Generic Search Service 2.2

Current released version

GSearch **2.5** (September 2012) is the currently released version.
Compatibility

Compatible with Fedora Version 3.1-3.3.

Table of Contents

About This Service

The Fedora Generic Search Service, abbreviated GSearch, is part of the Fedora Service Framework. It was developed by Gert Schmeltz Pedersen at the Technical University of Denmark, with feedback and contributions from members of the Fedora community, including Beth Kirschner, Binaya Poudyal, Blake Anderson, Boon Low, Christian Tønsberg, Eric Brown, Jun Yamog, Junran Lei, Leire Urcelay, Luis Zorita, Matt Zumwalt, Matthias Razum, Michael Appleby, Michael Hoppe, Nikolai Schwertner, Patrick Monbaron, Pierre-Yves Landron, Ranju Upadhyaya, Robert Sherratt, Ryan E. Scherle, Sam Liberman, Shunde Zhang, Steve DiDomenico Thierry Michel, and Xinjian Guo.

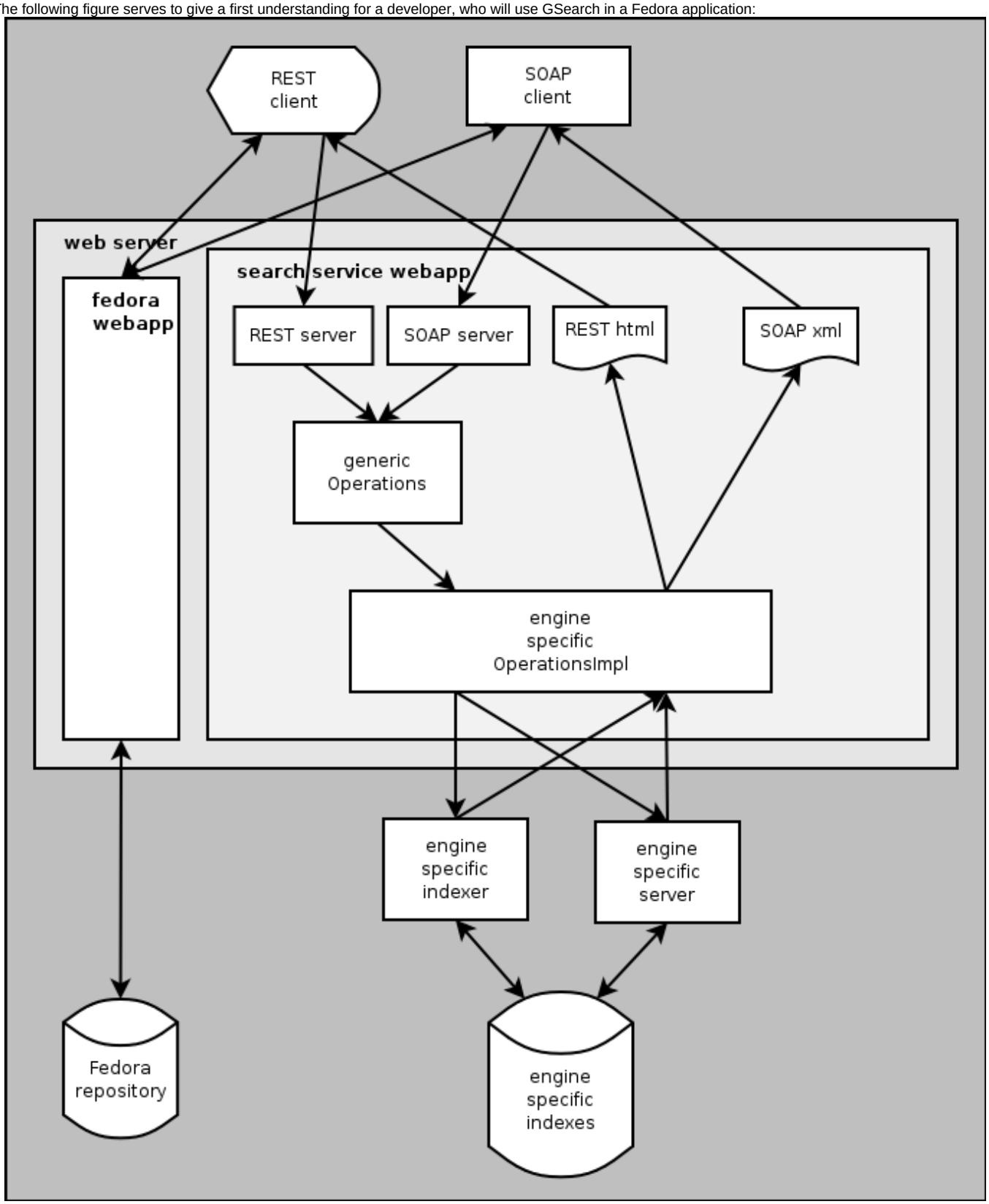
The work is funded by DEFF, Denmark's Electronic Research Library.

