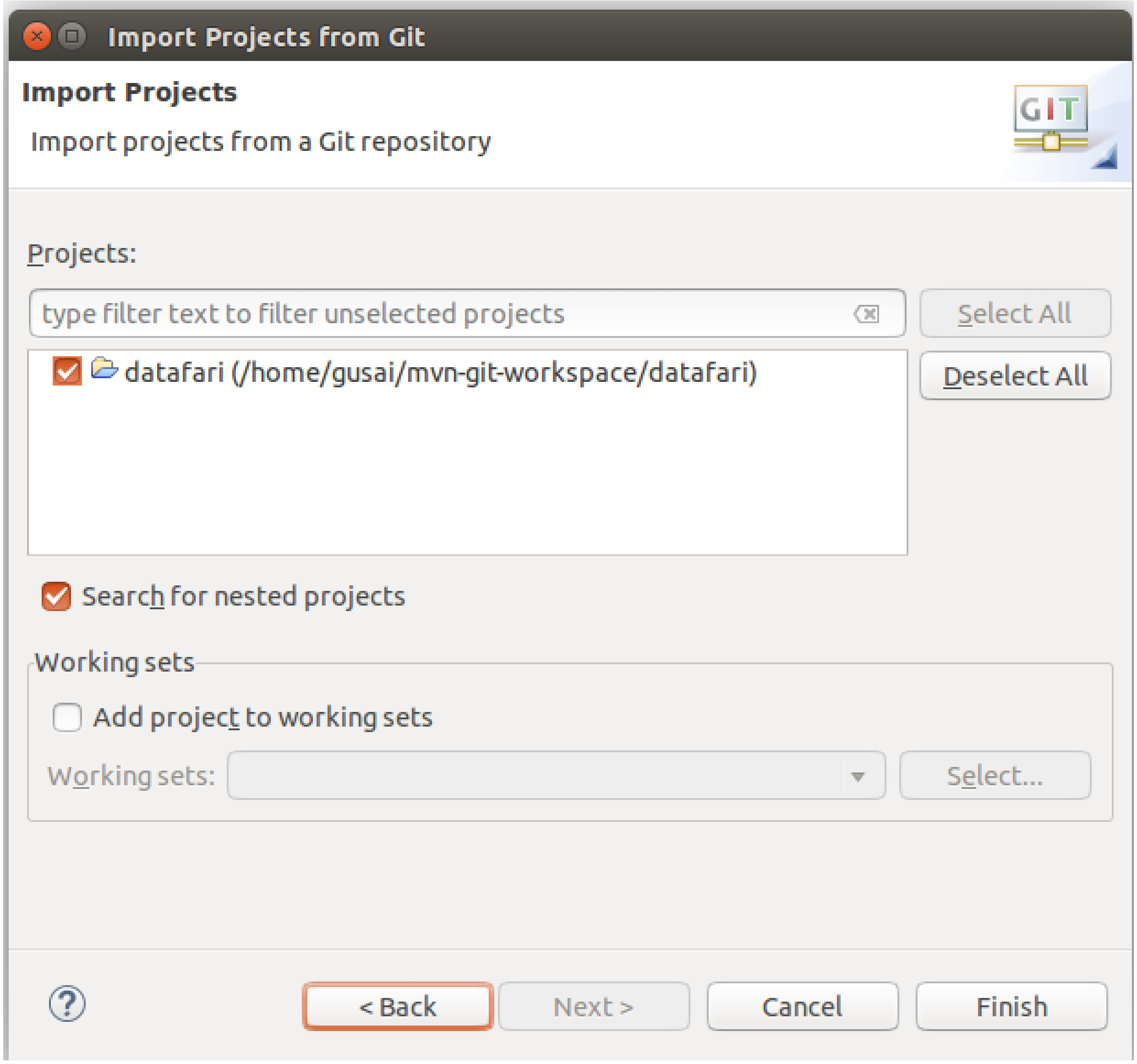
○ Import existing Eclipse projects

○ Import using the New Project wizard

● Import as general project

- Working Directory - /home/gusai/workspace/datafari
  - .git
  - .settings
  - WebContent
  - bin
  - cassandra
  - datafari-cassandra
  - datafari-core
  - datafari-dependencies

Select the DATAFARI_SRC folder as root directory and click Finish.



Import Projects from Git

Import Projects

Import projects from a Git repository

Projects:

type filter text to filter unselected projects

☑ datafari (/home/gusai/mvn-git-workspace/datafari)

☑ Search for nested projects

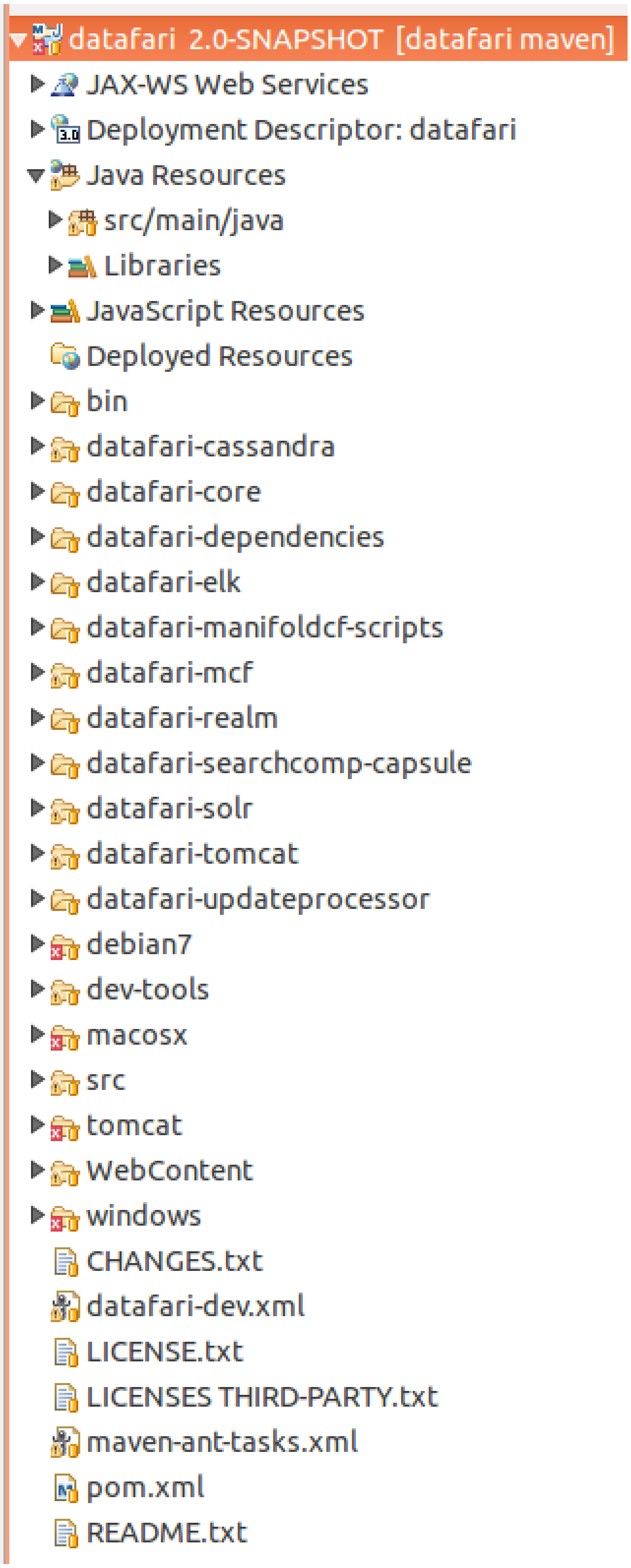
Working sets

□ Add project to working sets

Working sets:

Leave Eclipse build the project automatically.

You should get this output, in project explorer view:



```
datafari  2.0-SNAPSHOT  [datafari maven]
  JAX-WS Web Services
  Deployment Descriptor: datafari
  Java Resources
    src/main/java
    Libraries
  JavaScript Resources
  Deployed Resources
  bin
  datafari-cassandra
  datafari-core
  datafari-dependencies
  datafari-elk
  datafari-manifoldcf-scripts
  datafari-mcf
  datafari-realm
  datafari-searchcomp-capsule
  datafari-solr
  datafari-tomcat
  datafari-updateprocessor
  debian7
  dev-tools
  macosx
  src
  tomcat
  WebContent
  windows
  CHANGES.txt
  datafari-dev.xml
  LICENSE.txt
  LICENSES THIRD-PARTY.txt
  maven-ant-tasks.xml
  pom.xml
  README.txt
```

As you can see, there are some "red crosses". This is not a problem as they are coming from external packages like ManifoldCF and Solr UI.

Eclipse configuration

In order to properly configure the classpath and the facets of the project, navigate to DATAFARI_SRC folder and run the following script:

```
ant -f ./dev-tools/setup-devenv/eclipse/setup-eclipse.xml
```

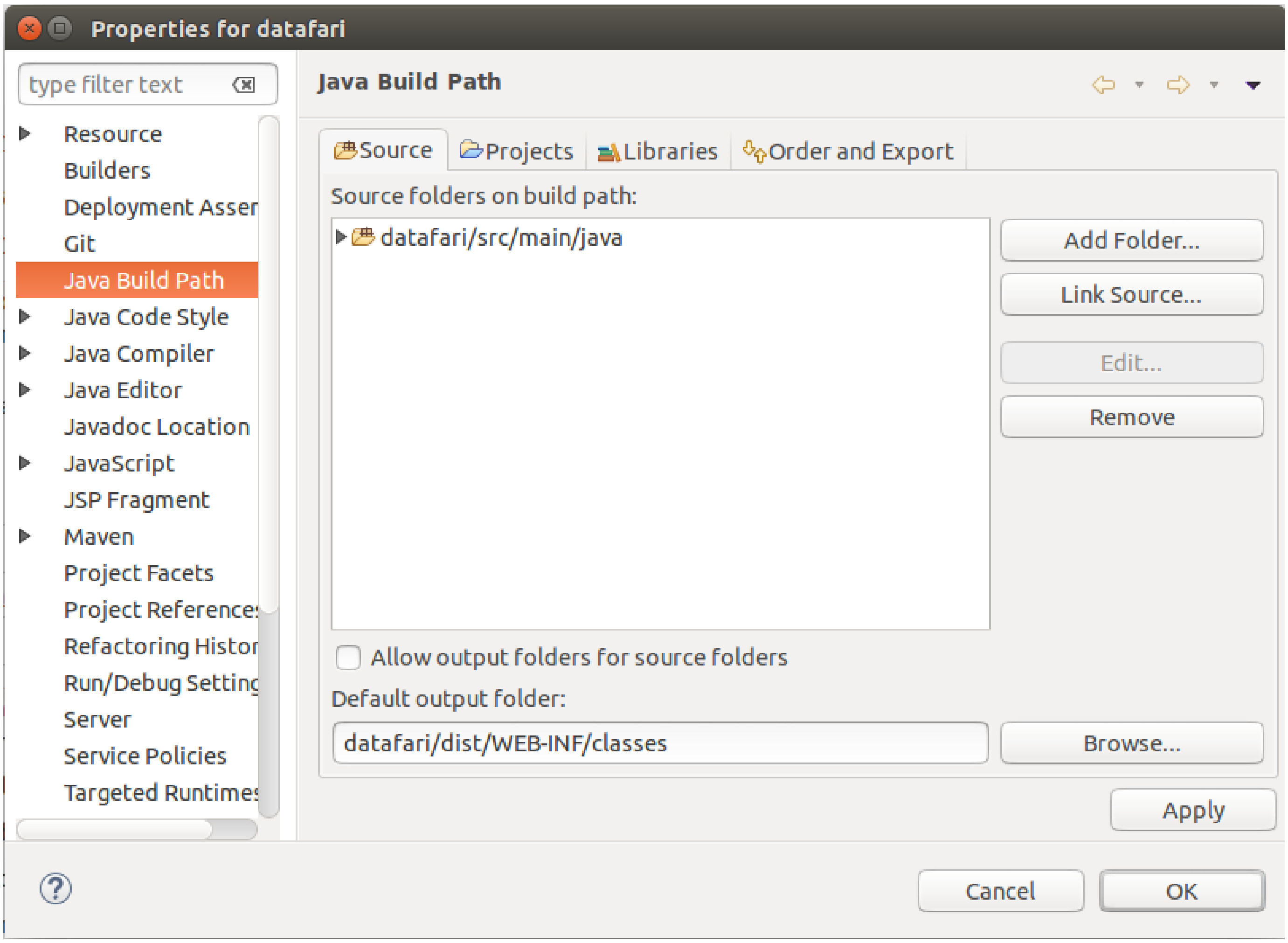
Check the Eclipse config

The Datafari Git repo contains some Eclipse config as well.

It's better to check that it has been correctly set-up =P:

Right click on datafari project -> Properties

Java Build path should be the following:



Properties for datafari

Java Build Path

Source folders on build path:

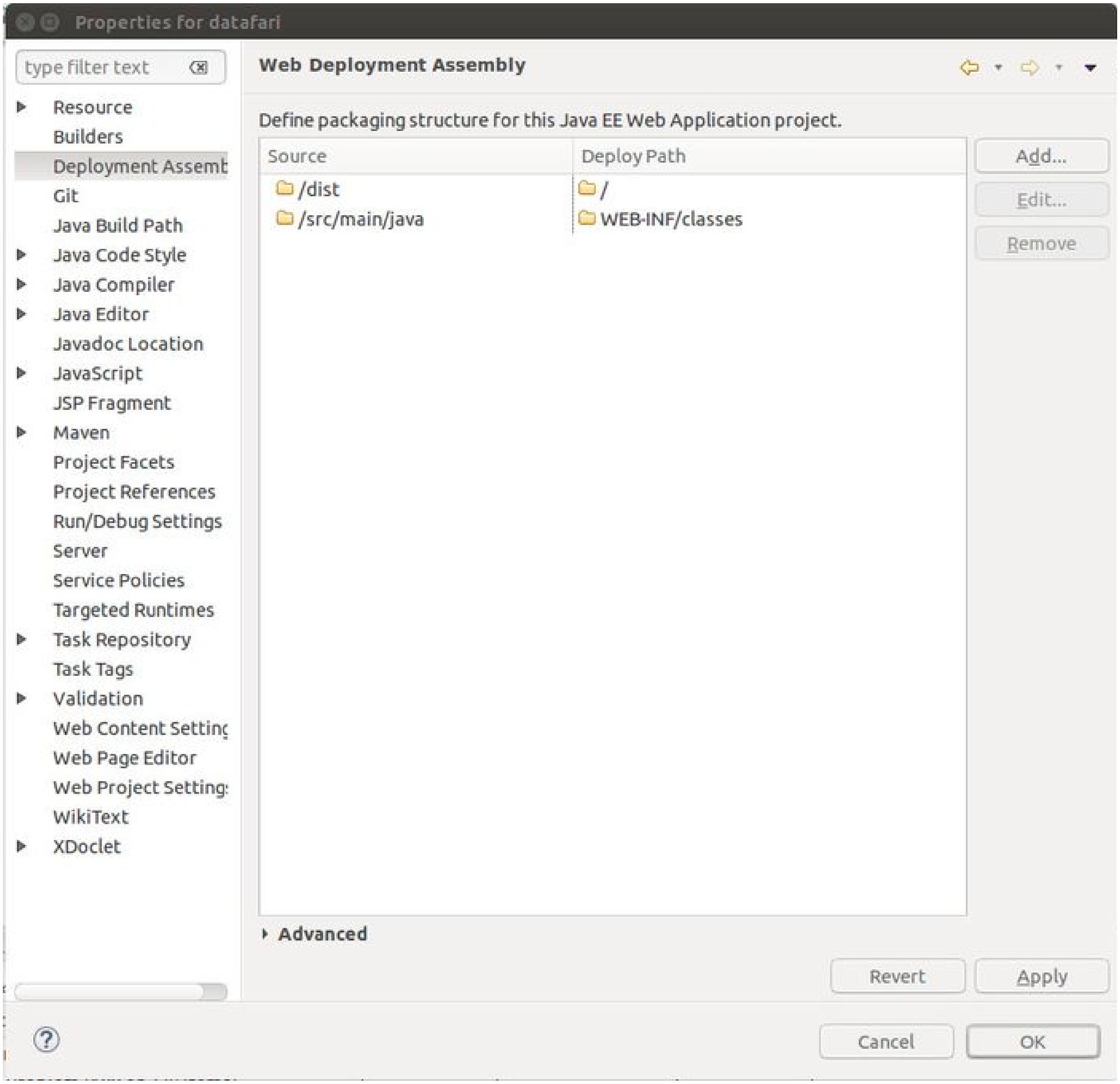
datafari/src/main/java

□ Allow output folders for source folders

Default output folder:

datafari/dist/WEB-INF/classes

Deployment assembly:

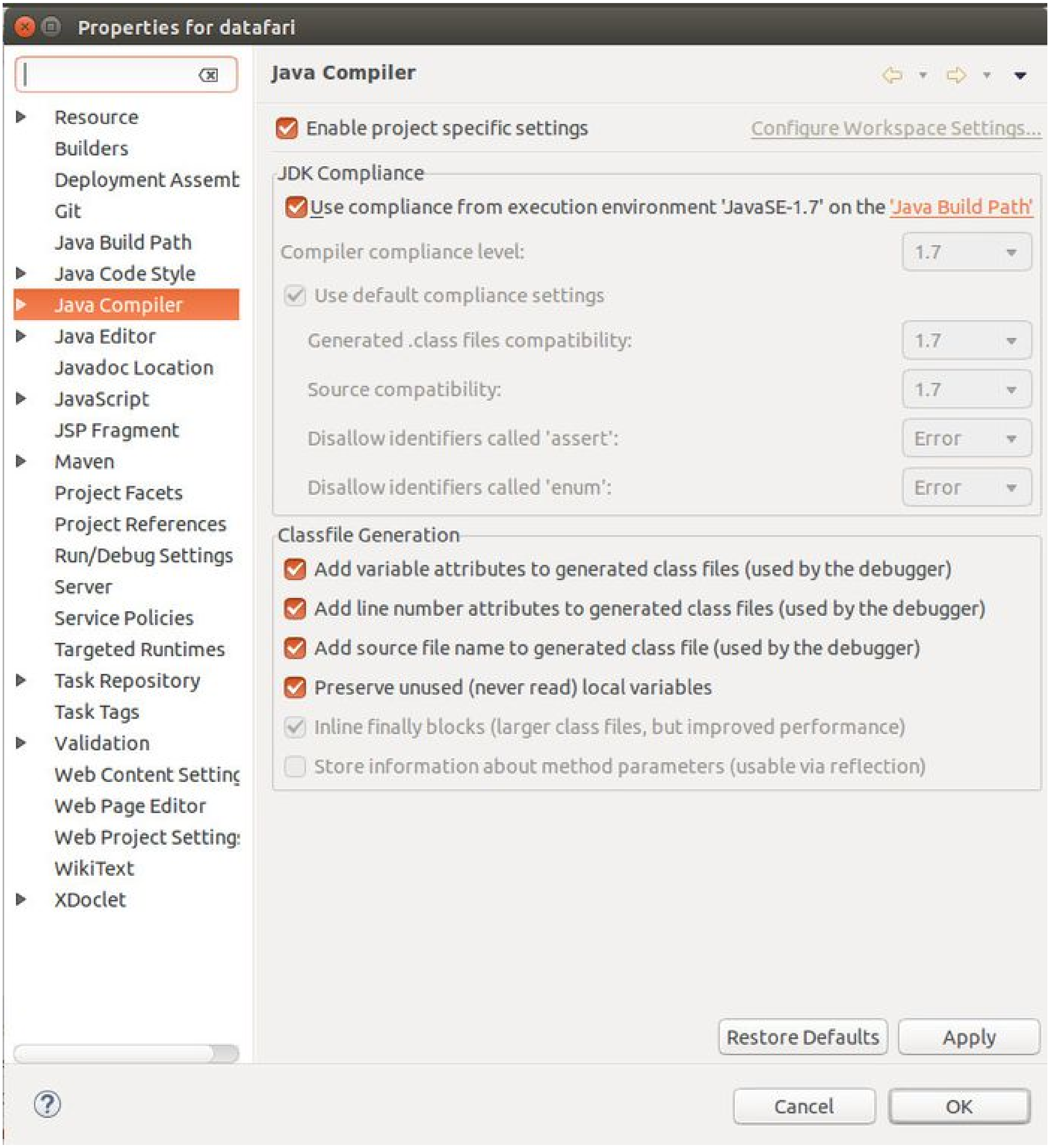


Web Deployment Assembly

Define packaging structure for this Java EE Web Application project.

Source	Deploy Path
/dist	/
/src/main/java	WEB-INF/classes

Java compiler should be as follows (Java 7):



Java Compiler

☑ Enable project specific settings   Configure Workspace Settings...

