Space Details

Key:	SESAT
Name:	Sesat
Description:	Project pages for Sesat - SEsam Search Application Toolkit
Creator (Creation Date):	kentv (Feb 26, 2007)
Last Modifier (Mod. Date):	mick (Nov 17, 2007)

Available Pages

- Home
  - Apache
  - download
- Community
  - Getting Help
  - Guide Helping
  - Kernel Roadmap
  - Team List
- External
  - External - FAST
  - External - Integration Platforms
- Features
  - Kernel Feature list
  - Product Description
    - Product Description - Business Managers & Business Developers
  - Product Whitepaper
  - Showcases - Sesam.no
  - Technical Feature Matrix
- Docs + Support
  - Architecture Overview
    - Design Proposals
      - New design proposal for SearchCommand and AbstractSearchCommand
        - QueryBuilder code example
        - SearchCommandParameter code example
        - Stream manipulation code examples
    - Personalization architecture
  - Debugging
    - Debugging Velocity Templates
    - Logging, Logfiles, and Statistics
    - Sesat-Interpreter
  - Development Guidelines
    - Building with FAST
    - Kernel Operations
    - Requirements

- Development platform architecture
  - Hardware Requirements
  - Software Requirements
- Development using the APIs
- Development using the SFC
  - Developing a quick and simple Enrichment tutorial
  - Developing using CSS
  - HOWTO develop a RunHandler
  - Navigation documentation
  - Search modes schema generator
  - Tutorial - Ajax examples
  - Tutorial - Building Sesam.com
- faq
- HowTo Solr Query Evaluation
- Upgrade Guides
  - 2.16 Upgrade
  - 2.17 Upgrade
  - 2.18 Upgrade
  - Writing Upgrade Guide guidelines
- About
  - Free Software License
  - Project
  - Propriety License
  - Sitemap

This page last changed on Sep 26, 2008 by mick.

Build your own federated search engine!

Sesat is an open sourced Search Middleware with federation capabilities and a built-in search portal framework. Sesat enables a single user query to be dispatched to multiple information sources. The result is analysed, weighted and presented to the user according to configurable business rules.

- Sesat is an acronym for "SEsam Search Application Toolkit" and is the core technology used to power http://sesam.no and http://sesam.se, which are scandinavian search, news and directory sites that utilise a large number of data sources including Yahoo!, PicSearch, Solr (Lucene), Youtube, and enterprise search systems from FAST.
- Sesat makes it easy to build applications that look for information in many different places simultaneously. Sesat can connect to almost any kind of data source that can be accessed using Java - databases, search indexes, files, back office systems, web services, ESBs. Similar product examples are FAST Unity, WebFeat, DeepWeb Explorit, dbWIZ, and Raritan SIFT.
- Sesat takes care of all the complex tasks of communicating with multiple search indexes simultaneously, query- and result analysis, business rules application; leaving the developers to focus on other aspects of their application, such as presentation and usability
- Sesat is developed in an open environment and released under the Affero General Public License. We invite you to participate in this open development project. To learn more about getting involved, click here.

Showcases

Site	Screenshots	Description
sesam.no		Is the portal that initiated the development of Sesat. The presentation layer is made by front end developers using the Velocity templating language. Each search on sesam.no results in several parallel searches in different data sources. When all search results have been collected, Sesat invokes the relevant templates and displays the result to the user. Sesam.no uses technology from Yahoo!, FAST and PicSearch,. Sesam.no was launched november 2005.

vg.sesam.no		VG is the largest newspaper in Norway, hosting Norways most popular web services. By using Sesat, Sesam.no has created both a tv-guide and a news service for VG, all running on Sesam.no's platform, but with VG look-and-feel.

News

Title	Author	Date Posted
Lucene and Solr patches accepted	Mck Semb Wever	Jan 31, 2009
Sesat 2.18 - "The acquisition of wealth is no longer the driving force in our lives."	Mck Semb Wever	Dec 03, 2008
Sesat 2.18 - "The acquisition of wealth is no longer the driving force in our lives."	Mck Semb Wever	Nov 20, 2008
federatedsearchblog.com - "New open source federated search middleware released"	Mck Semb Wever	Jul 09, 2008
Sesat on Sourceforge	Mck Semb Wever	Apr 28, 2008
Sesat on Freshmeat	Mck Semb Wever	Apr 28, 2008
Sesat 2.17 - "Are we human because we gaze at the stars or do we gaze at the stars because we are human?"	Mck Semb Wever	Apr 22, 2008
Sesat 2.17 - "Are we human because we gaze at the stars or do we gaze at the stars because we are human?"	Mck Semb Wever	Apr 08, 2008
jaxmag.com - Search Middleware Portal Generally Available To Access Data Source	Mck Semb Wever	Apr 03, 2008
Sesat, a federated search solution middleware, goes open source	Mck Semb Wever	Mar 31, 2008
Cominvent - Norweigan search portal Sesam.no releases middleware as GPL	Mck Semb Wever	Mar 30, 2008
DIGG - Build your own search engine with SESAT	Mck Semb Wever	Mar 30, 2008
Sesambloggen - Sesam Search Application Toolkit (SESAT) available as Free Software	Mck Semb Wever	Mar 21, 2008
Digi.no - Sesam deler ut kildekoden	Mck Semb Wever	Mar 21, 2008

Apache

This page last changed on Mar 06, 2009 by sshafroi.