The service has the following major features:

- Indexing of Fedora FOXML records, including the text contents of Datastreams and the results of disseminator calls.
- Search in the index.
- Plugin of selected search engines, so far Lucene, Solr and Zebra.

The service has been developed and tested on Linux. This release targets only Linux installations. If you want to use the service on other platforms, you may be expert enough to do so, at least several people have succeeded on Windows and Mac platforms. You are encouraged to share problems and experience with the Fedora community, send mail to fedora-commons-users, or to Chris Wilper, or to Gert Schmeltz Pedersen.

The following figure serves to give a first understanding for a developer, who will use GSearch in a Fedora application:



The figure shows:

- A REST client, running in a user's browser, which may combine accesses to Fedora and to the Search Service.
- A SOAP client, running anywhere, may do the same.
- The Search Service implements a generic set of operations:
  - **updateIndex** - indexing the contents of the Fedora repository.

    - **gfindObjects** - search similar to Fedora findObjects and to the SRW/SRU operation **searchRetrieve**.
    - **browseIndex** - browsing terms in a given index, similar to the SRW/SRU operation **scan**.
    - **getRepositoryInfo** - describing the properties of a repository,
    - **getIndexInfo** - describing the properties of an index.
- Engine specific implementations of the operations will receive client requests, communicate with the engine indexer and search server, and return the responses in the appropriate form to the clients.

GSearch may run in a separate web server and may index more than one Fedora repository, and it may update more than one index in parallel. For further architectural details, see Additional Information

New Features in Version 2.2

- Fedora 3.1 compatibility
- Lucene 2.4.0 compatibility
- Solr 1.3.0 compatibility
- For the lucene plugin: Search result filtering by access constraints, as defined by XACML policies, in order to show only those search hits that the user is actually permitted to read. Read more...

New Features in Version 2.1.1

- Fedora 3.0 compatibility

New Features in Version 2.1

- Fedora 3.0b2 compatibility
- Added an update listener which uses the Fedora Messaging Client to listen for updates being performed through API-M. These update messages contain the information needed to perform index updates, thereby keeping GSearch up-to-date with the Fedora repository.
- Enhanced the `sortFields` parameter to `gfindObjects` for Lucene, sorting search results by a custom Comparator class, see the `index.properties` file in `configTestOnLucene` and the test class `dk.defxws.fedoragsearch.test.ComparatorSourceTest`.
- Enhanced the `fromFoxmlFiles` action of `updateIndex` for Lucene, so that all files are attempted to be indexed, even though one or more may fail, in which case log messages are given. Before, one failure would cause abortion.

New Features in Version 2.0

- Added a plugin for the Apache Solr search server.
- Added easier configuration, so that you need only edit one file with property values, then run it with ant.
- Updated to Lucene version 2.3.0.
- Added params to indexing in the format: ...&indexDocXslt=[xslt-name][(paramname1=value1[,paramname2=value2[,...]])] Use the parameters at indexing time by putting xsl:param statement in the indexing xslt stylesheet, like this: <xsl:param name="someparamname" select="defaultvalue"/>
- Added optimize options for Lucene indexing:
  fgsindex.mergeFactor and fgsindex.maxBufferedDocs will affect performance, see the index.properties file in configTestOnLucene. Also added ...?operation=updateIndex&action=optimize which will perform IndexWriter.optimize() which merges all segments together into a single segment, optimizing an index for search. Removed the optimize() call after each updateIndex.
- Added untokenizedFields property to Lucene index.properties files. Adding the property with a list of all untokenized fields will ensure that they all select the appropriate analyzer.
- Added a sortFields parameter to gfindObjects for Lucene, sorting search results as specified, see the index.properties file in configTestOnLucene.
- Added properties snippetBegin and snippetEnd, making highlight code configurable, see the index.properties file in configTestOnLucene.
- Added property for custom URIResolver used by xslt transformers for basic auth and SSL, see the example dk.defxws.fedoragsearch.server.URIResolverImpl class and the index.properties file in configTestOnLucene.
- Removed encoding of special characters in indexFields. Snippets now show special characters without modification. Indexes should be reindexed.