JDK Compliance

☑ Use compliance from execution environment 'JavaSE-1.7' on the 'Java Build Path'

Compiler compliance level: 1.7

☑ Use default compliance settings

Generated .class files compatibility: 1.7

Source compatibility: 1.7

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

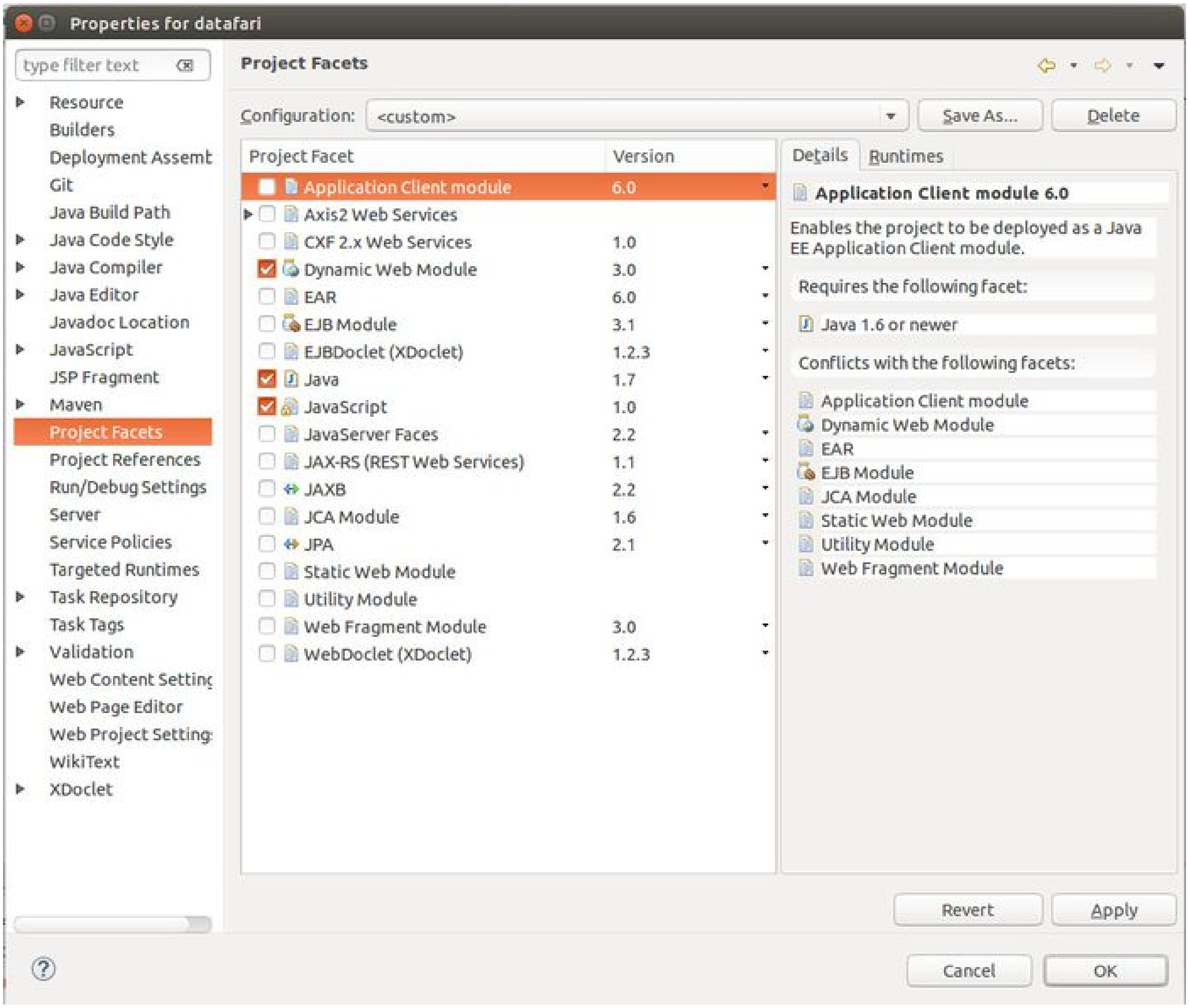
□ Store information about method parameters (usable via reflection)

Project facets:

Dynamic Web Module V 3.0

Java V 1.7

JavaScript V 1.0



Project Facets

Configuration: <custom>

Project Facet	Version
☐ Application Client module	6.0
☐ Axis2 Web Services	
☐ CXF 2.x Web Services	1.0
☑ Dynamic Web Module	3.0
☐ EAR	6.0
☐ EJB Module	3.1
☐ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.7
☑ JavaScript	1.0
☐ JavaServer Faces	2.2
☐ JAX-RS (REST Web Services)	1.1
☐ JAXB	2.2
☐ JCA Module	1.6
☐ JPA	2.1
☐ Static Web Module	
☐ Utility Module	
☐ Web Fragment Module	3.0
☐ WebDoclet (XDoclet)	1.2.3

Application Client module 6.0

Enables the project to be deployed as a Java EE Application Client module.

Requires the following facet:

Java 1.6 or newer

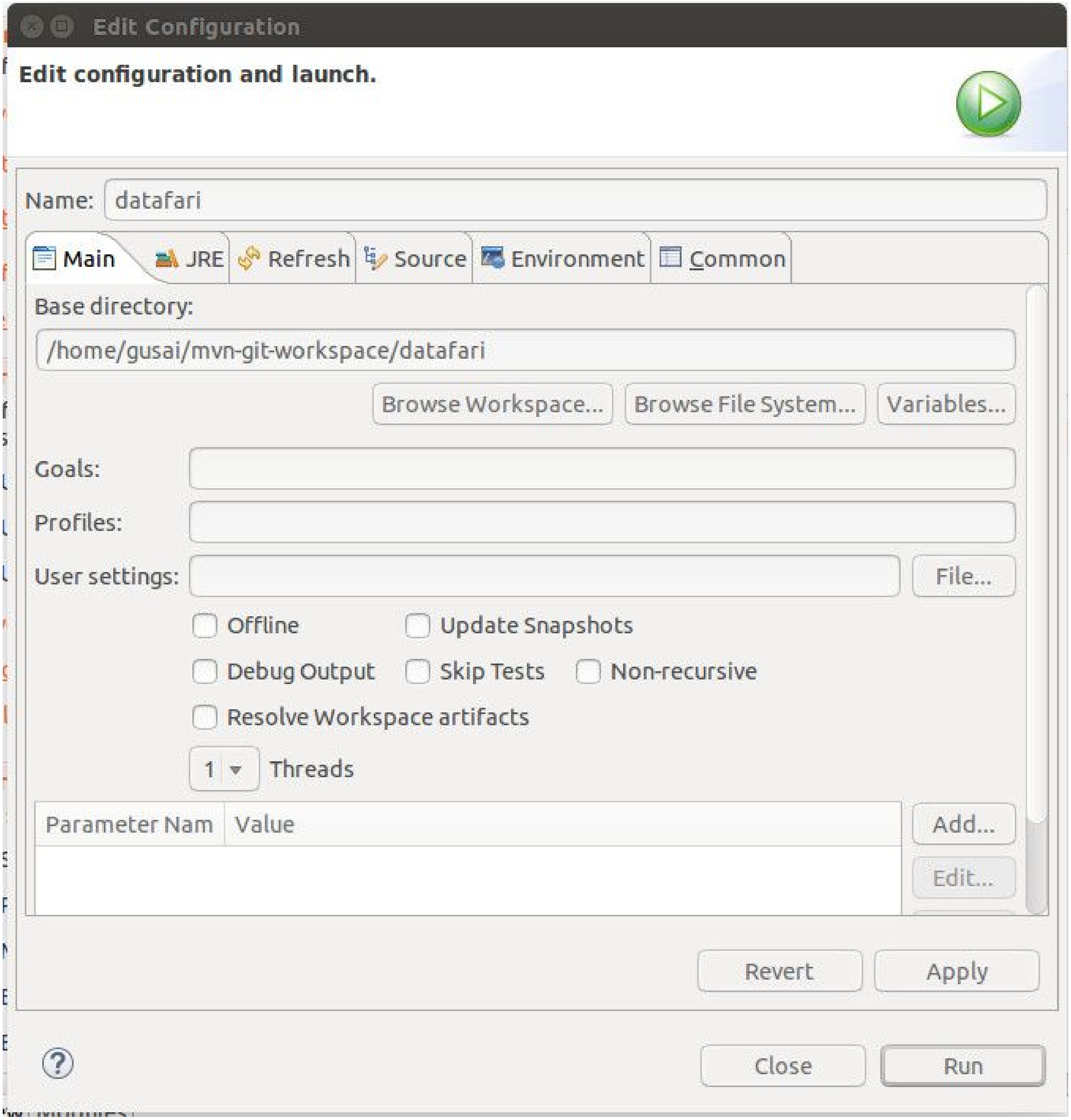
Conflicts with the following facets:

Application Client module
Dynamic Web Module
EAR
EJB Module
JCA Module
Static Web Module
Utility Module
Web Fragment Module

Run Maven build

Run Maven on DATAFARI_SRC/pom.xml in order to build the required libraries (Right click on pom.xml of datafari web project, then select Run As -> Maven build) :

Leave empty the goals section, the default one will be run.



Edit configuration and launch.

Name: datafari

Base directory: /home/gusai/mvn-git-workspace/datafari

Goals:

Profiles:

User settings:

□ Offline □ Update Snapshots

□ Debug Output □ Skip Tests □ Non-recursive

□ Resolve Workspace artifacts

1 Threads

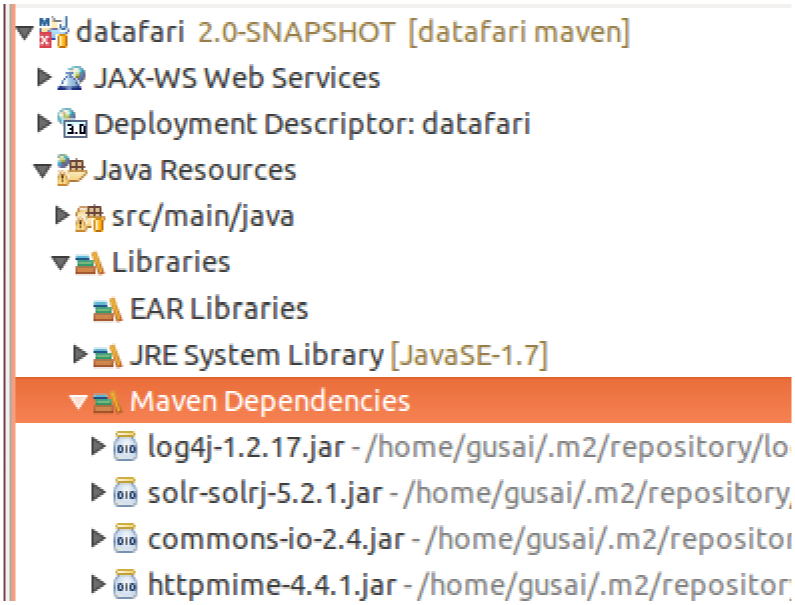
A console shows up with the output of Maven compilation.

At the end of the build process (it may take some time due to the download of external dependencies), you should see this output:

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Datafari Enterprise Search - Dependencies POM ...... SUCCESS [ 0.441 s]
[INFO] Datafari Enterprise Search - Parent POM ............ SUCCESS [ 32.289 s]
[INFO] Datafari Enterprise Search - Core module ........... SUCCESS [ 3.689 s]
[INFO] Datafari Enterprise Search - UpdateProcessor module SUCCESS [ 0.467 s]
[INFO] Datafari Enterprise Search - Capsule Search Component module SUCCESS [ 0.342 s]
[INFO] Datafari Enterprise Search - Realm module .......... SUCCESS [ 0.355 s]
[INFO] Datafari Enterprise Search - ManifoldCF Backup/Restore Scripts module SUCCESS [ 4.674 s]
[INFO] ------------------------------------------------------------------------
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 42.438 s
[INFO] Finished at: 2016-01-28T12:34:38+01:00
[INFO] Final Memory: 51M/700M
[INFO] ------------------------------------------------------------------------
```

Now check to have the some jars listed in Maven dependecies. They should point to your Maven's repo directory, eg user_home/.m2/repository.



```
▼ datafari 2.0-SNAPSHOT [datafari maven]
  ▶ JAX-WS Web Services
  ▶ Deployment Descriptor: datafari
  ▼ Java Resources
    ▶ src/main/java
    ▼ Libraries
        EAR Libraries
      ▶ JRE System Library [JavaSE-1.7]
      ▼ Maven Dependencies
        ▶ log4j-1.2.17.jar - /home/gusai/.m2/repository/lo
        ▶ solr-solrj-5.2.1.jar - /home/gusai/.m2/repository
        ▶ commons-io-2.4.jar - /home/gusai/.m2/reposito
        ▶ httpmime-4.4.1.jar - /home/gusai/.m2/repositor
```

If you see nothing in there or you have red crosses on some classes under Java Resources / src/main/java, right click on Datafari project, then select Maven entry and Update project.
Eclipse is not refreshing dependencies automatically.
The dependencies should show up under Maven Dependencies (if it's not the case, refresh the project).

Setup Tomcat server in Eclipse

ℹ Please note that we don't rely on Eclipse WTP environment, so the publish, clean, etc ... targets of server view are discouraged.
The only target you should run is the "start" of Tomcat.

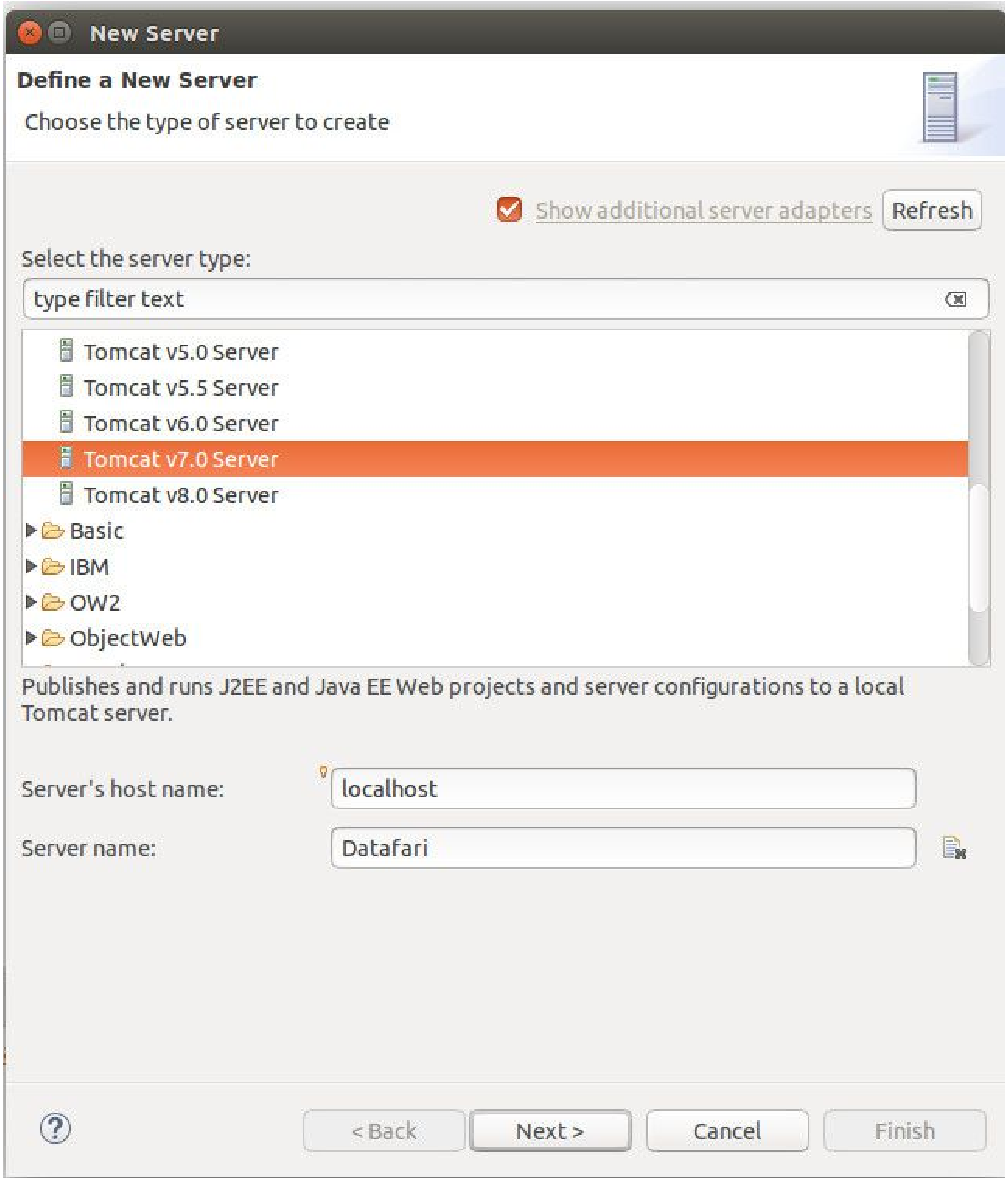
Add Server view: Window -> Show view -> Servers

and click on the link to create a new server.

Select Apache Tomcat V. 7.0 and put:

host name: localhost

server name: Datafari



Define a New Server

Choose the type of server to create

☑ Show additional server adapters

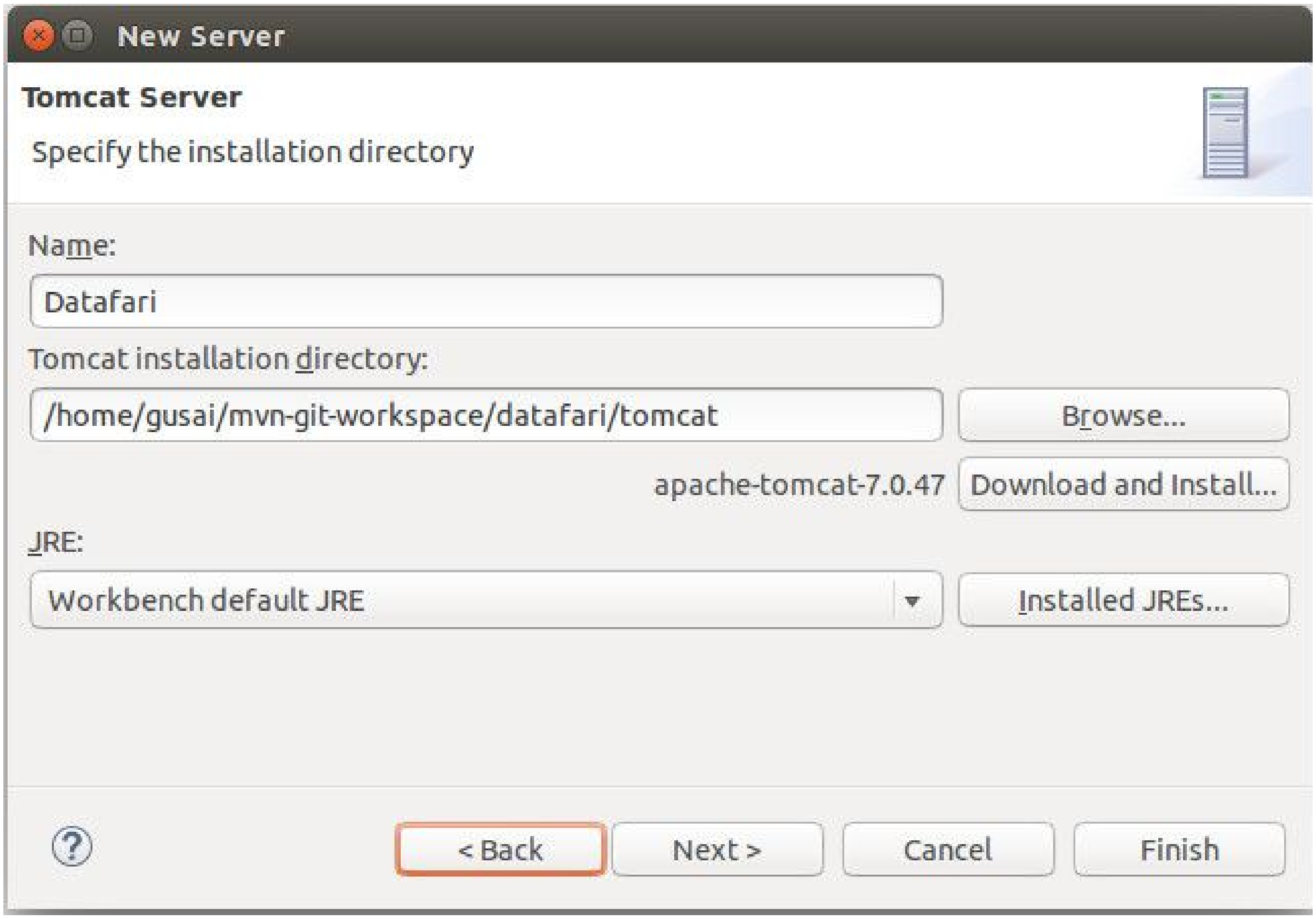
Select the server type: Tomcat v7.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Datafari

Select DATAFARI_SRC/tomcat for the Tomcat Installation directory and then next



New Server

Tomcat Server

Specify the installation directory

Name: Datafari

Tomcat installation directory: /home/gusai/mvn-git-workspace/datafari/tomcat

apache-tomcat-7.0.47

JRE: Workbench default JRE

DO NOT add Datafari web application to the server and then Finish.

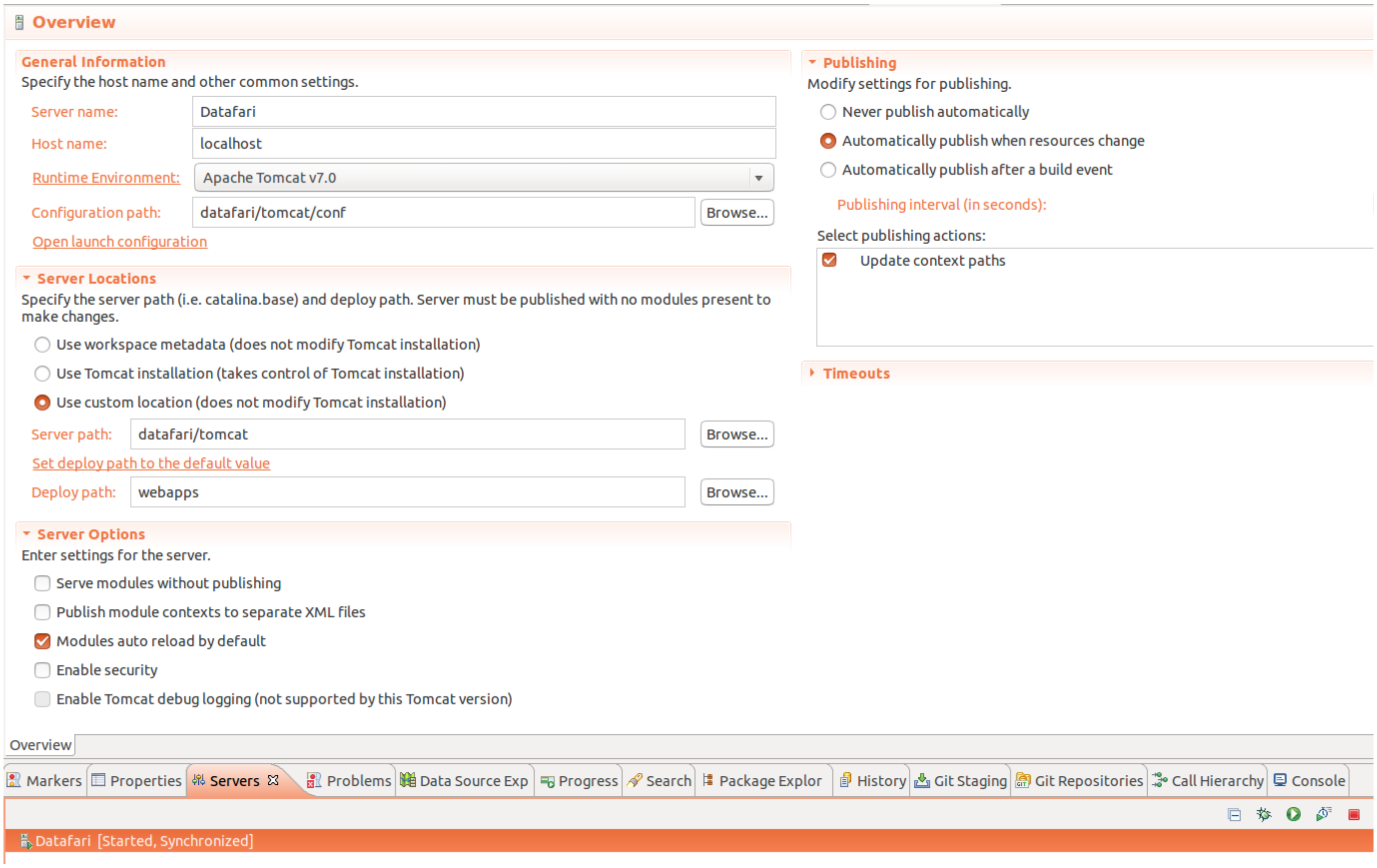
If Datafari-config project pops-out, you are safe to delete it (generated by Eclipse WTP).

Double click on Datafari server to change its configuration :

In Server Locations section, select "Use custom location" (to use the Tomcat embedded in Datafari).

Check that all the other configs are the same as in screenshot. You shouldn't see the Modules tab on the right of overview, or it should be empty (no deployed apps).

Notably, be careful that the paths are exactly as follows: datafari/tomcat/config and datafari/tomcat (no leading /)



Overview

General Information

Specify the host name and other common settings.

Server name: Datafari

Host name: localhost

Runtime Environment: Apache Tomcat v7.0

Configuration path: datafari/tomcat/conf

Open launch configuration

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

○ Use workspace metadata (does not modify Tomcat installation)

○ Use Tomcat installation (takes control of Tomcat installation)

● Use custom location (does not modify Tomcat installation)

Server path: datafari/tomcat

Set deploy path to the default value

Deploy path: webapps

Server Options

Enter settings for the server.

□ Serve modules without publishing

□ Publish module contexts to separate XML files

☑ Modules auto reload by default

□ Enable security

□ Enable Tomcat debug logging (not supported by this Tomcat version)

Publishing

Modify settings for publishing.

○ Never publish automatically

● Automatically publish when resources change

○ Automatically publish after a build event

Publishing interval (in seconds):

Select publishing actions:

☑ Update context paths

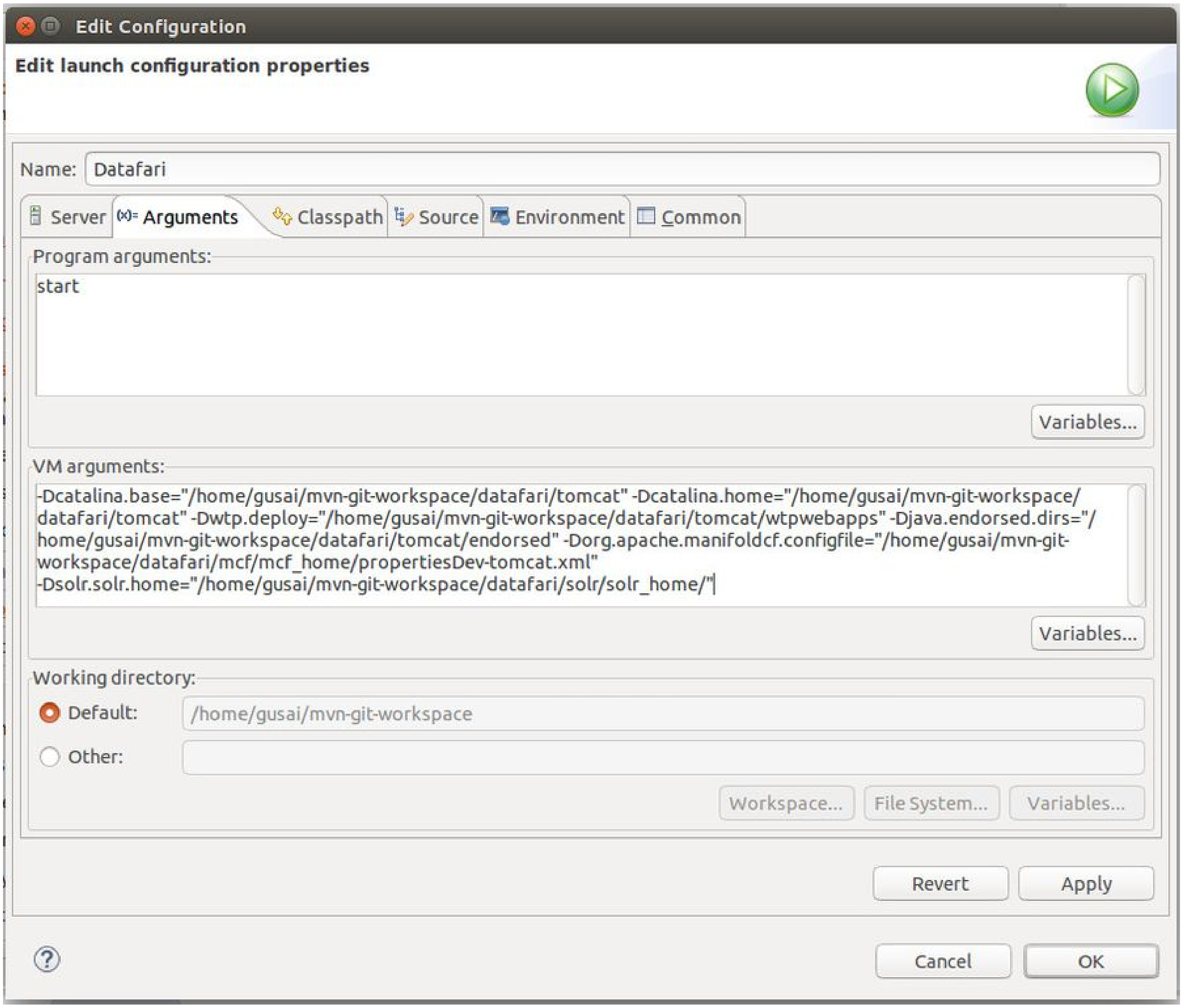
Timeouts

Datafari [Started, Synchronized]

In General Information section, click on « Open launch configuration ».

In Arguments tab, add this to VM arguments :