To have tomcat and jboss to play together we need apache with rewrite and proxy enables.

First you need this modules in apache.

rewrite_module

proxy_module

proxy_http_module

On ubuntu you can enable this with:

```
 sudo a2enmod proxy
sudo a2enmod proxy_http
 sudo a2enmod rewrite
```

You need to allowoveride -> all

```
DocumentRoot /var/www/
<Directory />
Options FollowSymLinks
AllowOverride All
</Directory>
```

and under virtualHost, add

```
ProxyPreserveHost on
```

In your /var/www/ put .htaccess with this:

```
RewriteEngine On

RewriteRule ^useradmin/(.*)$ http://localhost.no:9090/useradmin/$1 [P]
RewriteRule ^(.*)$           http://localhost.no:8080/$1 [P,L]
```

download

This page last changed on Mar 30, 2008 by mick.

Accessing the source code repository

Access the source code repository for this project in one of following ways:

- Browse source code online at http://sesat.no/svn to view this project's directory structure and files.
- Check out source code with a Subversion client using the following command.

  ```
  svn checkout http://sesat.no/svn/<project-name> <project-name>
  ```

  If you are new to Subversion, you may want to visit the Subversion Project website and/or read Version Control with Subversion.

Downloading binaries

Binaries can be downloaded directed from our maven2 repository.

Community

This page last changed on May 12, 2008 by mick.

work in progress

Coding	
Getting Started	Building Sesat-kernel from sources.
Building a skin	Building your first Sesat skin (Sesat Frontend Container).
Communication	
Getting Help	Questions and answers on our mailing lists.
Guide Helping	How to help out.
Team List	Current developers.
Roadmaps	
Kernel Roadmap	Where the Kernel is headed for...

This page last changed on May 06, 2008 by mick.

Overview of our mailman mailing lists

Normal mailing lists

List	Description	Links
Commons-development	Commons Use + Development	Browse Subscribe gmane archive
Kernel-development	Kernel Use + Development. **(start here if unsure)**	Browse Subscribe gmane archive
Sesat-discuss	General discussions around the website, license, and other non-code stuff	Browse Subscribe gmane archive

Read only mailing lists

Project	Commit lists	Build lists	Issue lists
Commons	Browse Subscribe gmane archive	Browse Subscribe gmane archive	todo
Kernel	Browse Subscribe gmane archive	Browse Subscribe gmane archive	Browse Subscribe gmane archive

This page last changed on Mar 23, 2008 by mick.

How to Contribute

1. Ask questions and give answers on our mailing lists,
2. Check out and build the Kernel and Sesam.com skin,
3. Enter bugs and improvements into our issue tracker (pending to bring issue tracker live).

Kernel Roadmap

This page last changed on Sep 26, 2008 by mick.

In an attempt to formally define what the Sesat Kernel open source project is trying to achieve, and how we plan to achieve it, we've come up with the Kernel Roadmap.

Please bear in mind that as with any good open source project all decisions are consensus-based by the community. Today this community consists of Schibsted Søk AS and Schibsted Sök AB.
Feel free to contribute to these priorities on kernel-development@sesat.no
Upgrade Guides exist between each version.

Version	Expected Features
2.x	***Sesat Kernel concepts still under development***
2.17	* ~~SKER4290 Design and code with JSPs in skins~~ * ~~SKER3654 Generalized enrichment handling~~ * ~~SKER3733 New methods for offset~~ * ~~SKER4302 Building Sesam.com tutorial~~
2.18	* ~~SKER2146 modes.dtd generator~~ * SKER2149 Divide & Conquer AbstractSearchCommand to delegates * ~~SKER3540 Anonymous TokenPredicates & Token Evaluator SPI~~ * SKER2273 SyndicationGenerator to work for any tab * ~~SKER4949 Solr (Lucene) search command~~ ~~* Yahoo "Contextual Web Search" search command~~
3.0	***Sesat Kernel original goals met, all Service Provider Interfaces defined.*** * SKER1757 SESAT Kernel * SKER4182 Sesat-ise and standardise decorators
3.1	***Second phase of goals from Sesat Kernel's original specification.*** * SKER1793 XML API (SESAT API) * SKER2163 SESAT Testing Environment * SKER1609 Immutability and the flyweight pattern within the DataModel * Swing integration and tutorial * Seam integration and tutorial
?	***Ideas we'd like to see and think easily feasible.*** * Google search command * Sharepoint integration * Compass integration??

Team List

This page last changed on Apr 17, 2008 by mick.