Installation

To install the service:

- Deploy `fedoragsearch.war` to the webapps directory of your web server, e.g. the Tomcat supplied with Fedora, or similar. You may rename the .war file, before you copy it into the webapps directory, in order to give it another webapp name.
- Edit the configuration settings.

The SOAP service operations are deployed with the .war file, and, when deployed, the .wsdl file is available at `services/FgsOperations?wsdl`.

Configuring the Service

- Edit the property values in the configvalues.xml file in .../webapps/<WEBAPPNAME>/ (where <WEBAPPNAME> by default is 'fedoragsearch'):
  o Set the property values for your environment.
  o Select the default config in configDefault.
  o Save this edited file outside of the web server.
  o Run target configOnWebServer after deployment from command line:
  >ant -f configvalues.xml configOnWebServer
  This will set your values into fedoragsearch.properties, repository.properties, and index.properties. Read these files to make sure they are correct.
- Then you may restart <WEBAPPNAME> and call http://<HOSTPORT>/<WEBAPPNAME>/rest in order to index and search. The name "rest" may be reconfigured in .../webapps/<WEBAPPNAME>/WEB-INF/web.xml

- Try the demo command line client. Change directory to .../webapps/<WEBAPPNAME>/client/ make the file executable, and run sh runRESTClient.sh then you will get the usage instruction.
- Tailor the demos for your own purpose by editing renamed copies of the demo xslt stylesheets in .../webapps/<WEBAPPNAME>/WEB-INF/classes/config/rest/ Then edit fedoragsearch.properties.
- Tailor the demo Lucene indexing stylesheet for your own purpose by editing a renamed copy of the demo xslt stylesheet in .../webapps/<WEBAPPNAME>/WEB-INF/classes/config/index/<INDEXNAME>/demoFoxmlToLucene.xslt For the sake of the example, the stylesheet indexes only active Fedora objects with PID starting with "demo" The options for tailoring include fields from other metadata datastreams than DC, field types and field boosts, see the stylesheet for options. Then edit index.properties.
- For your real applications, you must carefully edit all stylesheets for your purpose.

- Inspect the Lucene index with Luke. Notice, Luke cannot open an empty Lucene index.
- You may tailor the highlight of search terms in demo.css.
- You may want to experiment with more than one configuration, in which case you may maintain them under different names in parallel to the "config" configuration, which is the default configuration. In order to activate an alternative configuration you may use the semi-secret operation configure with parameter configName, either using the demo command line client or the REST interface.

Alternate Web Application Context

If you are configuring GSearch to work with a Fedora 3.2+ server and you have elected to use a web application context other than 'fedora' (i.e. the URL to your Fedora server is not http://host:port/fedora

) you will need to edit the repository.properties file to update the context in the repository URL. You will also need to update any links provided in XSLT files, such as in basicGfindObjectToHtml.xslt.

Configuring the service for Automatic Updates

As of version 2.1, GSearch has the ability to listen to update messages provided by Fedora. These messages are sent via JMS, so a JMS provider must be available (a JMS provider is included with Fedora 3.0). In order to configure the update listener, open updater.properties and set the following property values. These values will most likely be the same as those specified in your Fedora configuration.

- **java.naming.factory.initial**
  - Default: org.apache.activemq.jndi.ActiveMQInitialContextFactory
  - Specifies the JNDI initial context which will be used to look up JMS administered objects.
- **java.naming.provider.url**
  - Default: tcp://localhost:61616
  - Specifies the address at which a connection can be made to the messaging provider.
  - The update listener will attempt to connect to the messaging provider at this address on server startup, so make sure that your provider is running and available, otherwise you will see a connection error.
- **connection.factory.name**
  - Default: ConnectionFactory
  - Specifies the JNDI name of the ConnectionFactory object needed to create a connection to the JMS provider.
- **topic.fedoraAPIM**
  - Default: fedora.apim.update
  - Specifies the topic on which to listen for updates.
- **client.id**
  - Default: fedoragsearch0
  - The identifier of the GSearch client. If you have more than one instance of GSearch running they must have different client identifiers.