```
-Dorg.apache.manifoldcf.configfile="DATAFARI_SRC/mcf/mcf_home/propertiesDev-tomcat.xml"
-Dsolr.solr.home="DATAFARI_SRC/solr/solr_home/"
```



Name: Datafari

Program arguments:

```
start
```

VM arguments:

```
-Dcatalina.base="/home/gusai/mvn-git-workspace/datafari/tomcat" -Dcatalina.home="/home/gusai/mvn-git-workspace/datafari/tomcat" -Dwtp.deploy="/home/gusai/mvn-git-workspace/datafari/tomcat/wtpwebapps" -Djava.endorsed.dirs="/home/gusai/mvn-git-workspace/datafari/tomcat/endorsed" -Dorg.apache.manifoldcf.configfile="/home/gusai/mvn-git-workspace/datafari/mcf/mcf_home/propertiesDev-tomcat.xml"
-Dsolr.solr.home="/home/gusai/mvn-git-workspace/datafari/solr/solr_home/"
```

Working directory:

● Default: /home/gusai/mvn-git-workspace

○ Other:

and then OK.

Modules tab should be empty or not visible at all.

Run ZK, Solr and Cassandra

Start Datafari: Open a terminal and navigate to DATAFARI_SRC/dev-tools/script/debian.

```
bash start-datafari-devmode.sh
```

After that, you will need to initialize Cassandra. To do it, go to DATAFARI_SRC/dev-tools/script/debian.

```
bash configure-cassandra-dev.sh
```

The command ends without any feedback.

The Cassandra keyspace and the tables are created. A new user 'admin' is created too.
The credentials to login to Datafari admin are : admin/admin

Finally we have to initialize Solr : send the Solr configuration to Zookeeper and create the Solr collections. Launch the script configure-zk-solr-dev.sh in DATAFARI_SRC/dev-tools/script/debian

```
bash configure-zk-solr-dev.sh
```

Run database and ManifoldCF agents

Open a terminal and navigate to DATAFARI_SRC/mcf/mcf_home.

Run

```
bash start-database-dev.sh.
```

In another terminal, run (ONLY ONCE, the first time you set-up Datafari dev environment)

```
bash initialize-dev.sh
```

and then

```
bash start-agents-dev.sh.
```

Keep the two terminals open.

Run Tomcat

Go to Eclipse -> Servers view.

Start it (right click on the server name -> start).
Datafari application is already deployed by Maven/Ant.

ℹ Please note that we don't rely on Eclipse WTP environment, so the publish, clean, etc ... targets of server view are discouraged. The only target you should run are the "start / stop" of Tomcat.

Note: if you get this exception

INFO: Exception performing authentication. Retrying...
javax.naming.CommunicationException: 52.16.74.128:389 [Root exception is java.net.SocketTimeoutException: connect timed out]

it's the, let's say "normal" behaviour =P! (Fix on-going)

Go to the Datafari URL (http://localhost:8080/Datafari)

To login into Datafari, the credentials are : admin/admin

HAPPY SEARCHING 😁 !!!

💡 Need to get Datafari ready for production? Have a look here Distributions HOW-TO

Datafari incremental build and redeployment

For redeployment of Datafari Java sources and web resources ONLY (e.g. during an on-going dev), you can use the handy Ant targets "all" or "redeploy" of datafari-dev.xml.
This script recompiles the Datafari core sources and deploys the application on Tomcat.
ℹ If you get strange errors while running Ant, double check to run it on a JDK and not on a JRE (Oracle one, for instance).

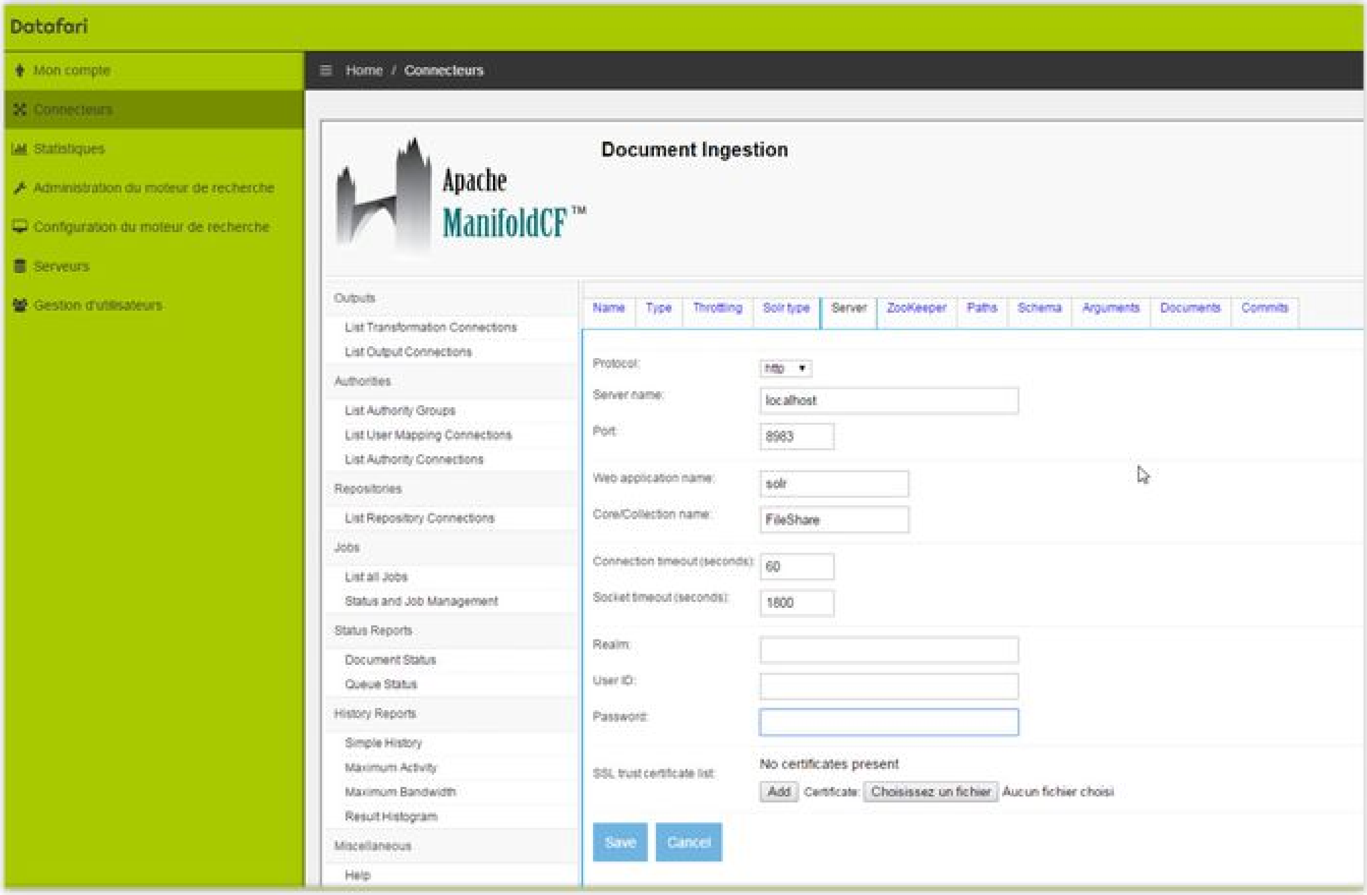
Configure connectors

Run Datafari tomcat. Go to the Datafari URL (http://localhost:8080/Datafari). Login in administrator (admin/admin).

In the left menu, click on Connectors; the Apache ManifoldCF login page shows up. Enter admin/admin to login (click on Login button, as the Enter key may not work).

Click on « List Output Connections » of Output section, and add a new « Output Connection ».
In Name tab, set the name to « DatafariSolr ».
In Type tab, set the connection type to Solr and then click on Continue button.
In the Server tab, set port to « 8983 », set web application name to « solr » and « Core/Collection name » to FileShare.

You can now save:



Protocol: http

Server name: localhost

Port: 8983

Web application name: solr

Core/Collection name: FileShare

Connection timeout (seconds): 60

Socket timeout (seconds): 1800

Realm:

User ID:

Password:

SSL trust certificate list: No certificates present

Add Certificate: Choisissez un fichier Aucun fichier choisi

Your development environment is now ready.

For more information about ManifoldCF configuration, please have a look here: Crawling

Setup development environment with Eclipse (Linux) for Datafari 3.1

Pre-requisites

- Have a JDK installed on your PC (suggested version Java 7)
- Have an IDE installed on your PC (in the guide we use Eclipse) and be able to lunch it as root
- Have git installed on your PC

Clone repository

Open a terminal and navigate to the folder where you want to have Datafari source code to be checkouted (usually your workspace folder).

Perform a

```
git clone https://github.com/francelabs/datafari.git
```

to checkout the code. The root directory name is datafari.

Build Datafari for the development environment

Navigate to the folder datafari. We now call it **DATAFARI_SRC**.

Run a : 'mvn install' in DATAFARI_SRC.

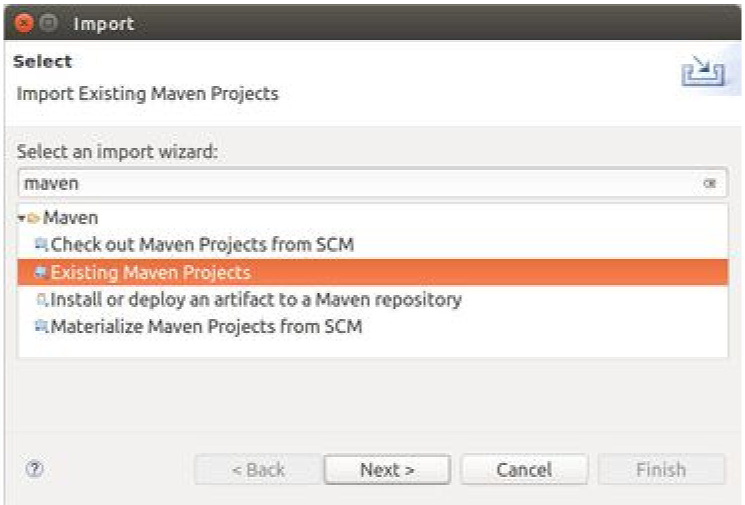
Run a : 'ant all' in DATAFARI_SRC/debian7

Install Datafari

Run a 'dpkg -i DATAFARI_SRC/debian7/installer/dist/datafari.deb and type an admin password.

Open the project in Eclipse

In Eclipse,go to File -> Import... , type maven and « Existing Maven project ».


Import
Select
Import Existing Maven Projects
Select an import wizard:
maven
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
< Back
Next >
Cancel
Finish

Select the DATAFARI_SRC folder as root directory and click Finish.

Change permissions to Datafari :

chmod -R 777 /opt/datafari

chmod -R 700 /opt/datafari/pgsql

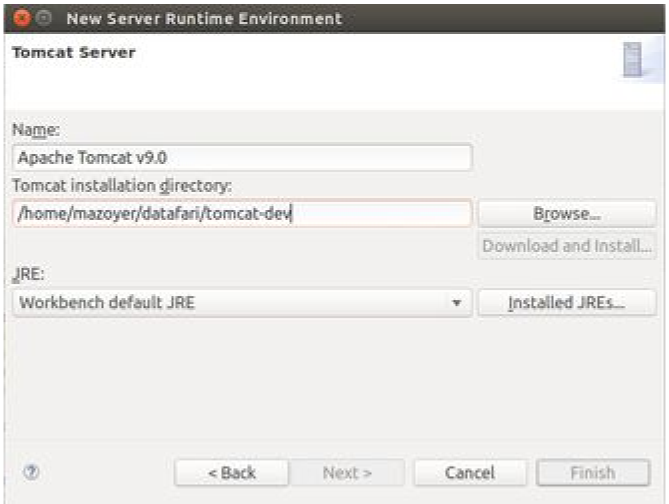
To add access rights to any user on datafari installation folder. Be careful : this should be done only for the development environment and should be avoided for a production deployment!

Setup the Tomcat server in Eclipse

Go to Window -> Preferences -> Server -> Runtime Environments -> Add…

and select Apache Tomcat V.9.0.

Select DATAFARI_SRC/tomcat-dev for the Tomcat Installation directory and Finish.



New Server Runtime Environment

Tomcat Server

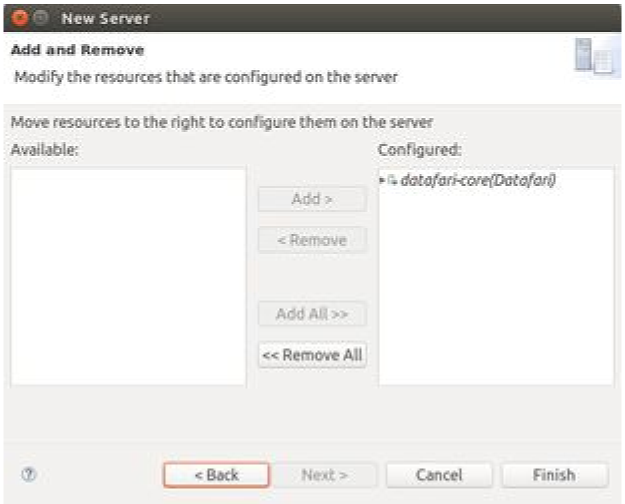
Name: Apache Tomcat v9.0

Tomcat installation directory: /home/mazoyer/datafari/tomcat-dev

JRE: Workbench default JRE

If Eclipse does not accept the Tomcat version, stop Eclipse, and copy the patch you can download here https://bugs.eclipse.org/bugs/attachment.cgi?id=262418&action=edit in the plugins directory of Eclipse installation directory.

In a Servers window (for example in Debug Perspectives), click on "add a server", select a "tomcat 9.0 server". Click on next and add Datafari webapp :



New Server

Add and Remove

Modify the resources that are configured on the server

Move resources to the right to configure them on the server

Available:

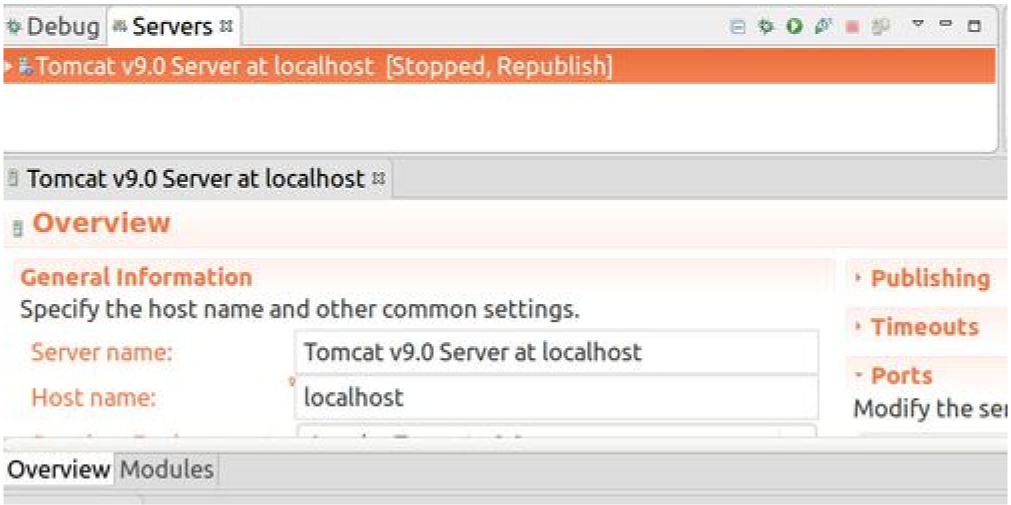
Configured: datafari-core(Datafari)

Click Finish.

Run Datafari

Run Datafari in normal mode. Go to /opt/datafari/bin and run ./start-datafari.sh.

Start Tomcat server with a right click Start on the server:



Tomcat v9.0 Server at localhost [Stopped, Republish]

Tomcat v9.0 Server at localhost

General Information

Specify the host name and other common settings.

Server name: Tomcat v9.0 Server at localhost

Host name: localhost

Modify the ser

Go to the Datafari URL (http://localhost:9080/Datafari)

Update Solr configuration :

If you change the solr configuration in DATAFARI_SRC/datafari-solr/solr_home, you have to run the script updateSolrConfig.sh in DATAFARI_SRC/datafari-solr to update the config in Solr Cloud.

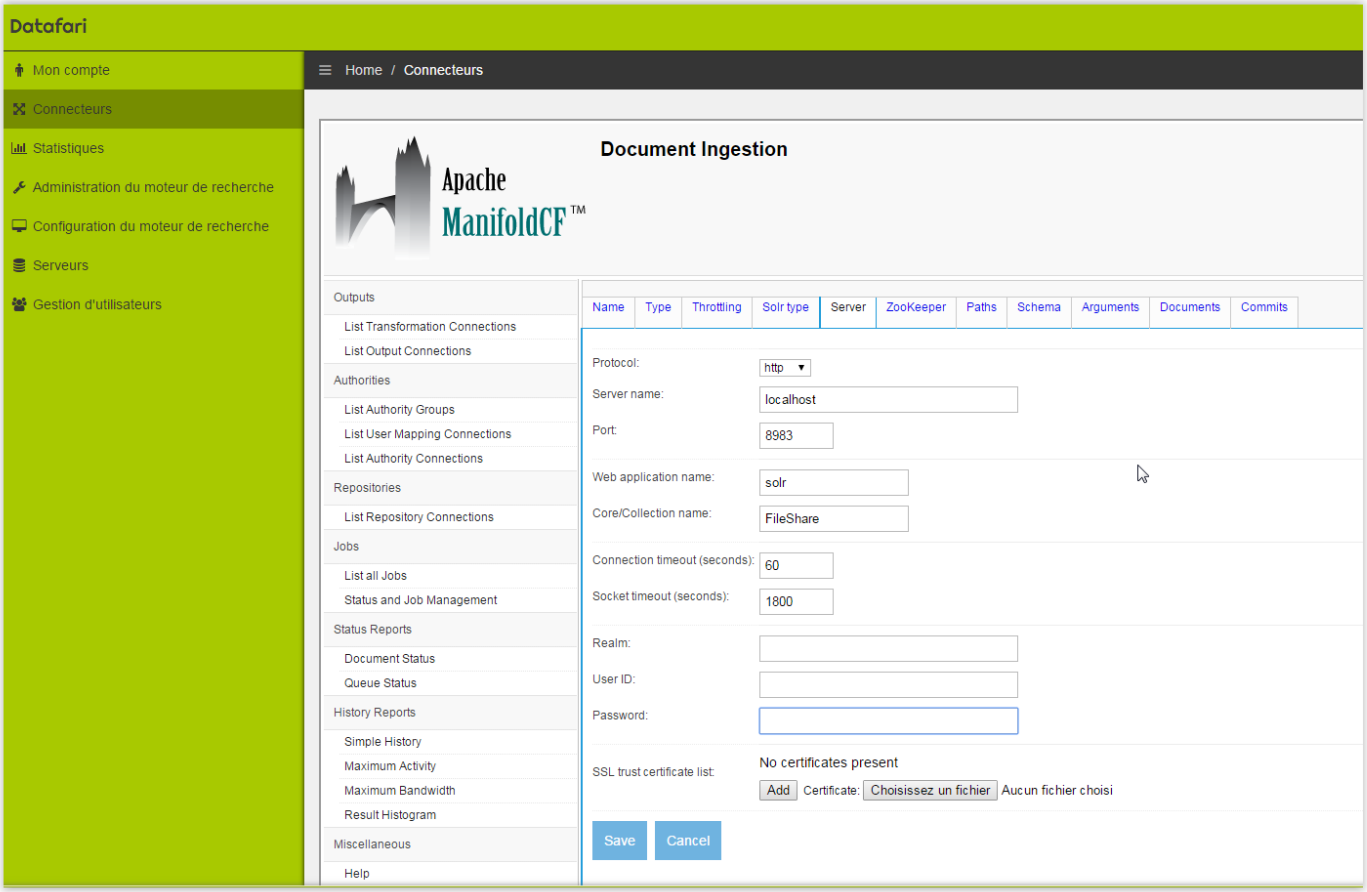
Configure connectors

Run Datafari tomcat. Go to the Datafari URL (http://localhost:9080/Datafari). Login in administrator (admin/admin).

In the left menu, click on Connectors; the Apache ManifoldCF login page shows up. Enter admin/admin to login (click on Login button, as the Enter key may not work).

Click on « List Output Connections » of Output section, and add a new « Output Connection ».
In Name tab, set the name to « DatafariSolr ».
In Type tab, set the connection type to Solr and then click on Continue button.
In the Server tab, set port to « 8983 », set web application name to « solr » and « Core/Collection name » to FileShare.

You can now save:



Document Ingestion

Protocol: http

Server name: localhost

Port: 8983

Web application name: solr

Core/Collection name: FileShare

Connection timeout (seconds): 60

Socket timeout (seconds): 1800

Realm:

User ID:

Password:

SSL trust certificate list: No certificates present

Certificate: Aucun fichier choisi

Your development environment is now ready.

For more information about ManifoldCF configuration, please have a look here: Crawling

Setup development environment with Eclipse (Linux) for Datafari 3.x - Maven

Pre-requisites

- Have a JDK installed on your PC (suggested version Java 7)
- Have an IDE installed on your PC (in the guide we use Eclipse)
- Have git installed on your PC
- Have Maven installed on your PC (at least version 3.0.5)
- Have Ant installed on your PC
- Have an Internet connection available (No offline installation permitted)

Wondering what are the possible Datafari build flows ? Have a look here Distributions HOW-TO

Clone repository

Open a terminal and navigate to the folder where you want to have Datafari source code to be checkouted (usually your workspace folder).

Perform a

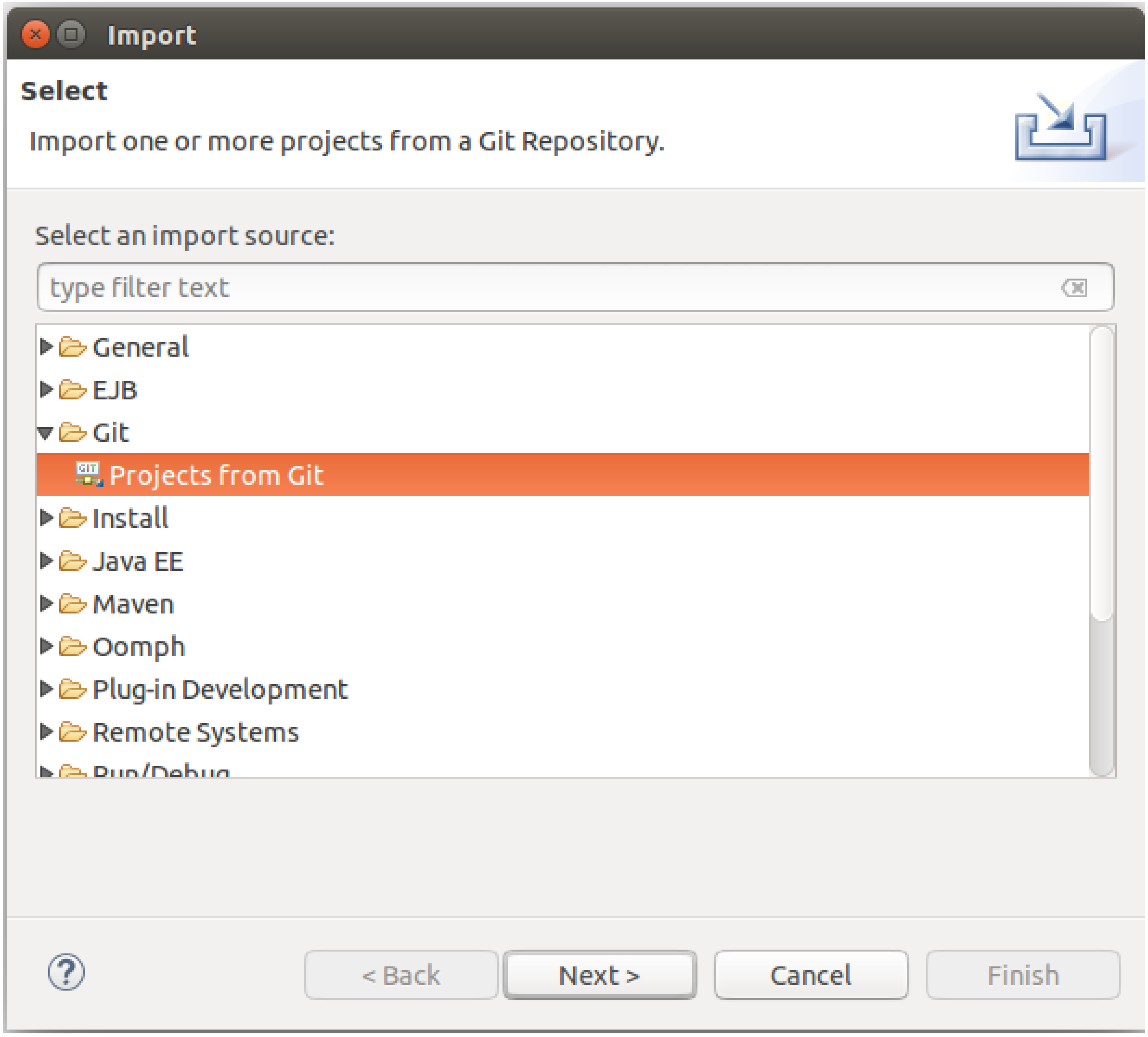
```
git clone https://github.com/francelabs/datafari.git -b master datafari
```

to checkout the code. The root directory name must be "datafari": we now call it **DATAFARI_SRC**.

Open the project in Eclipse

Start Eclipse by calling the executable from the workspace folder (the one directly containing DATAFARI_SRC).

In eclipse, go to File -> Import... , expand Git and do a « Projects from Git ».



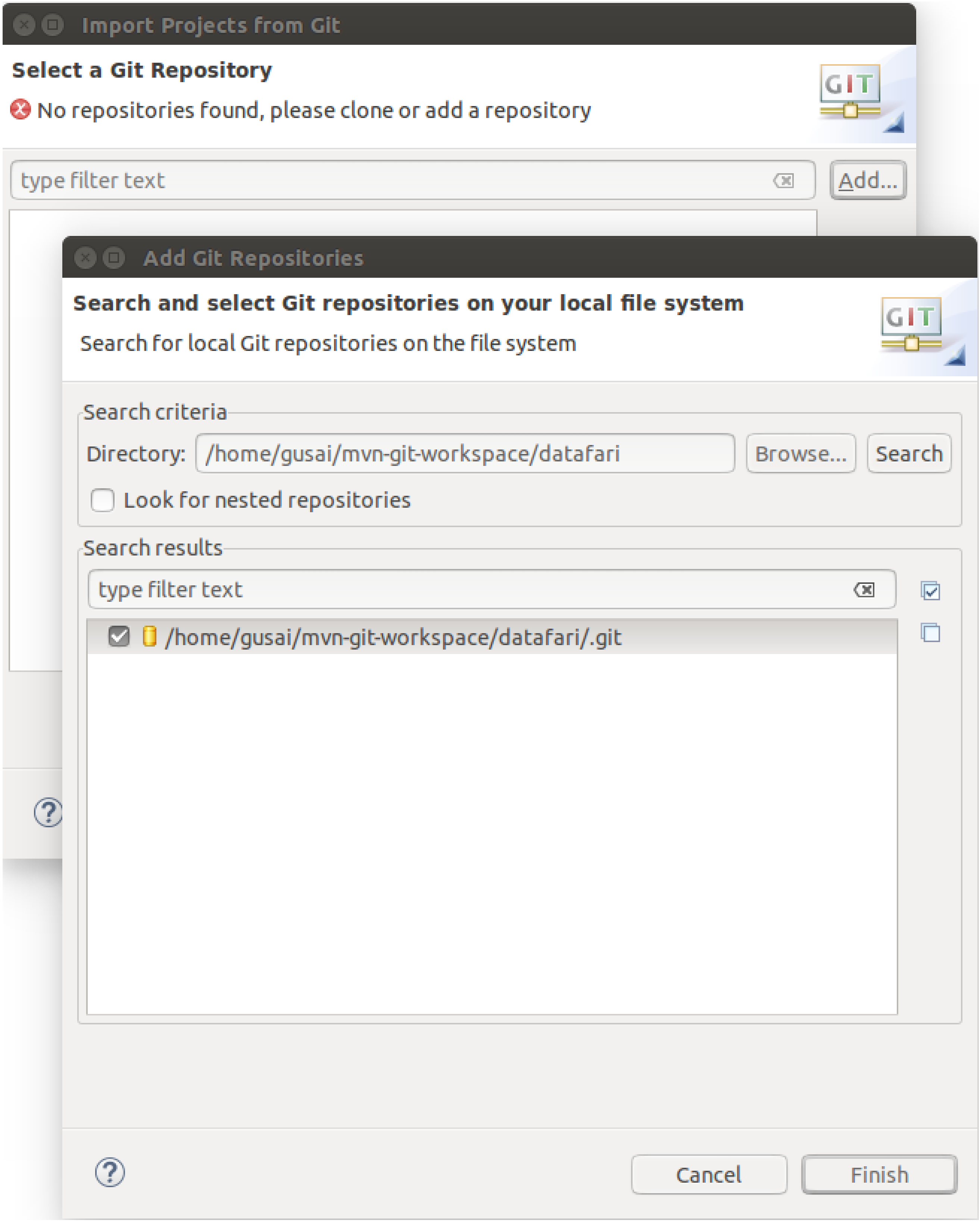
Import

Select

Import one or more projects from a Git Repository.

Select an import source:

Select "Existing local repository" and add the repo we have just cloned (by means of browse pop-up).



Select a Git Repository

No repositories found, please clone or add a repository

Search and select Git repositories on your local file system

Search for local Git repositories on the file system

Search criteria

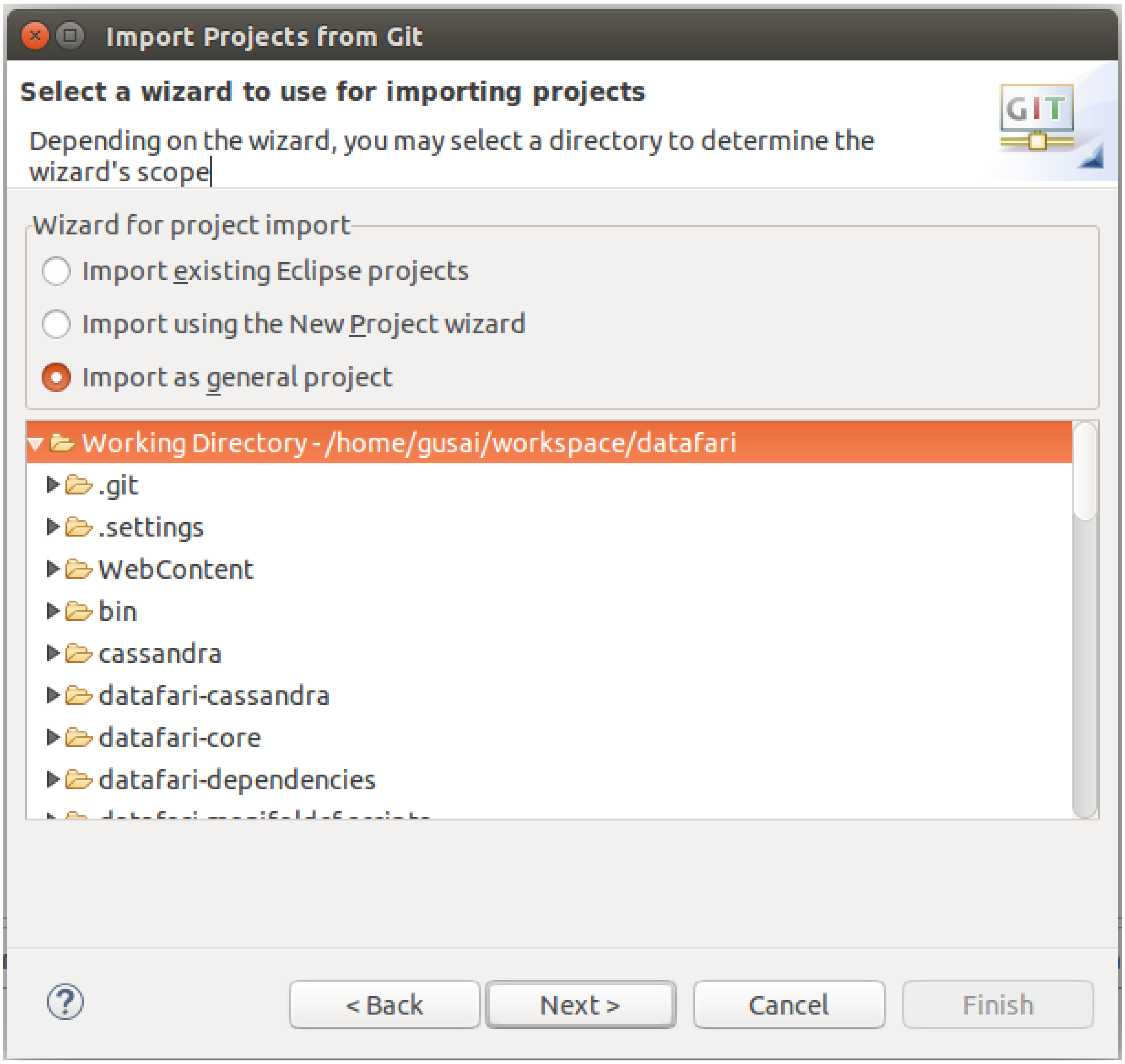
Directory: /home/gusai/mvn-git-workspace/datafari

□ Look for nested repositories

Search results

☑ /home/gusai/mvn-git-workspace/datafari/.git

Use "Import as general project" as Wizard. (Don't worry, the classpath and Eclipse settings will be set-up by Ant script).


Import Projects from Git

Select a wizard to use for importing projects

Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

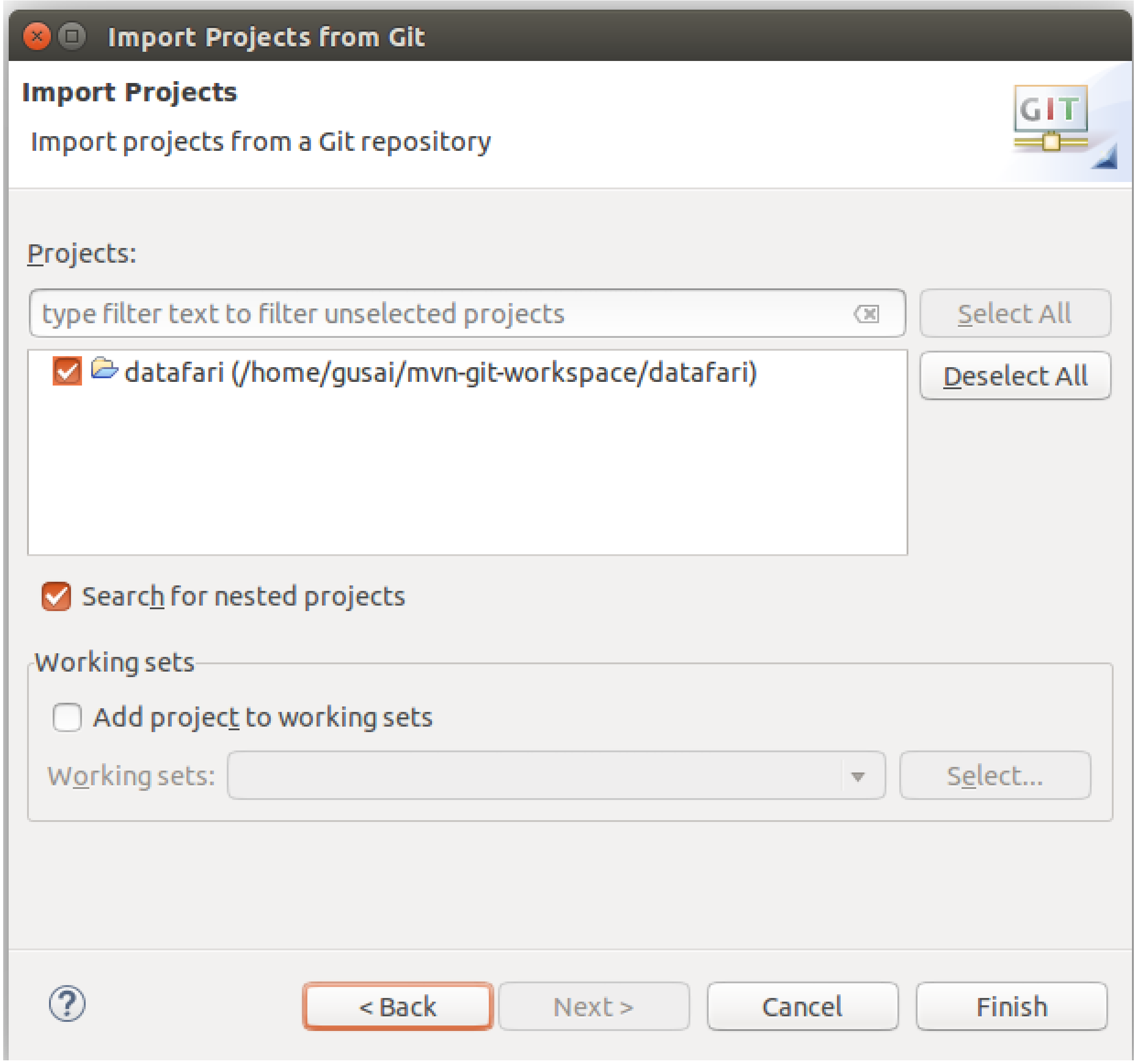
○ Import existing Eclipse projects

○ Import using the New Project wizard

● Import as general project

- Working Directory - /home/gusai/workspace/datafari
  - .git
  - .settings
  - WebContent
  - bin
  - cassandra
  - datafari-cassandra
  - datafari-core
  - datafari-dependencies

Select the DATAFARI_SRC folder as root directory and click Finish.



Import Projects from Git

Import Projects

Import projects from a Git repository

Projects:

type filter text to filter unselected projects

☑ datafari (/home/gusai/mvn-git-workspace/datafari)

☑ Search for nested projects

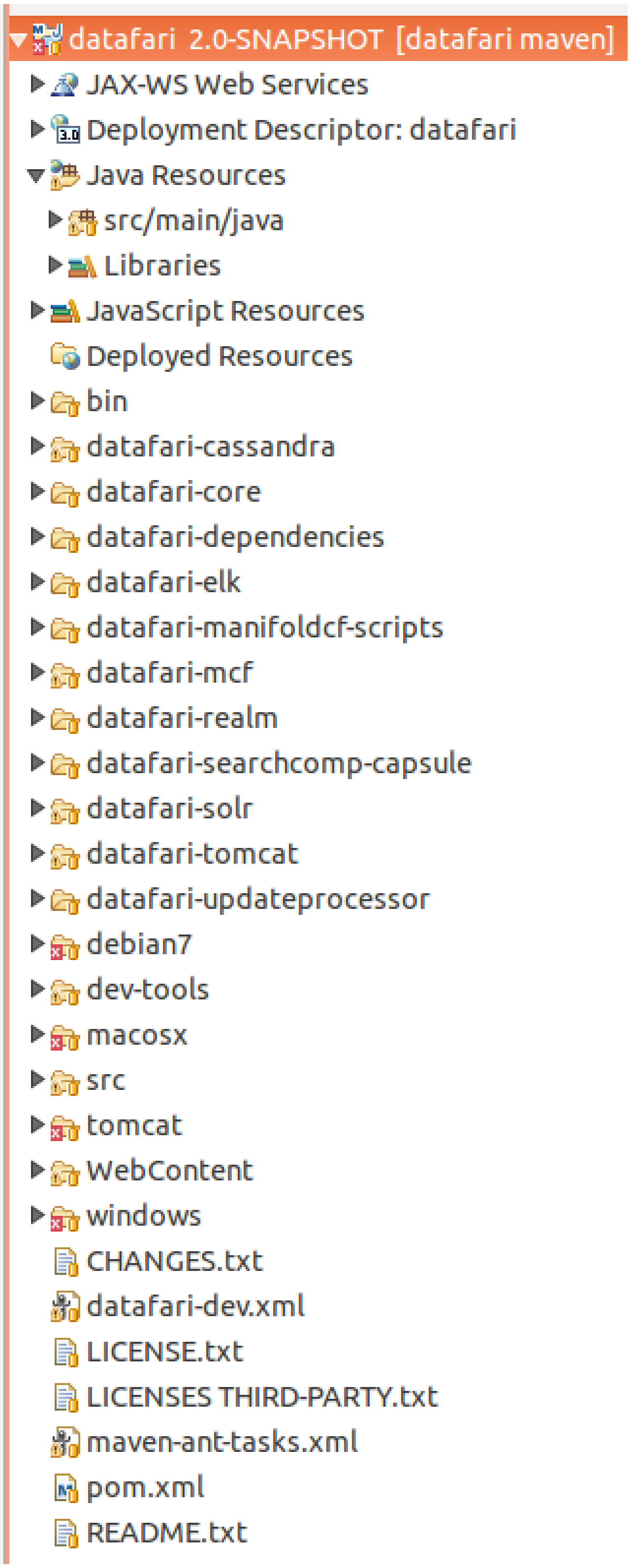
Working sets

☐ Add project to working sets

Working sets:

Leave Eclipse build the project automatically.

You should get this output, in project explorer view:



```
datafari  2.0-SNAPSHOT  [datafari maven]
  JAX-WS Web Services
  Deployment Descriptor: datafari
  Java Resources
    src/main/java
    Libraries
  JavaScript Resources
  Deployed Resources
  bin
  datafari-cassandra
  datafari-core
  datafari-dependencies
  datafari-elk
  datafari-manifoldcf-scripts
  datafari-mcf
  datafari-realm
  datafari-searchcomp-capsule
  datafari-solr
  datafari-tomcat
  datafari-updateprocessor
  debian7
  dev-tools
  macosx
  src
  tomcat
  WebContent
  windows
  CHANGES.txt
  datafari-dev.xml
  LICENSE.txt
  LICENSES THIRD-PARTY.txt
  maven-ant-tasks.xml
  pom.xml
  README.txt
```

As you can see, there are some "red crosses". This is not a problem as they are coming from external packages like ManifoldCF and Solr UI.

Eclipse configuration

In order to properly configure the classpath and the facets of the project, navigate to DATAFARI_SRC folder and run the following script:

```
ant -f ./dev-tools/setup-devenv/eclipse/setup-eclipse.xml
```

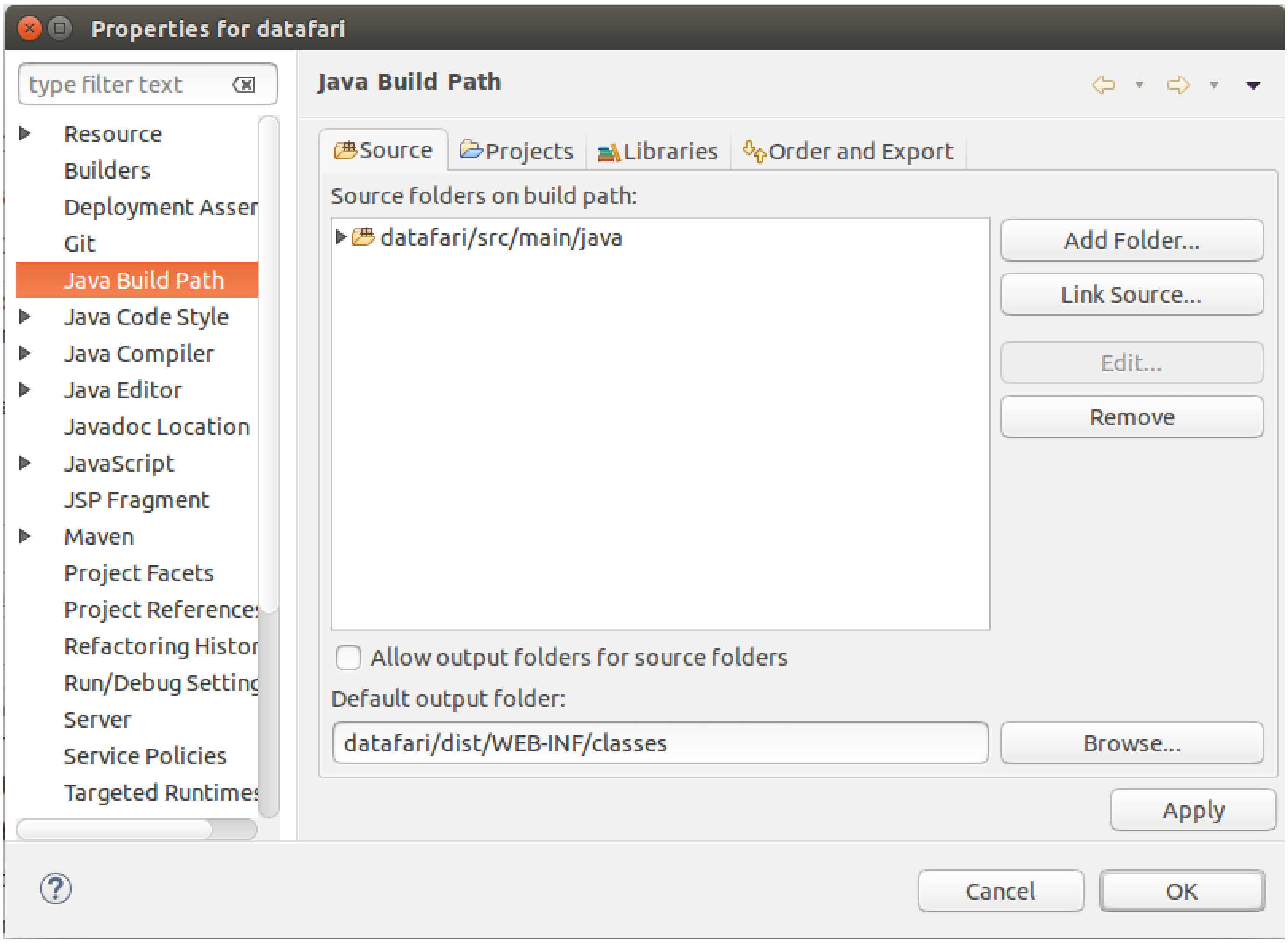
Check the Eclipse config

The Datafari Git repo contains some Eclipse config as well.

It's better to check that it has been correctly set-up =P:

Right click on datafari project -> Properties

Java Build path should be the following:



Properties for datafari

Java Build Path

Source folders on build path:

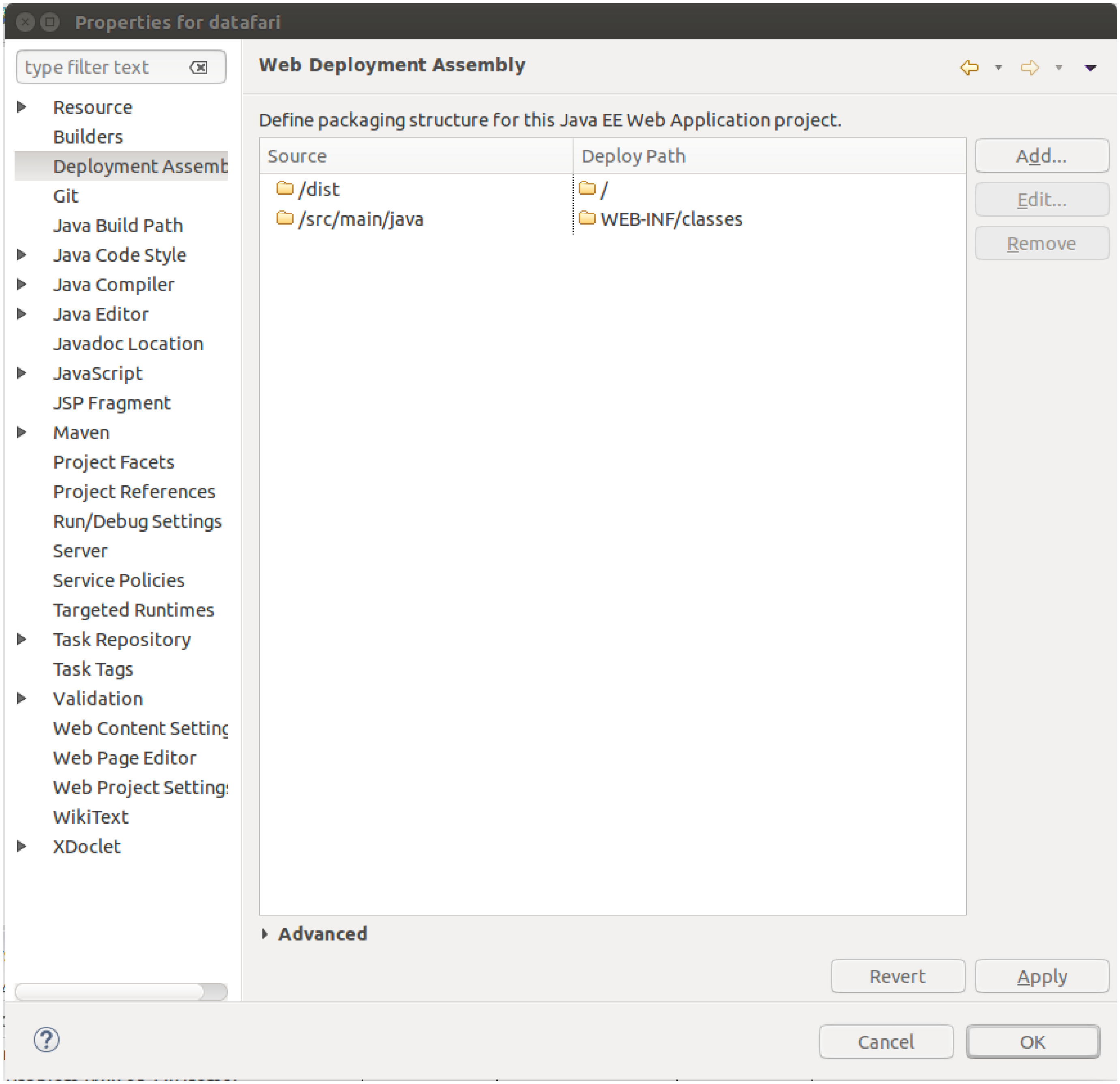
datafari/src/main/java

☐ Allow output folders for source folders

Default output folder:

datafari/dist/WEB-INF/classes

Deployment assembly:

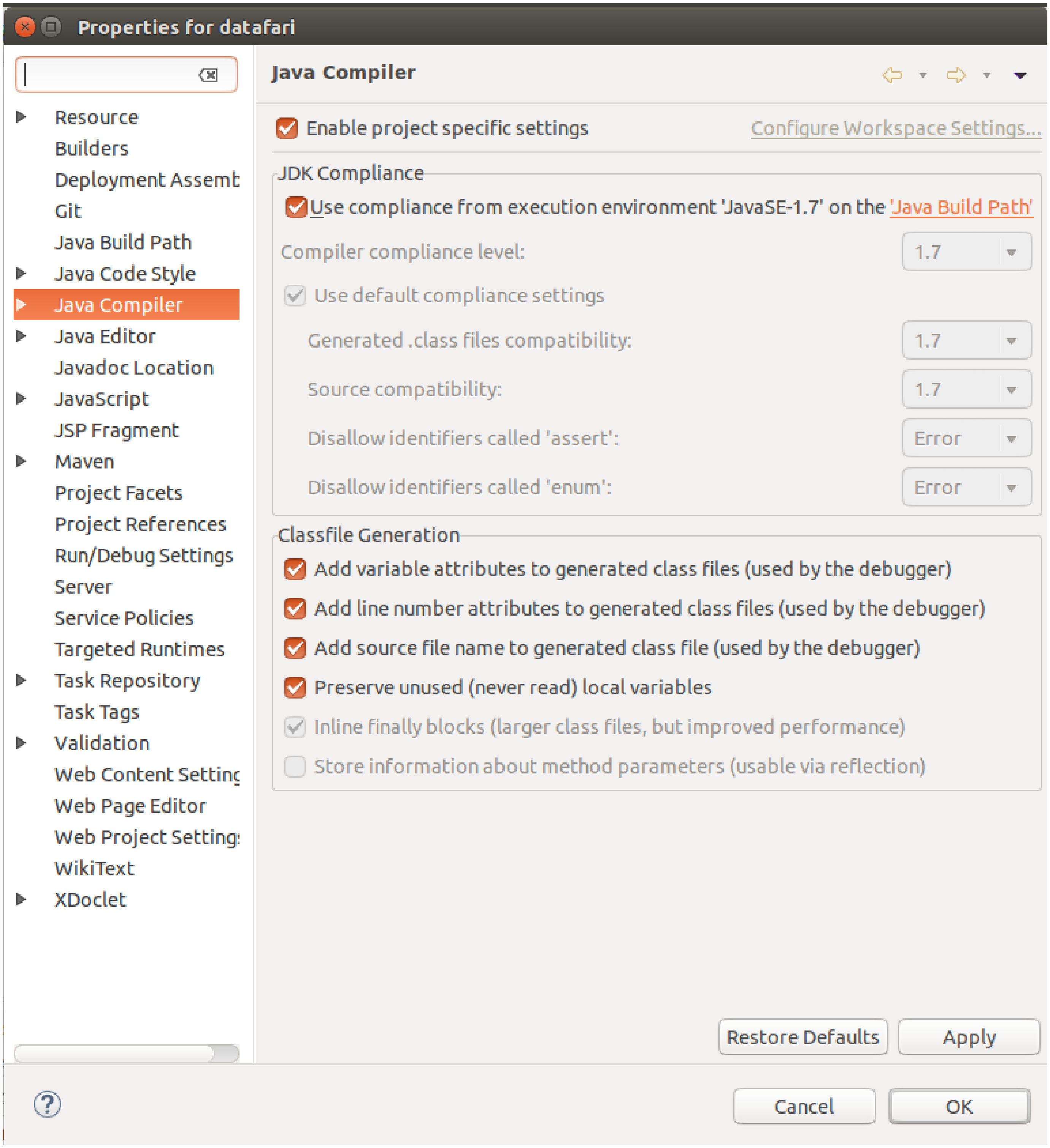


Web Deployment Assembly

Define packaging structure for this Java EE Web Application project.

Source	Deploy Path
/dist	/
/src/main/java	WEB-INF/classes

Java compiler should be as follows (Java 7):



Java Compiler

☑ Enable project specific settings

JDK Compliance

☑ Use compliance from execution environment 'JavaSE-1.7' on the 'Java Build Path'

Compiler compliance level: 1.7

☑ Use default compliance settings

Generated .class files compatibility: 1.7

Source compatibility: 1.7

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

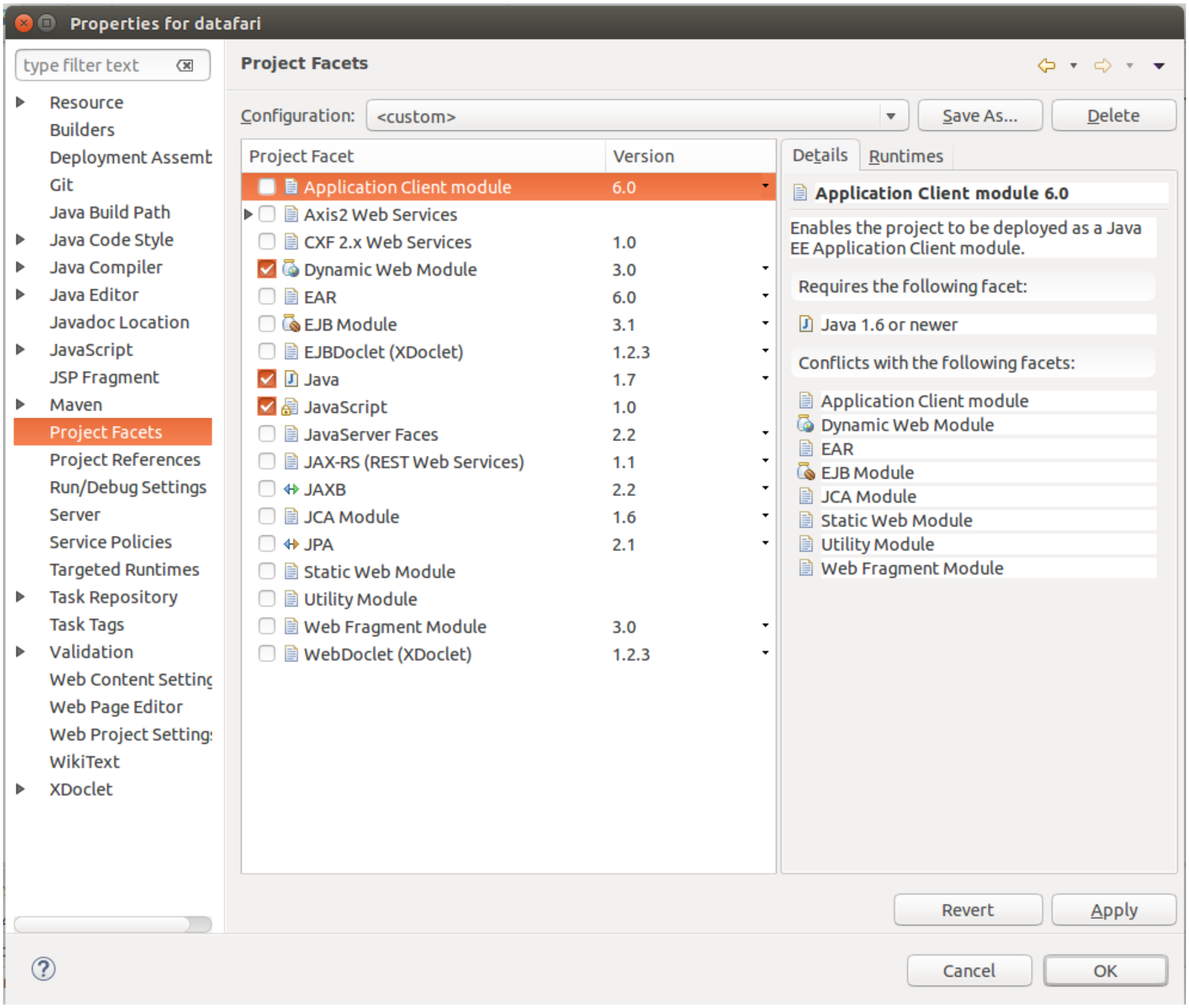
☐ Store information about method parameters (usable via reflection)

Project facets:

Dynamic Web Module V 3.0

Java V 1.7

JavaScript V 1.0



Project Facets

Configuration: <custom>

Project Facet	Version
□ Application Client module	6.0
□ Axis2 Web Services	
□ CXF 2.x Web Services	1.0
☑ Dynamic Web Module	3.0
□ EAR	6.0
□ EJB Module	3.1
□ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.7
☑ JavaScript	1.0
□ JavaServer Faces	2.2
□ JAX-RS (REST Web Services)	1.1
□ JAXB	2.2
□ JCA Module	1.6
□ JPA	2.1
□ Static Web Module	
□ Utility Module	
□ Web Fragment Module	3.0
□ WebDoclet (XDoclet)	1.2.3

Application Client module 6.0

Enables the project to be deployed as a Java EE Application Client module.

Requires the following facet:

Java 1.6 or newer

Conflicts with the following facets:

Application Client module
Dynamic Web Module
EAR
EJB Module
JCA Module
Static Web Module
Utility Module
Web Fragment Module

Add mcf-core lib to your maven repo

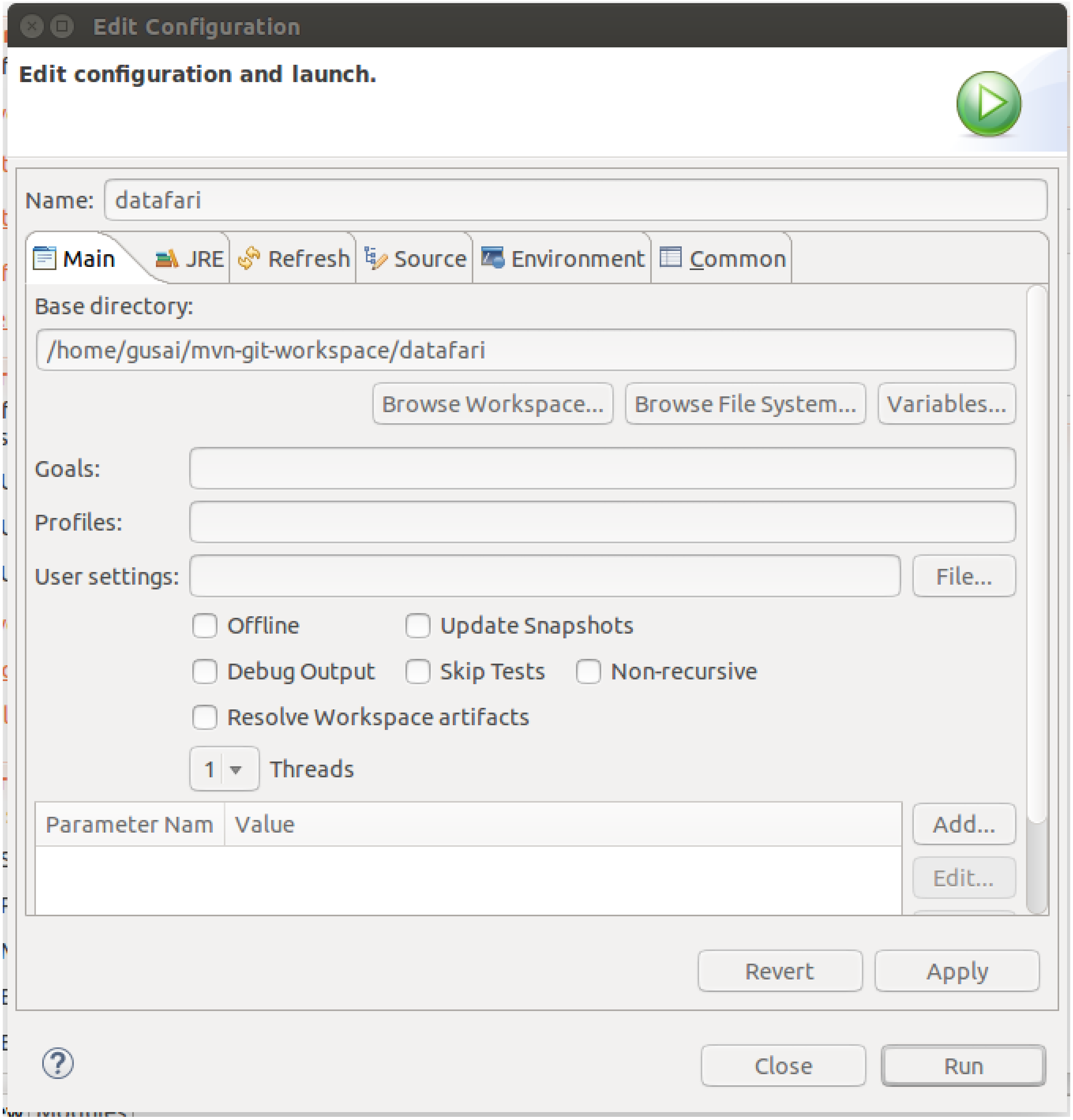
Datafari needs the mcf-core lib as a dependency. The problem is that Apache ManifoldCF does not have a maven repository so the library must be added manually to the maven repository. To do so, open a terminal and change directory to the main directory of Datafari, then execute the following command:

```
mvn install:install-file -Dfile=./datafari-mcf/lib/mcf-core-2.4.jar
-DgroupId=org.apache.manifoldcf -DartifactId=mcf-core -Dversion=2.4
-Dpackaging=jar -DgeneratePom=true
```

Run Maven build

Run Maven on DATAFARI_SRC/pom.xml in order to build the required libraries (Right click on pom.xml of datafari web project, then select Run As -> Maven build) :

Leave empty the goals section, the default one will be run.



Edit Configuration

Edit configuration and launch.

Name: datafari

Base directory:

/home/gusai/mvn-git-workspace/datafari

Goals:

Profiles:

User settings:

□ Offline □ Update Snapshots

□ Debug Output □ Skip Tests □ Non-recursive

□ Resolve Workspace artifacts

1 Threads

Parameter Nam	Value

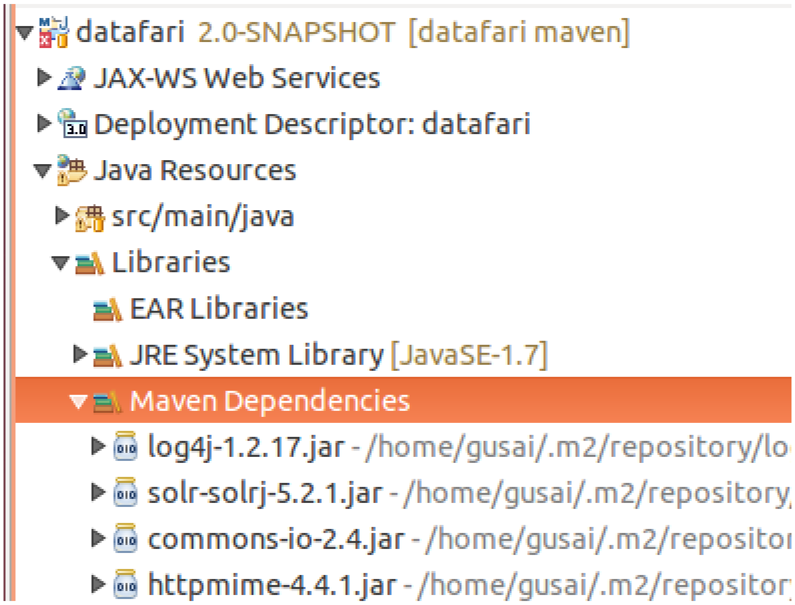
A console shows up with the output of Maven compilation.

At the end of the build process (it may take some time due to the download of external dependencies), you should see this output:

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Datafari Enterprise Search - Dependencies POM ...... SUCCESS [ 0.441 s]
[INFO] Datafari Enterprise Search - Parent POM ............ SUCCESS [ 32.289 s]
[INFO] Datafari Enterprise Search - Core module ........... SUCCESS [ 3.689 s]
[INFO] Datafari Enterprise Search - UpdateProcessor module SUCCESS [ 0.467 s]
[INFO] Datafari Enterprise Search - Capsule Search Component module SUCCESS [ 0.342 s]
[INFO] Datafari Enterprise Search - Realm module .......... SUCCESS [ 0.355 s]
[INFO] Datafari Enterprise Search - ManifoldCF Backup/Restore Scripts module SUCCESS [ 4.674 s]
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 42.438 s
[INFO] Finished at: 2016-01-28T12:34:38+01:00
[INFO] Final Memory: 51M/700M
[INFO] ------------------------------------------------------------------------
```

Now check to have the some jars listed in Maven dependecies. They should point to your Maven's repo directory, eg user_home/.m2/repository.



```
datafari 2.0-SNAPSHOT [datafari maven]
  JAX-WS Web Services
  Deployment Descriptor: datafari
  Java Resources
    src/main/java
    Libraries
      EAR Libraries
      JRE System Library [JavaSE-1.7]
      Maven Dependencies
        log4j-1.2.17.jar - /home/gusai/.m2/repository/lo
        solr-solrj-5.2.1.jar - /home/gusai/.m2/repository
        commons-io-2.4.jar - /home/gusai/.m2/reposito
        httpmime-4.4.1.jar - /home/gusai/.m2/reposito
```

If you see nothing in there or you have red crosses on some classes under Java Resources / src/main/java, right click on Datafari project, then select Maven entry and Update project.
Eclipse is not refreshing dependencies automatically.
The dependencies should show up under Maven Dependencies (if it's not the case, refresh the project).

Setup Tomcat server in Eclipse

ℹ Please note that we don't rely on Eclipse WTP environment, so the publish, clean, etc ... targets of server view are discouraged.
The only target you should run is the "start" of Tomcat.

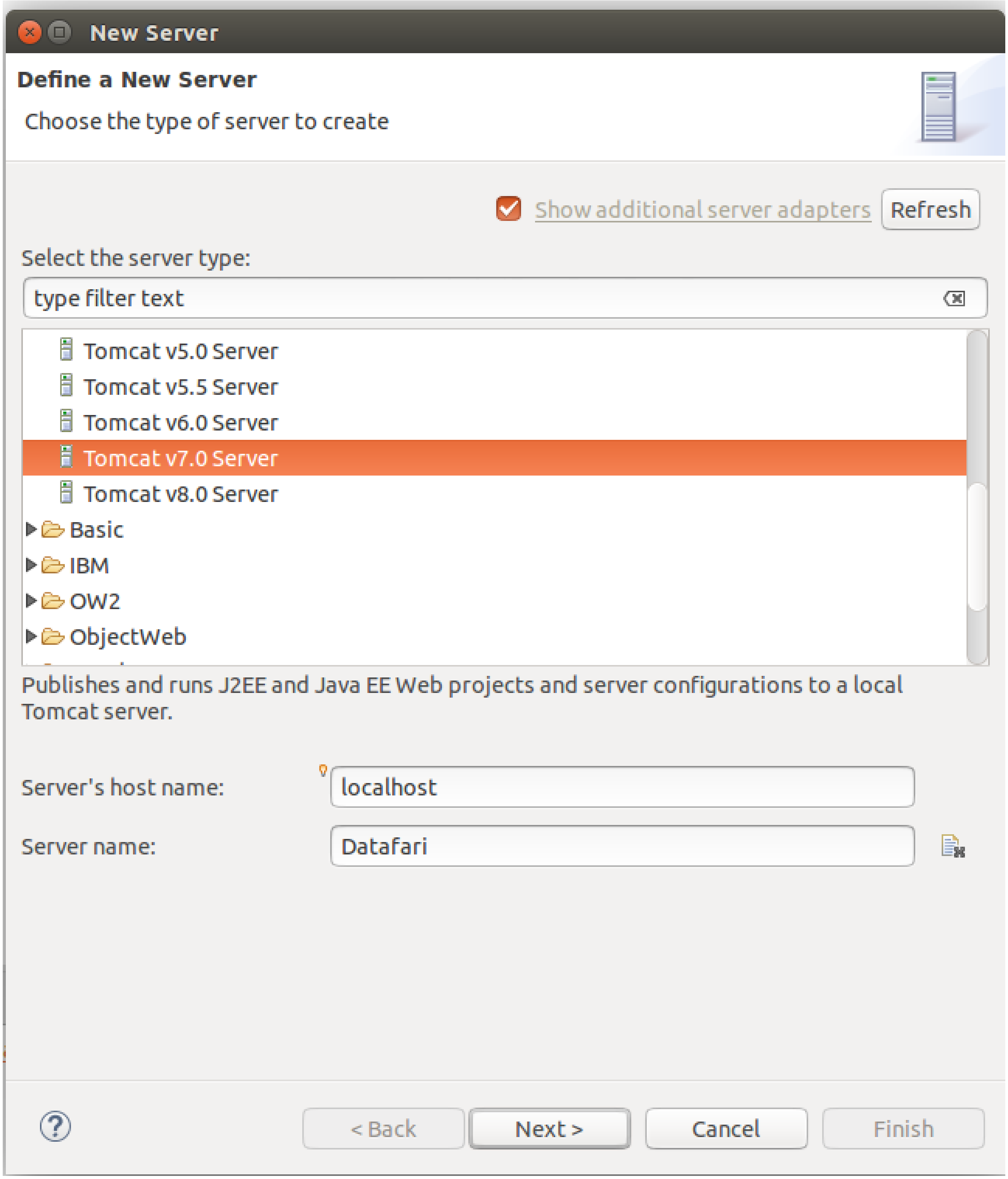
Add Server view: Window -> Show view -> Servers

and click on the link to create a new server.

Select Apache Tomcat V. 7.0 and put:

host name: localhost

server name: Datafari



Define a New Server

Choose the type of server to create

☑ Show additional server adapters

Select the server type:

type filter text

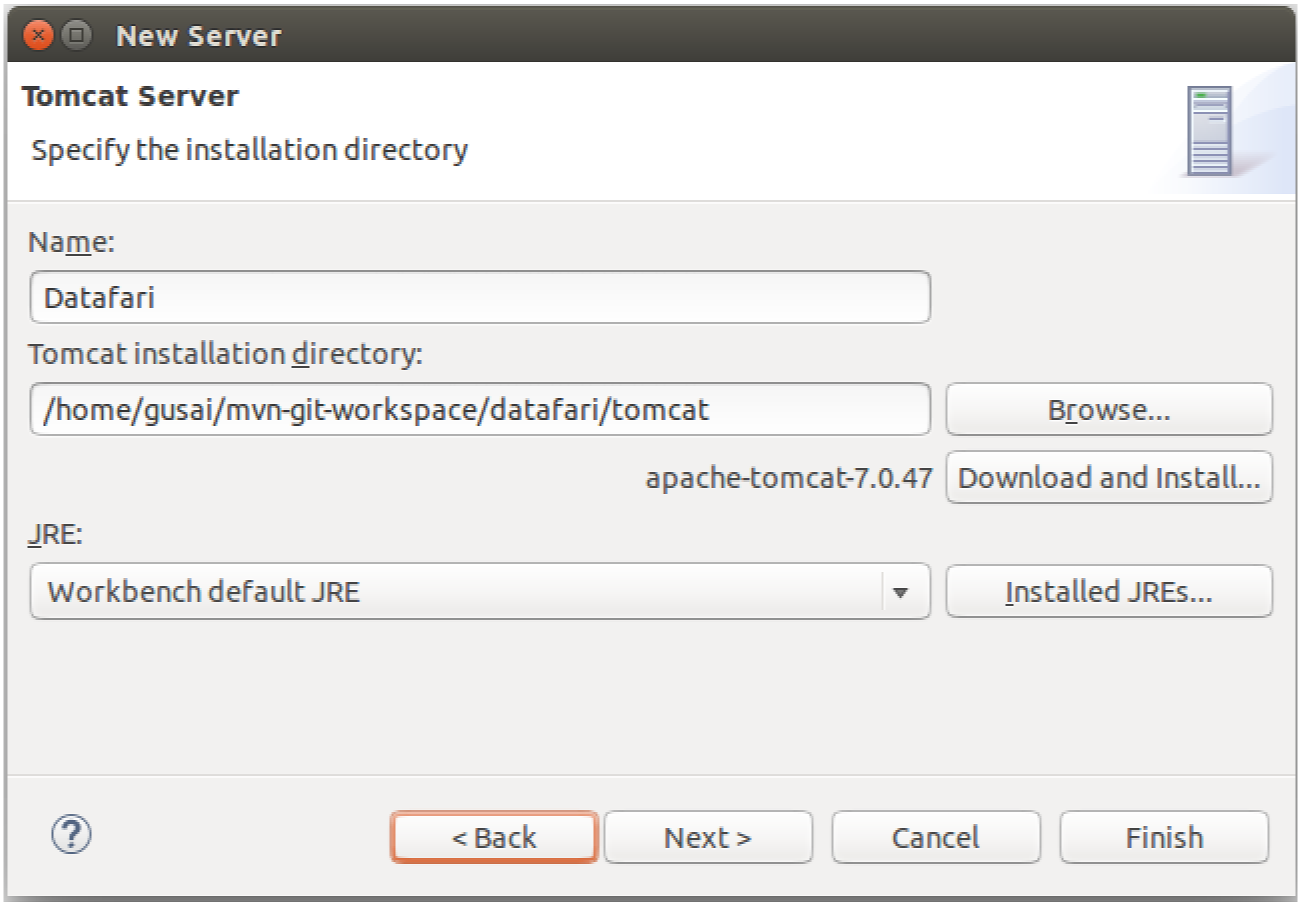
- Tomcat v5.0 Server
- Tomcat v5.5 Server
- Tomcat v6.0 Server
- Tomcat v7.0 Server
- Tomcat v8.0 Server
- ▸ Basic
- ▸ IBM
- ▸ OW2
- ▸ ObjectWeb

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Datafari

Select DATAFARI_SRC/tomcat for the Tomcat Installation directory and then next


New Server

Tomcat Server

Specify the installation directory

Name:

Datafari

Tomcat installation directory:

/home/gusai/mvn-git-workspace/datafari/tomcat

apache-tomcat-7.0.47

JRE:

Workbench default JRE

DO NOT add Datafari web application to the server and then Finish.