If you decide not to use the automatic updates feature in GSearch, you'll need to open fedoragsearch.properties and remove (or comment out) the line specifying fedoragsearch.updaternames. This will disable the update listener.

Configuring Fedora for Automatic Updates

Fedora 3.0 added the ability to send a message whenever a change is made to the content of the repository (through API-M.) This messaging capability must be enabled and configured to work properly. See the Fedora documentation for instructions on configuring messaging.

As an alternative to updates via messaging, it is possible to configure Fedora to send a signal via REST to the Generic Search Service when objects are added, modified, and purged. Using messaging is the preferred method for automatic updates, and this technique, while still available, should be considered deprecated. It it not recommended to use both the update listener and REST-based updates.

To enable REST-based updates, edit your fedora.fcfg file and change the class of the fedora.server.storage.DOManager module to fedora.server.storage.GSearchDOManager. Then populate the following module parameters as needed:

- gSearchRESTURL - The REST endpoint for GSearch, for example, http://localhost:8080/fedoragsearch/rest

- gSearchUsername - If GSearch is protected by authentication, this is the username that Fedora should use to authenticate.
- gSearchPassword - The password for the above user, if applicable

Additional Information

Search Engine Plugins
Architectural Snapshots
Multilingual Configuration

Search Engine Plugins

Lucene

The Lucene plugin comes as the java package dk.defxws.fgslucene together with the Apache Lucene java libraries. The set of classes was inspired by the JDBC API specification.

The Lucene plugin is used by configuration as explained above, see also the DemoOnLucene example.

Lucene has a very rich functionality, and this plugin exploits a small fraction of it. As a java programmer, you may have ideas for further exploitation, which you may realize by implementing an enhanced version of the plugin. Please, share such ideas and implementations with the Fedora community.

Solr

The Solr plugin comes as the java package dk.defxws.fgssolr.

The Solr plugin is used by configuration as explained above, see also the DemoOnSolr example.

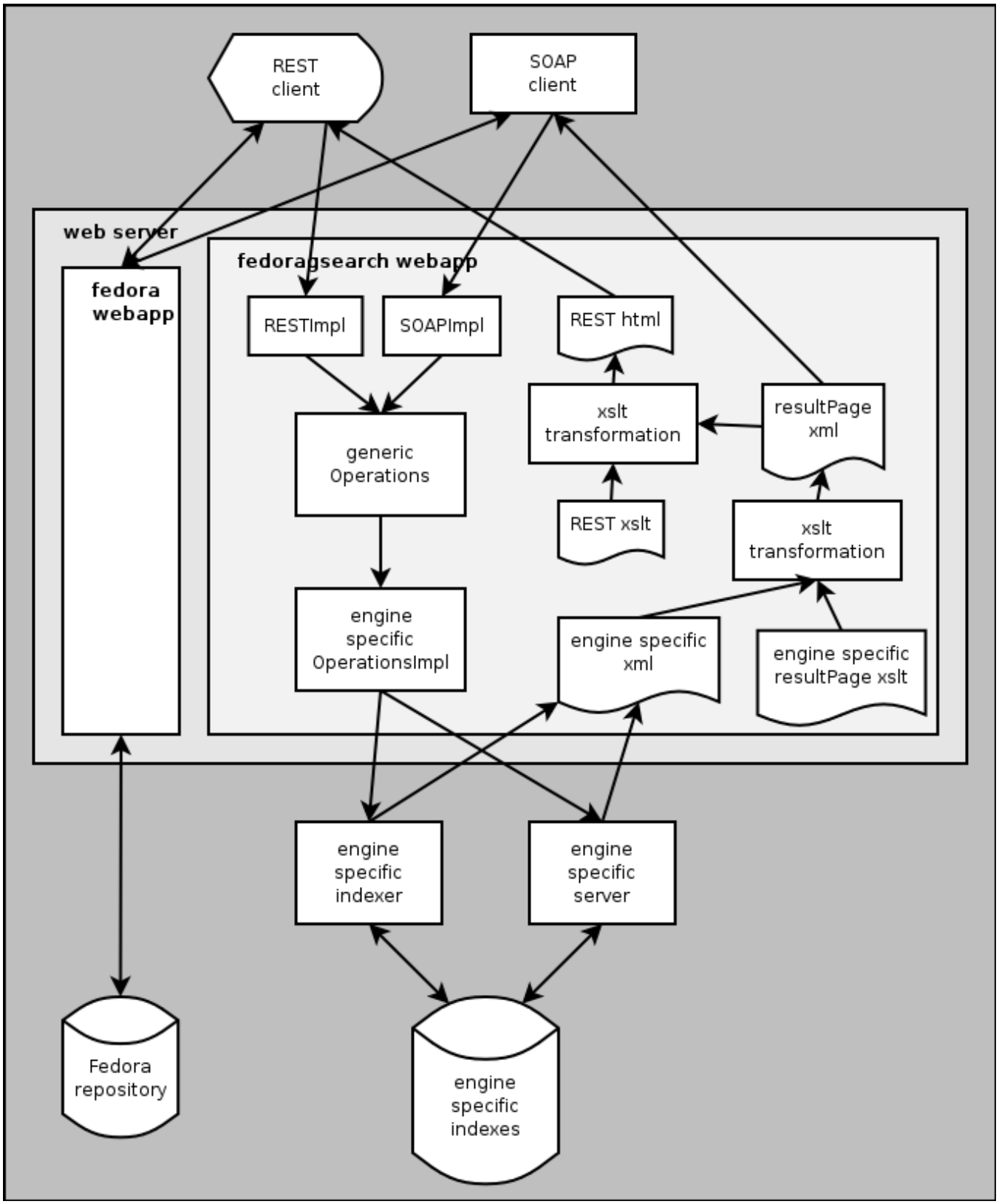
This plugin indexes documents via the HTTP POST interface of Solr. Searches may be performed via the Solr native HTTP GET to the Solr server and via gfindObjects, which accesses the Lucene index directly. Solr functionality does not include browsing, however, this is offered by the plugin via the browseIndex operation.