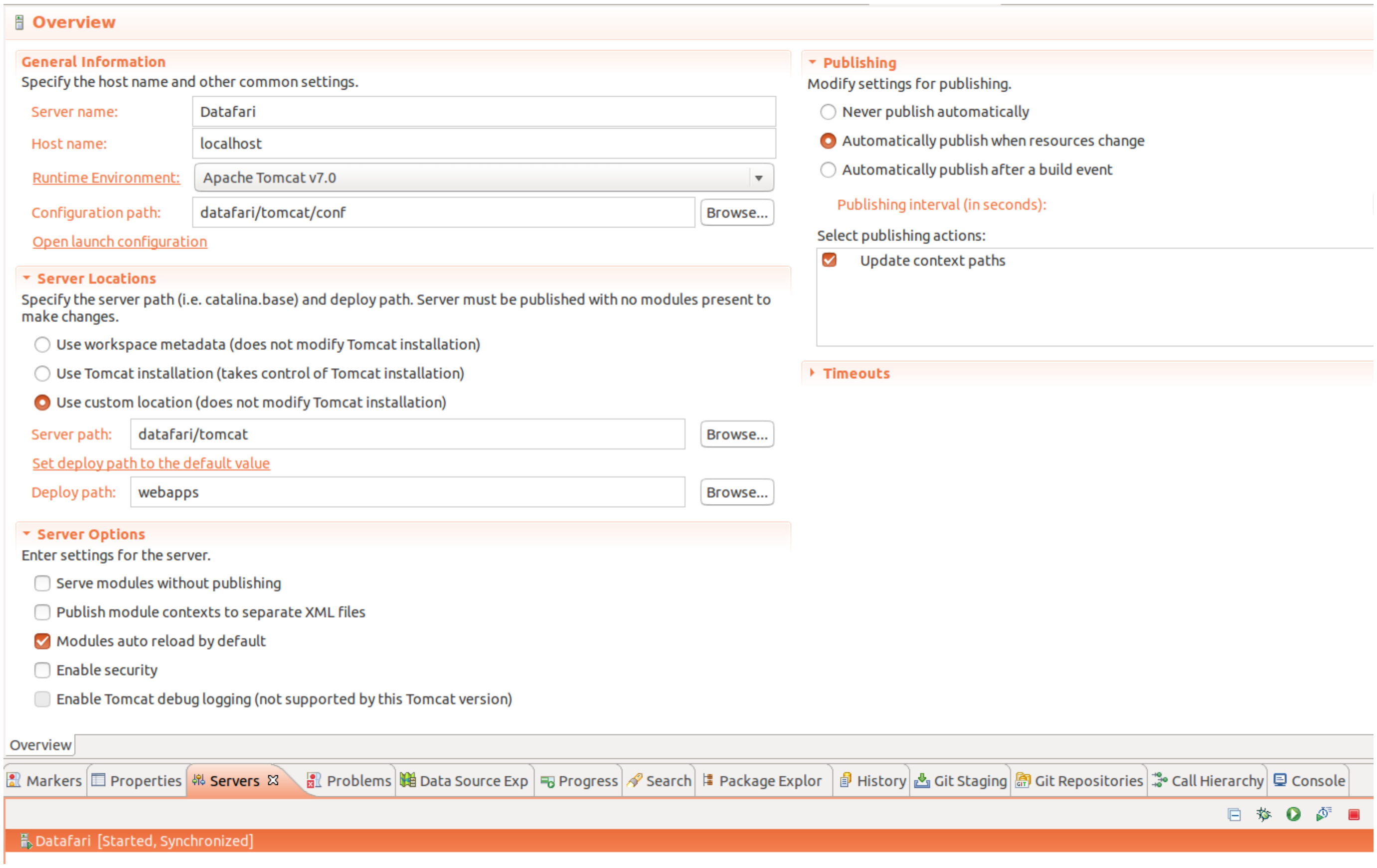
If Datafari-config project pops-out, you are safe to delete it (generated by Eclipse WTP).

Double click on Datafari server to change its configuration :

In Server Locations section, select "Use custom location" (to use the Tomcat embedded in Datafari).

Check that all the other configs are the same as in screenshot. You shouldn't see the Modules tab on the right of overview, or it should be empty (no deployed apps).

Notably, be careful that the paths are exactly as follows: datafari/tomcat/config and datafari/tomcat (no leading /)



Overview

General Information

Specify the host name and other common settings.

Server name: Datafari

Host name: localhost

Runtime Environment: Apache Tomcat v7.0

Configuration path: datafari/tomcat/conf

Open launch configuration

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

○ Use workspace metadata (does not modify Tomcat installation)
○ Use Tomcat installation (takes control of Tomcat installation)
● Use custom location (does not modify Tomcat installation)

Server path: datafari/tomcat

Set deploy path to the default value

Deploy path: webapps

Server Options

Enter settings for the server.

□ Serve modules without publishing
□ Publish module contexts to separate XML files
☑ Modules auto reload by default
□ Enable security
□ Enable Tomcat debug logging (not supported by this Tomcat version)

Publishing

Modify settings for publishing.

○ Never publish automatically
● Automatically publish when resources change
○ Automatically publish after a build event

Publishing interval (in seconds):

Select publishing actions:

☑ Update context paths

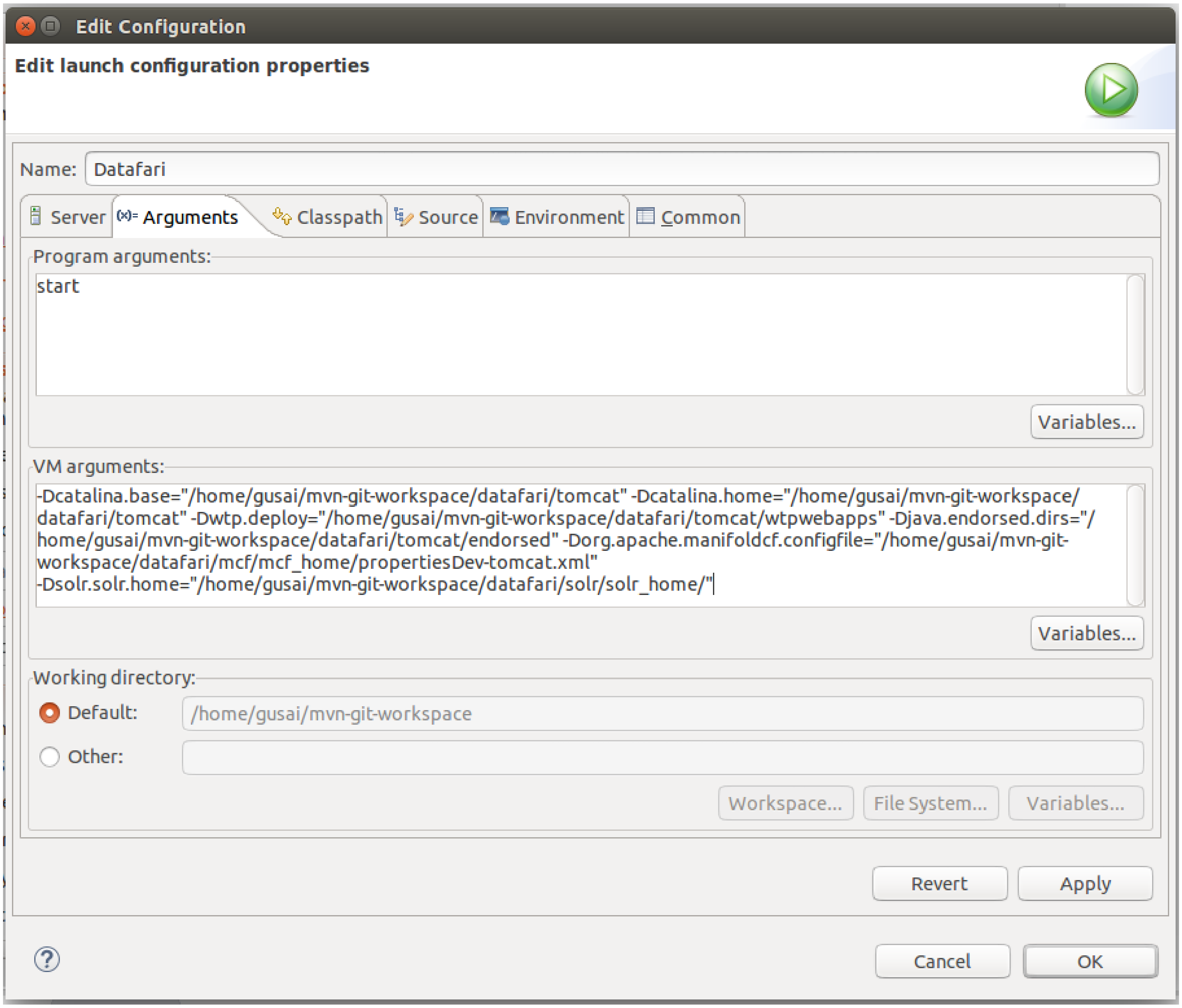
Timeouts

Datafari [Started, Synchronized]

In General Information section, click on « Open launch configuration ».

In Arguments tab, add this to VM arguments :

```
-Dorg.apache.manifoldcf.configfile="DATAFARI_SRC/mcf/mcf_home/propertiesDev-tomcat.xml"
-Dsolr.solr.home="DATAFARI_SRC/solr/solr_home/"
```



Name: Datafari

Program arguments:

```
start
```

VM arguments:

```
-Dcatalina.base="/home/gusai/mvn-git-workspace/datafari/tomcat" -Dcatalina.home="/home/gusai/mvn-git-workspace/datafari/tomcat" -Dwtp.deploy="/home/gusai/mvn-git-workspace/datafari/tomcat/wtpwebapps" -Djava.endorsed.dirs="/home/gusai/mvn-git-workspace/datafari/tomcat/endorsed" -Dorg.apache.manifoldcf.configfile="/home/gusai/mvn-git-workspace/datafari/mcf/mcf_home/propertiesDev-tomcat.xml"
-Dsolr.solr.home="/home/gusai/mvn-git-workspace/datafari/solr/solr_home/"
```

Working directory:

● Default: /home/gusai/mvn-git-workspace

○ Other:

and then OK.

Modules tab should be empty or not visible at all.

Run ZK, Solr and Cassandra

Start Datafari: Open a terminal and navigate to DATAFARI_SRC/dev-tools/script/dev.

- Edit the variable DEV_OS in dev-datafari.properties. if you are on Debian (let debian 7 for all versions of Debian) or MacOSX :

```
DEV_OS=debian7
```

The two possible variables are : debian7 or macosx.

The property STATE indicates if you launch for the first time the development environment.
The possible values are : installed if you never launched it and initialized if you have already launched at least one time the environment.

- Then start the script start-datafari-devmode.sh :

```
bash start-datafari-devmode.sh
```

For the first launch, it will need some time to launch and initialize Zookeeper, Solr adn Cassandra.

A new user 'admin' will be created in Cassandra database.

The credentials to login to Datafari admin are : **admin/admin.**

- To stop datafari :

```
bash stop-datafari-devmode.sh
```

Note that a convenient script is also present : restart-solr-devmode.sh in order to reload only Solr and not other services like ZK.

Run database and ManifoldCF agents

Open a terminal and navigate to DATAFARI_SRC/mcf/mcf_home.

Run

```
bash start-database-dev.sh.
```

In another terminal, run (ONLY ONCE, the first time you set-up Datafari dev environment)

```
bash initialize-dev.sh
```

and then

```
bash start-agents-dev.sh.
```

Keep the two terminals open.

Run Tomcat

Go to Eclipse -> Servers view.

Start it (right click on the server name -> start).
Datafari application is already deployed by Maven/Ant.

ℹ Please note that we don't rely on Eclipse WTP environment, so the publish, clean, etc ... targets of server view are discouraged.
The only target you should run are the "start / stop" of Tomcat.

Note: if you get this exception

INFO: Exception performing authentication. Retrying...
javax.naming.CommunicationException: 52.16.74.128:389 [Root exception is java.net.SocketTimeoutException: connect timed out]

it's the, let's say "normal" behaviour =P! (Fix on-going)

Go to the Datafari URL (http://localhost:8080/Datafari)

To login into Datafari, the credentials are : admin/admin

HAPPY SEARCHING 😀 !!!

💡 Need to get Datafari ready for production? Have a look here Distributions HOW-TO

Datafari incremental build and redeployment

For redeployment of Datafari Java sources and web resources ONLY (e.g. during an on-going dev), you can use the handy Ant targets "all" or "redeploy" of datafari-dev.xml.
This script recompiles the Datafari core sources and deploys the application on Tomcat.
ℹ If you get strange errors while running Ant, double check to run it on a JDK and not on a JRE (Oracle one, for instance).

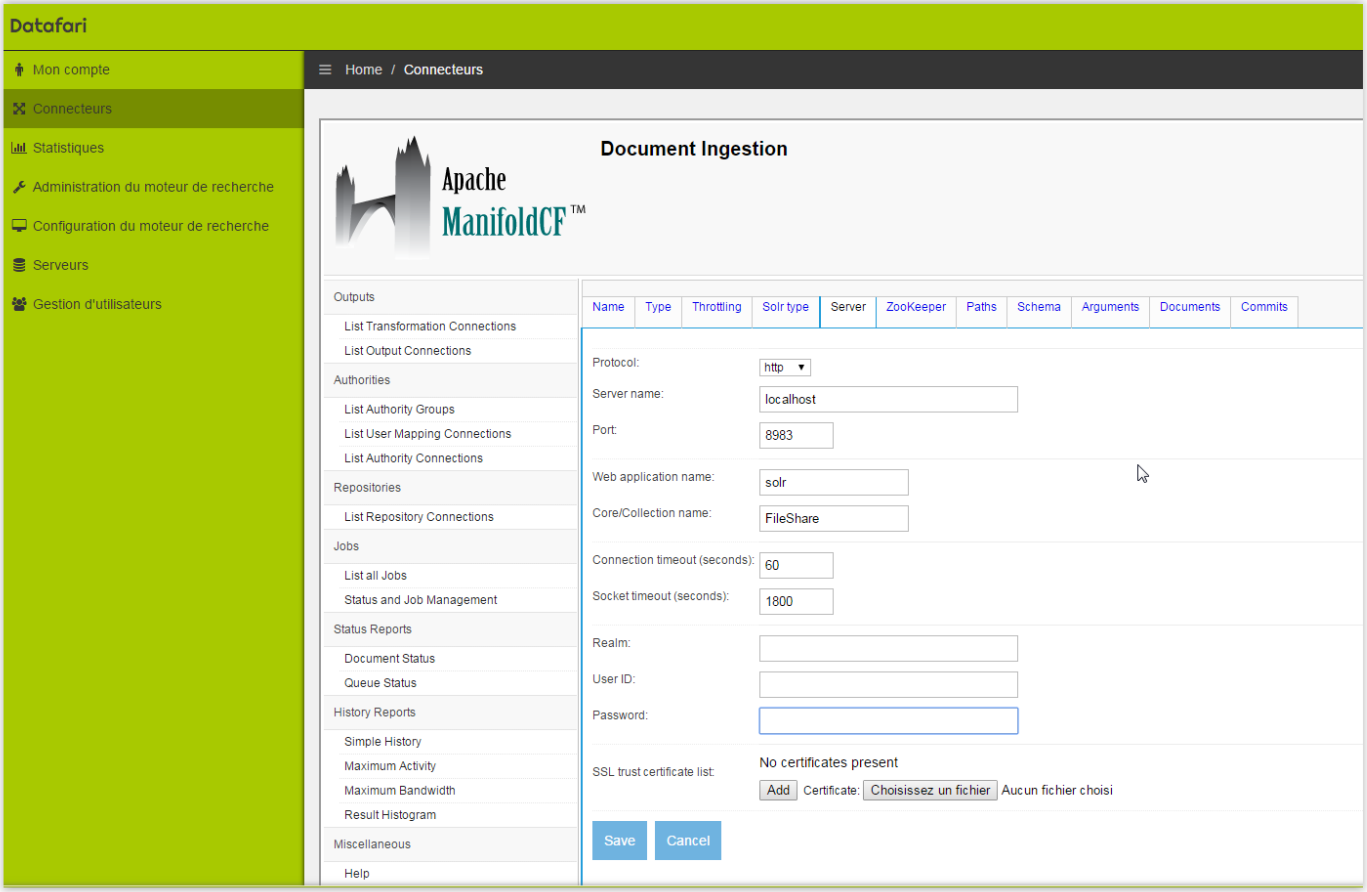
Configure connectors

Run Datafari tomcat. Go to the Datafari URL (http://localhost:8080/Datafari). Login in administrator (admin/admin).

In the left menu, click on Connectors; the Apache ManifoldCF login page shows up. Enter admin/admin to login (click on Login button, as the Enter key may not work).

Click on « List Output Connections » of Output section, and add a new « Output Connection ».
In Name tab, set the name to « DatafariSolr ».
In Type tab, set the connection type to Solr and then click on Continue button.
In the Server tab, set port to « 8983 », set web application name to « solr » and « Core/Collection name » to FileShare.

You can now save:



Protocol: http
Server name: localhost
Port: 8983
Web application name: solr
Core/Collection name: FileShare
Connection timeout (seconds): 60
Socket timeout (seconds): 1800
Realm:
User ID:
Password:
SSL trust certificate list: No certificates present
Certificate: Aucun fichier choisi

Your development environment is now ready.

ⓘ For more information about ManifoldCF configuration, please have a look here: Crawling

Unit Testing

Datafari 3.x comes with some Unit Tests (at last !)

Three frameworks for Unit Testing are embedded in this version of Datafari :

- JUnit : the standard JAVA Unit Testing framework
- Mockito : famous framework that allow to mock JAVA objects
- XMLUnit : very useful framework for XML comparison

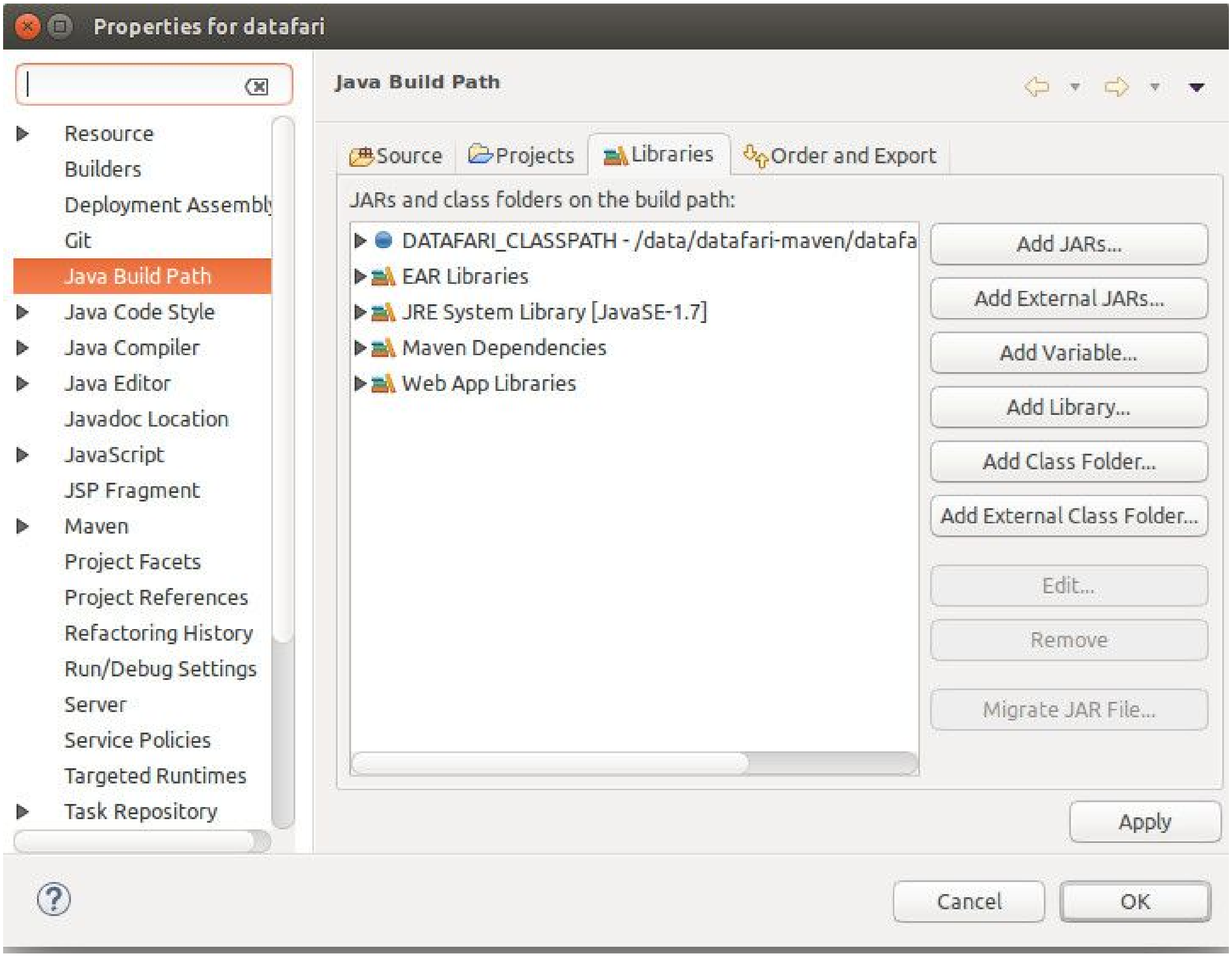
We think that these three frameworks cover more than 90% of the needs when creating a Unit Tests, so now you have no good excuses to create Unit Tests when it is possible for your development.