Zebra

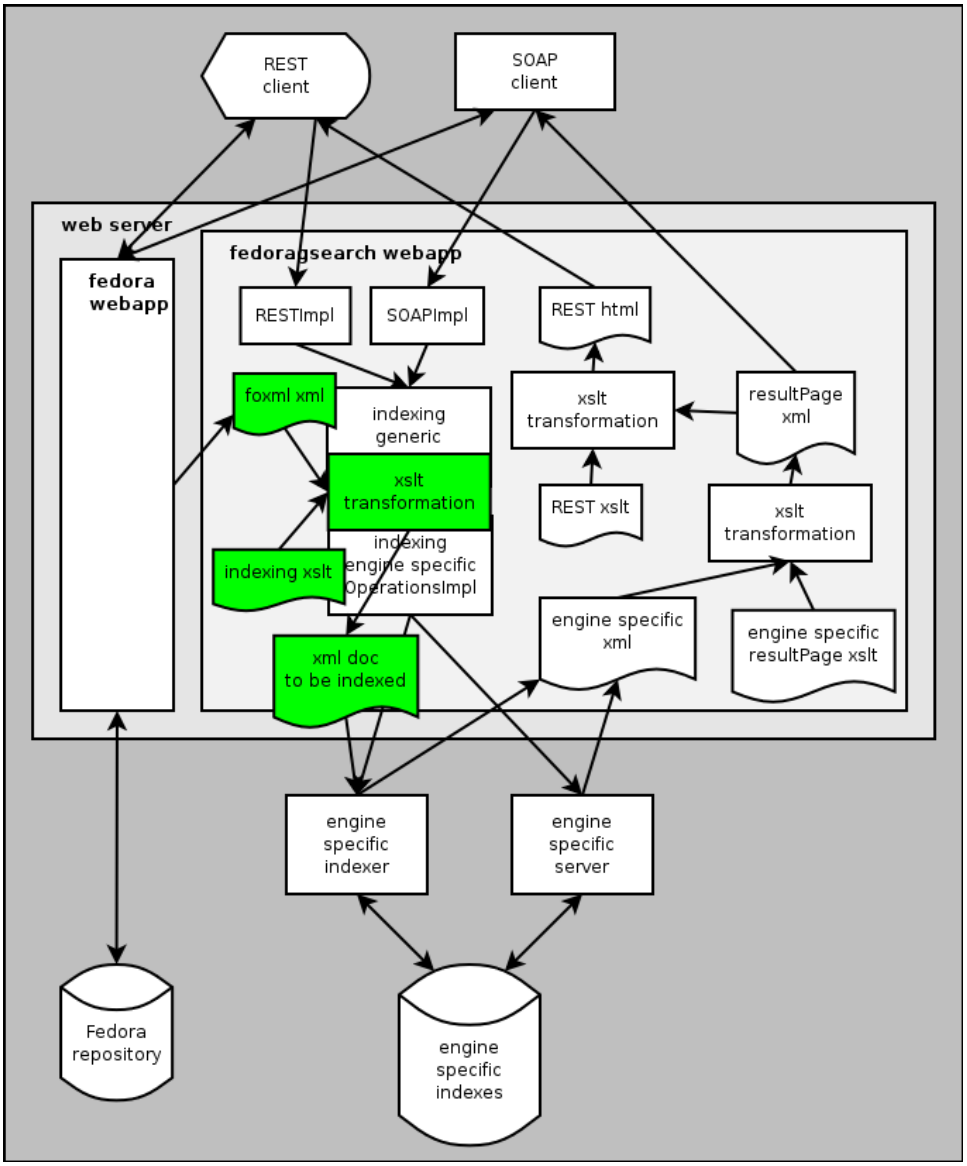
The Zebra plugin comes as the java package `dk.defxws.fgszebra`.

The Zebra plugin is used by configuration as seen from the DemoOnZebra example, which includes a README file, which explains how to get and install Zebra, and how to configure it.

Architectural Snapshots

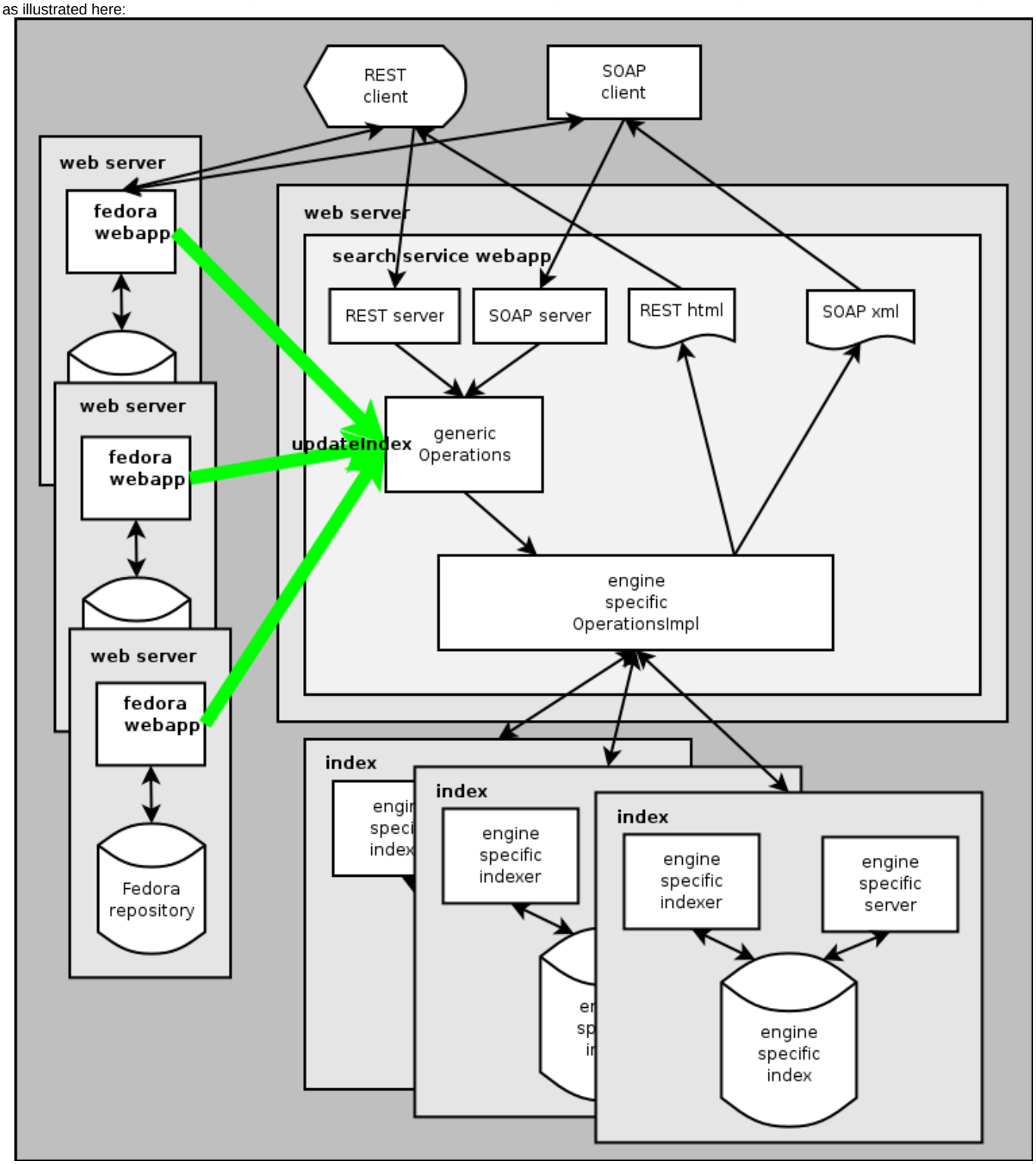


- All engine specific operations return an engine specific xml answer, which is transformed by an engine-specific xslt stylesheet into result page xml. For a SOAP request this is the answer. For a REST request this is transformed to an html answer. There may be any number of xslt stylesheets to select from, the default ones are selected in the properties file. Selecting a copy stylesheet will allow the transfer of an answer untransformed. An alternative result page format is OpenSearch, which is an RSS2.0 extension.
- Parameters allow clients to select repository, index, and xslt stylesheets by name. In a real application, these values may be determined by the developer in the code, or by the administrator in the properties file.



- Objects in the Fedora repository are exported in FOXML format, transformed into an appropriate document format by the indexing stylesheet, and indexed by the engine in question. The XML datastreams are indexed as decided in the stylesheet. One managed or external datastream may be indexed per FedoraObject (which one is configurable), assuming that they contain the same text in different mimetypes.
- The following updateIndex actions are available:
  - **createEmpty** - creating or emptying the index. For a new index, you have to run createEmpty once, before you can run the other actions.
  - **fromFoxmlFiles (filePath)** - indexing FOXML records; filePath may be null, in which case the configured Fedora Object Directory is used, so that the whole of the Fedora registry is indexed.
  - **fromPid (PID)** - indexing one FOXML record, as exported by Fedora API-M; in case a previous index document with the same PID exists, it is first deleted. This is the incremental update operation that shall be called after all of Fedora's API-M operations that modifies a FedoraObject.
  - **deletePid (PID)** - deleting one index document.