Now, let's try to explain our usages and good practices of the Unit Tests in Datafari:

- The first thing to know is that all the existing and future Unit Tests have to be located in Datafari_Home/src/test. As you will notice, there are two sections in this directory:

- main/java : That contains all the JUnit JAVA classes
- resources : That contains the resource files needed by the Unit Tests. Each Test which is using one or several resource files, must have his own repertory in this section. It is then easier to access to the resource files corresponding to a specific Test

To be able to run the tests from the eclipse environment, you will need to add a Variable to the classpath of Datafari :
Right click on the project Build Path Configure Build Path



Now go to the libraries tab and click on "Add Variable", then "Configure Variables" and click on "New". Name it "DATAFARI_CLASSPATH" and for the Path browse to the Datafari_Home/dist/WEB-INF/classes.Then click "OK" till you return to the Libraries tab and click "Apply" then "OK"

- The second point is that the ant script "datafari-dev.xml" which is used to compile and deploy the web part of Datafari (inluding the servlets) to the tomcat server, is also configured to automatically run all the JUnit tests found in the Datafari_Home/src/test/main/java directory. So when you are done with a new development or a modification of Datafari, check the output of this ant script to see if all the tests have passed with success.

- The third and last thing to understand is the kind of Unit Tests we are expecting in Datafari. The usage of Unit Testing is limited in Datafari, because the product has many interactions with tier applications (ELK, Solr, ManifoldCF, Cassandra) and that it makes no sense to try to mock those components as they represents the core of Datafari. So, based on this observation, we have focused our Unit Tests on another important part : the manipulation of the configuration files. It applies for now to the config files of Datafari, Solr and ELK. Our tests generate calls to the servlets in charge of the modifications/access to the config files and compare their behavior and their outputs to the expected ones. In that way we insure well formed configuration files and presence of every expected parameters and values.

So now that you know what is our goal with the Unit Tests and that Datafari provides you the tools to make your owns, we hope and encourage you to enrich Datafari with a lot of Unit Tests.

Utility logs

The 2.2 version of Datafari introduces utility logs, which are distributed in two types:

- the Statistics logs
- the Core Monitoring logs

Core Monitoring logs

Datafari periodically generates monitoring logs about the main Solr core. Those logs are intended to provide general informations on the core contents (like number of indexed doc by types, by source, etc.).

By default, the generation rate is once per hour, and can be currently modified through the java class "com.francelabs.datafari.monitoring.IndexMonitoring".
Here is how monitoring logs look like:

monitoring logs

```
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:175 - ec6d7ff440530d73e020766403a3e058|2015-11-06T18:14:00.000+0100|40|no|no
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - dd3fb399e47dced897b708d9c6e78d2f|2015-11-06T18:14:00.000+0100|11|docx|extension
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - 039435d1512eb763234a2690fb06c63c|2015-11-06T18:14:00.000+0100|6|pdf|extension
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - 611555203c75e7880c8dc67afd41062d|2015-11-06T18:14:00.000+0100|4|txt|extension
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - 2ce23ed298b044a3eda0a0e695921024|2015-11-06T18:14:00.000+0100|3|png|extension
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - 0fc7bbcad0efcfc2037ae2a2f1fbfe04|2015-11-06T18:14:00.000+0100|2|doc|extension
2015-11-06 18:14:00 MONITORING IndexMonitoring$FileShareMonitoringLog:184 - 31f8884b788c7872e5b2a286faa42747|2015-11-06T18:14:00.000+0100|2|html|extension
```

```
2015-11-06 18:14:00 MONITORING
IndexMonitoring$FileShareMonitoringLog:184 -
e155fbb2f018035d65553e763e0d8e63|2015-11-06T18:14:00.000+0100|1|gif|exte
nsion
2015-11-06 18:14:00 MONITORING
IndexMonitoring$FileShareMonitoringLog:184 -
c9dc854a334e1dc339d6cdfd10cb37cf|2015-11-06T18:14:00.000+0100|1|jar|exte
nsion
2015-11-06 18:14:00 MONITORING
IndexMonitoring$FileShareMonitoringLog:184 -
c10e628086bdc9fe4126c5f9c3b222c6|2015-11-06T18:14:00.000+0100|1|jpg|exte
nsion
2015-11-06 18:14:00 MONITORING
IndexMonitoring$FileShareMonitoringLog:184 -
24ca7afe128eb7b33b36c4919e830509|2015-11-06T18:14:00.000+0100|16|en|lang
uage
2015-11-06 18:14:00 MONITORING
IndexMonitoring$FileShareMonitoringLog:184 -
8879abd5830038c4a630d07526857faf|2015-11-06T18:14:00.000+0100|15|fr|lang
uage
2015-11-06 18:14:00 MONITORING
```

```
IndexMonitoring$FileShareMonitoringLog:184 -
dc8834aa358f02cff45e71052dd648c0|2015-11-06T18:14:00.000+0100|30|file|so
urce
```

They respect a specific format which is:
[log_id] | [timestamp] | [num_doc] | [facet_value] | [facet_field]

Let's explain each field:

- [log_id] : the log id. This id IS UNIQUE per time event, per facet value and field, and is generated from these 3 values. The reason behind this is the ELK.
- [timestamp] : the timestamp of the log. It is formatted and could be different from the log4j timestamp so it is mandatory
- [num_doc] : the number of documents found according to the facet
- [facet_value] : the facet value
- [facet_field] : the facet field used, in relation with the facet value

At each monitoring log generation iteration, a log line is created for each facet value of the selected facet fields, plus a log line for the global number of documents indexed (in that case, the [facet_value] and [facet_field] are both set to "no").
The default selected facet fields are "extension", "language" and "source". One can change the facet fields selected in the class "com.francelabs.monitoring.IndexMonitoring.FileShareMonitoringLog".

Statistics logs

Datafari generates a statistic log each time a query is performed by a user or he/she clicks on a facet/result/page.
Here is how a statistic log looks like:

statistic log

```
2015-11-05 15:24:45 STAT  StatsPusher:95 -
508e9b3e-fdb0-4831-8fab-2acad81f1cb5|2015-11-05T14:21:29.740+0100|engine
|0|4|1|7|2|1|[engine//////4///7///1//////, engine///(extension:docx
)///2///6///1//////,
///////////////file:/home/youp/Downloads/doc/Alertes.docx///2]|file:/hom
e/youp/Downloads/doc/Alertes.docx
```

It respects a specific format which is :
[query_id] | [query_timestamp] | [query] | [noHits] | [numFound] | [numClicks] | [QTime] | [positionClickTot] | [click] | [history] | [spell] | [suggest] | [url]

Let's explain each field:

- [query_id] : the id of the query. This id is used to keep track of the user behaviour.
  A query id is generated each time the user clicks on the search button, or presses the "Enter" key when the focus is on the search field of Datafari. However, when the user performs a "sub-query" like selecting a facet or a page or even clicking on a result, the "sub-query" keeps the id of the root query.
  For example, if the user searches for the word "engine", a query id is generated. Then, if the user clicks on the facet "fr" on the query results, the facet "sub-query" will keep the id of the root "engine" query. So several log lines can concern one query id as one log is generated on each action performed
- [query_timestamp] : the full timestamp of the query, measured by Datafari
- [query] : the literal query performed by the user
- [noHits] : indicates if the query does or does not have hits. Two values are possible, '0' if the query has hits, '1' if the query does not have any hit
- [numFound] : the number of documents found for the query
- [numClicks] : the number of clicks on the found documents
- [QTime] : the query time in milliseconds. This value is directly provided by Solr
- [positionClickTot] : represents the sum of the positions of the documents clicked.

For example, if the user clicks on the first result and the third, the positionClickTot will be 1+3=4
The positions are absolute and are not based on the page. If there are 10 results by page and the user clicks on the first link of the second page, the position number will be 11

- [click] : like [noHits], it is a boolean value where '0' indicates that the user did not click on any result, and '1' indicates that the user has clicked on at least on one result.
- [history] : keeps track of the user behaviour. It represents a list of user "actions" : [user_action1,user_action2,...]
  A user action is formatted as follows: "query"///"facet_used"///"num_doc_found"///"query_time"///"num_of_page"///"url"///"url_position"
  - "query" : the query performed
  - "facet_used" : filled if the user has clicked on a facet, it is formatted like this : ("facet_field":"facet_value")
    For example, if you look at the second "user action" in the history of the example log : (extension:docx), you can deduce that the user has clicked on the facet "docx" which is based on the field named "extension")
  - "num_doc_found" : number of docs returned by the query
  - "query_time" : the query time measured by Solr
  - "num_of_page" : the number of the page where was the user
  - "url" : the URL of the clicked document
  - "url_position" : the position of the clicked document (first result = 1, second = 2, etc.). Remember that the position is absolute and based on all the available results.

  When the user clicks on a document, only two values of the history are set : the "url" and the "url_position"
- [url] : the clicked document URL. This value is only set if the log itself represents the click

Now that the basics are settled, let's decode the example log. Here is what it says :

- The log has been generated on a click of the user on a document (the [url] field is not empty)
- The original query was "engine", which returned 4 results and took 7 milliseconds
- The user has clicked on one document of which the URL is "file:/home/youp/Downloads/doc/Alertes.docx"
- Relying on the history, we can say that :
  - the original query was "engine", returned 4 results and took 7 milliseconds
  - then the user clicked on the facet "docx" which is based on the field named "extension". This "sub-query" returned 2 results and took 6 milliseconds
  - the last action (which is the log subject) is a click on the 2nd result of which the URL is "file:/home/youp/Downloads/doc/Alertes.docx"

By default, those logs are not displayed in the console, but are written into specifics log files.
The configuration of the log files (path, size, number etc.) can be set in the log4j properties located in tomcat/lib/log4j.properties.

Widgets and CSSs good practices

Datafari allows you to create your own widgets an CSSs and add them to enrich and customize the UI. The cost of such power is of course the possibility to break everything in the UI. So by reading carefully and following the few steps presented here, you will ensure an easy development, more compliant code for next Datafari updates, and fast rollback in case of problems:

- **Modify an existing AjaxFranceLabs widget for enhancement or bugfix** - On one hand, if you find an improvement or a bug fix on an existing widget, our recommandation is for you to apply your modifications directly on it and create a ticket on our Jira. On another hand, If you want to change the main behavior or main features of an existing widget, we recommand that you make your own and replace the original widget by yours in the searchView.jsp and search.js
- **Modify an existing CSS for enhancement or bugfix** - Same thing as the widgets: On one hand, if you find an improvement or a bug fix on an existing CSS, you can apply your modifications directly on it and create a ticket on our Jira. On another hand, if you need to change an existing CSS of Datafari, we recommand that you make your own copy and include it in the proper file. **A custom css already exists for the searchView.jsp**, it is available in {Datafari_Home}/tomcat/webapps/Datafari/css/custom/customSearchView.css
- **Create your own CSS per widget** - To make things clean and avoid mess, it is strongly recommended to have at least one separate CSS file per widget you made. Furthermore, there is a dedicated location for widget CSSs which is {Datafari_Home}/tomcat/webapps/Datafari/css/widgets
- **Make one file per widget** - If you develop multiple widgets, be sure to have each of them in a separate file. It is easier to maintain and to deactivate in case of problems.

Search Index

Datafari leverages Solr Cloud for its search capacities. Datafari utilizes 3 distributed Solr indexes. The main index, that contains all the crawled documents is FileShare.

FileShare index

We do not cover exhaustively the index fields and their configuration, as you can directly check them in the file schema.xml available here :

/opt/datafari/solr/solrcloud/FileShare/conf/schema.xml

Field names

Field Name	Description
title_fr	contains french title of a document
title_en	contains english title of a document
content_fr	contains french content of a document (the data is indexed and not stored in the index)
content_en	contains english content of a document (the data is indexed and not stored in the index)
content_hl	contains the content of the document (not depending on the language). This field is stored and not indexed and is used for highlighting. This field is truncated with the update processor TruncateFieldUpdateProcessor in the datafari update chain in solrconfig.xml
source	source of the data (for example, Web)
last_modified	last modified date of the document
extension	file format
allow_token*	These fields are used to store information on the access rights of the document
deny_token*	These fields are used to store information on the access rights of the document
suggest	Stores the terms used for autocompletion
spell	Stores the terms used of spellchecking
url	url of the document

Field types

FieldTypes for title_* and content_* are text_* (for example, text_en is the fieldType for title_en). This fieldtype contains specific analyzer for full text search on english text. Analysis phase is described in next section. FieldType for source is string. String FieldType doesn't contains analyzer and is used for example for facetting capabalities.

Data analysis at the indexing phase

As a reminder, a Lucene analysis chain holds a set of components able to tokenise and filter data in order to extract the terms to be stored in the index.There is one analysis chain per field. For each field, the analysis chains in Datafari have been optimized. For instance, for field content_lang1 and title_lang1 the field type text_lang1 includes components such as stemming that are specific to the language. Some other components have been added, such as the word_delimiter which allows to extract a file name from a URL. The LimitToken Filter is used to limit the number of terms indexed for each field of each document in order to be able to truncate very big documents before creating the inverted index. A big index can lead to poor search performances.

Modifying structure of the index

The structure of the index is defined by a schema. The schema template is stored here : /opt/datafari/solr/solrcloud/FileShare/conf/schema.xml. Then, the schema is loaded (at first start, or on demand with the updateSolrConfig.sh script) in the Zookeeper that holds all Solr configurations. When the core is loaded, a new file, called managed-schema is created in Zookeeper. This files is a copy of schema.xml and will reflect the modification pushed through the schema API. For example, the custom fields with the script addCustomSchemaInfo.sh (see dedicated section Custom Solr configuration).

Data Schemas

This section presents the way our data is modeled and used.

User data

Several types of user data are stored and managed by Datafari. It goes from user search history to user roles, including user preferences.

Likes and Favorites Schema

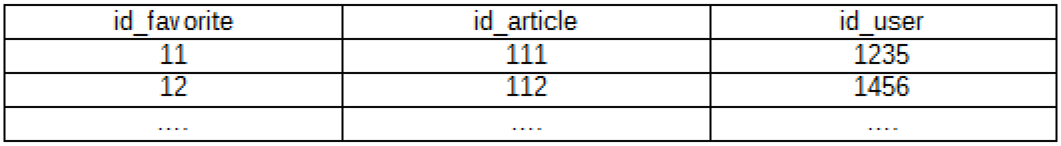
Problematic :

Starting with Datafari 2.0, users have the possibility to store pointers to documents shown in results list.

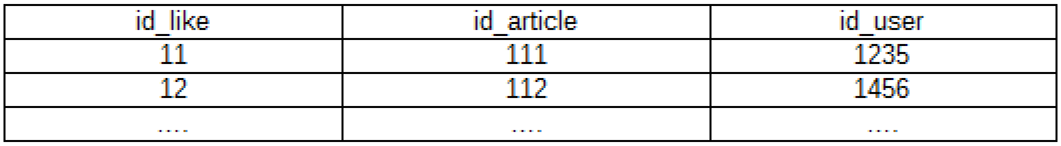
In the following document, “likes” corresponds to the functionality where a connected user can like or unlike a document, and “favorites” corresponds to the functionality where a connected user can save a pointer to a document he found in the results list.

Technically wise, we use Apache Cassandra for saving all the likes and favorites and Solr for saving the counters of likes of each document.

Using Cassandra, we cannot use standard relational db schemas, like the following:



id_favorite	id_article	id_user
11	111	1235
12	112	1456
....	....	....

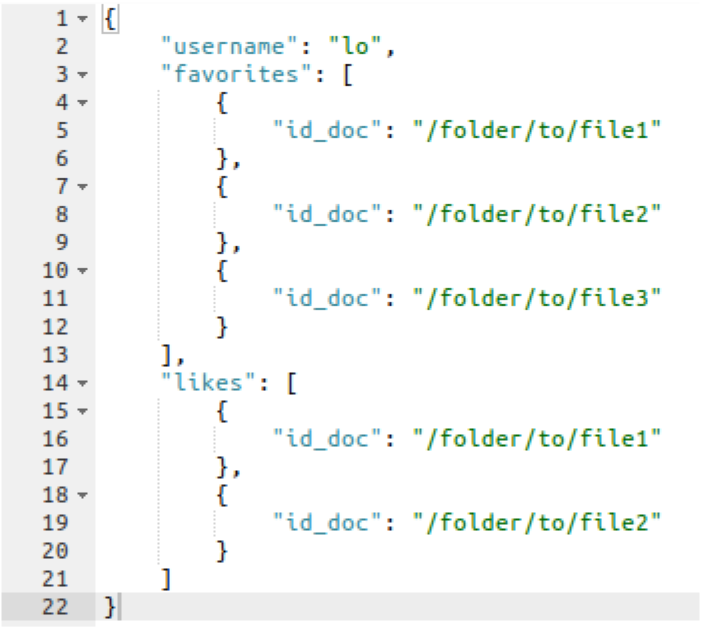


id_like	id_article	id_user
11	111	1235
12	112	1456
....	....	....

Modeling with Nosql in mind:

For our nosql modeling, we took our inspiration from this article http://docs.mongodb.org/manual/tutorial/aggregation-with-user-preference-data/ .

Applied to our problematic, we get the following :



```
{
    "username": "lo",
    "favorites": [
        {
            "id_doc": "/folder/to/file1"
        },
        {
            "id_doc": "/folder/to/file2"
        },
        {
            "id_doc": "/folder/to/file3"
        }
    ],
    "likes": [
        {
            "id_doc": "/folder/to/file1"
        },
        {
            "id_doc": "/folder/to/file2"
        }
    ]
}
```

That means we have dedicated to likes and favorites (which is distinct from the collection dedicated to user authentication and authorization).

Preferred language

Starting with Datafari 3.2.0, user's preferred language is now automatically saved and applied when user connects to Datafari or change the language.
The language info is stocked in the 'lang' table of the Cassandra database. This architecture follows the logic that was implemented by the 'Likes and Favorites' feature.

The 'lang' table is composed of two columns:

- username : *varchar* - it is the PRIMARY KEY of the table and represents the username returned by the Tomcat session.
- lang : *varchar* - represents the lang code that the user has selected ("en", "fr" ...)

The preferred language is saved/applied in several cases :

1) User connection

When a user successfully authenticates himself through the 'login.jsp' page, he is redirected by the jsp code to the servlet 'applyLang' (GET HTTP method). The servlet class is 'com.francelabs.datafari.servlets.ApplyUserLang'.
The servlet try to find a corresponding entry in the 'lang' table of the Cassandra database thanks to the username. Two possibilities :

- No entry is found in Cassandra : the servlet creates a new entry with the default selected language which will be the new preferred language of the user. After this, the servlet redirects to the admin page with the default language.
- An entry is found for the provided username : In that case the servlet redirects to the admin page and forces the language retrieved in Cassandra.

2) User changes the language

When a user changes the language of Datafari, an ajax call with a HTTP POST method is done to the 'applyLang' servlet with the selected language as POST data. The servlet simply creates/updates the username entry (if the user is authenticated in Datafari) with the provided language in the 'lang' table of Cassandra

3) Language keeper

The 'main.js' script has been updated to ensure that the 'lang' parameter provided by the url corresponds to the preferred user language.
To do so, an ajax call with a HTTP GET method is performed to the 'applyLang' servlet to retrieve the user preferred language. Two possibilities :

- The returned language corresponds to the one provided as a GET parameter : everything is OK, no further treatments.
- The returned language does not correspond to the one provided in the url as a GET parameter : the preferred language is forced through a new call to the 'applyLang' servlet which will redirect to the current page with the correct language.

Roles