A typical application will index one repository in one index. However, you have the possibility to index many repositories in one or more indexes in parallel, as illustrated here:



- There are OperationsImpl classes for Zebra, Lucene and Solr. The configManyToMany example has indexes for two engines, therefore similar searches may be compared.

Multilingual Configuration

Luis Zorita had this problem and solved it (his mail to fedora-users on 24 August 2006):

Hello Gert: I have solved this multilingual problem adding the attribute URIEncoding="UTF-8" to .../tomcat/conf/server.xml and to .../tomcat/conf /server_fedoraTemplate.xml Now I can search special Spanish characters like "ñ", "í" etc. with fedoragsearch. Luis

Search Result Filtering

Search result filtering by access constraints, as defined by XACML policies, will show only those search hits that the user is actually permitted to read. Three solutions have been investigated and demonstrated and presented here. Besides, the demonstration is included with the GSearch distribution in ... /WEB-INF/classes/configDemoSearchResultFiltering/ . In brief, the three solutions are:

- **Post-search filtering**, which requires a request to the XACML mechanism for each hit, and the total number of permitted hits is only known at the end, a costly procedure especially when few hits are permitted out of a large number.
- **In-search filtering**, which requires additional index fields and query rewriting, that is, a logical partitioning of the index.
- **Pre-search filtering**, which requires a physical partitioning of the index and selection of the pertinent index at query time.

Both in-search and pre-search filtering face the challenge of exact correspondence between the filtering mechanism and the XACML policies.

For your own purpose, in fedoragsearch.properties, you have to select the preferred searchResultFilteringType and set the searchResultFilteringModule to a class that you have to program, as a subclass of the demo class dk.defxws.fedoragsearch.server.SearchResultFilteringDemoImpl or as an implementation of the interface dk.defxws.fedoragsearch.server.SearchResultFiltering .

Last Modified by Gert Schmeltz Pedersen