Member	Profile page
Active	
Anders Berneby	
Anna Larsson	
Bernt Rostad	
Endre Midtgård Meckelborg	
Kristian Saebdal	
Håvard Frøiland	
Magne Thyrhaug	
Magnus Lambert	
Michael Semb Wever	http://wever.org/
Thomas Kjærstad	
Past	
Anders Johan Jamtli	http://www.jamtli.no/
Kent Vilhelmsen	
Magnus Eklund	
Ola Marius Hoff Sagli	

External

This page last changed on Jan 25, 2008 by mick.

External - FAST

This page last changed on Jan 25, 2008 by mick.

This page last changed on Jan 25, 2008 by mick.

Integration Platforms

There are several suitable integratino platform available. The following lists some of the more common. Note that a later version of FAST **will** include its own ETL-tool for easy feeds of documents into the index.

System	Homepage	Description	License
BODI	http://www.businessobjects.com	Business Objects Data Integrator	Commercial, expensive
Mule	http://mule.codehaus.org	Mule - ESB glue / ETL tool. Links: • http://www.itmanagersjournal.com/feature/22422	Open Source
Apatar	http://www.apatar.com/	Data integration software. Connectivity to Oracle, MS SQL, MySQL, Sybase, DB2, MS Access, PostreSQL, XML, InstantDB, Paradox, BorlandJDataStore, Csv, MS Excel, Qed, HSQL, Compiere ERP, SalesForce.Com, SugarCRM, Goldmine, any JDBC data sources and more.	Open Source

Features

This page last changed on Jan 25, 2008 by mick.

Kernel Feature list

This page last changed on Sep 26, 2008 by mick.

> Work In Progress

The SESAT Kernel, the core to the search front, and the generic.sesam skin (SFC) serves as a search engine's controlling and presentation framework.

It provides developers will the following features:

- A **modularised model-view-control architecture** specialised for a data fetch and present paradigm aka a search engine. Each layer forms a separate maven-2 built library project, fully extendable by defined Service Provider Interfaces (SPIs). Dependencies through each layer are supported with a Contextual Inversion of Control implementation. Default implementation of these SPIs exist within the generic.sesam skin (SFC), and reference implementations found in the genericno.sesam.no & genericse.sesam.se skins (SFCs).
- A **Search Command SPI**, also comprising of individual for SPIs Query Transformations, Search Executions, and Result Handling. Provides sequential and parallel thread execution for maximum performance.
- Default **Query Transformer implementations** for Age filtering, Exact field matching, Regular Expression transformations, Synonym transformations, Term prefixing, and Query-Matching Token (un)masking.
- Default **Result Handler implementations** for Age calculations, Modifier combining, Date formatting, Detect/Find File/Url formats, Mathematical & Scientific number operations, Phone number formatting, Spelling Suggestion formatting.
- Default **Search Command implementations** for Fast-4 Simple search, Fast-4 Advanced search, Fast-5 ESP search, generic XML search, Web Services seach, Blending (FAST) search, Clustering (FAST) search, Correcting (FAST) search, Mathematical/Calculator search, Danish Mobile search, Overture PPC search, PicSearch search, Platefood PPC search, Sensis search, Stock market search, Yahoo IDP search, Yahoo Contextual Web Service search, Yahoo Media search, and Youtube search.
- Reference Search Command implementations for Blinkx Video search, Blocket search, Finn search, GeoData Map search, HittaMap search, Hitta search, HittaWeather search, Prisjakt search, Solr (Lucene), Storm Weather Blinkx Video search, Tasteline search, TV-guide (FAST) search, Whitepages (FAST) search, Yellowpages (FAST) search.
- **DataModel infrastructure** holding definitions and access to all data required through the process stack and in the presentation layer. The DataModel provides compile-time type safety to the data, an interface for presentation pluggable renderers, templating simplicity, client and developer security, scalability through immutability and re-use of objects, and ajax (remoting) interaction.
- The DataModel contains (**Data Access Objects**) nodes for http-requests, http-sessions, users and locations, skins (SFCs), Queries, Pages and verticals, and Searches (results, navigators, enrichments, advertisements, rss).
- The DataModel extends and supports the **JavaBean specification**, each node a bean with associated BeanInfos.
- **Query parser and tree** implementation. Based on javacc. Provides a superior approach over string manipulation from the user's inputted query through the query transformations and into the search commands. The resulting Query tree is manipulated through the remainder of the process stack with visitor patterns. Each node (token or operator) permits storage of meta-data and becomes immutable providing better than linear scalability.
- **Query parser supports tokens** for words, numbers, phone numbers, urls, email addresses, and phrases.
- **Query parser supports operators** of and, or, xor, and default hidden operator.
- Default **visitor pattern implementations** for counting nodes, finding nodes, finding positions within the tree, finding forests of like nodes, finding parents, searching meta-data, name and location separation and extraction.
- Default **Query tree alternations** for Fullname detection, and Tree rotations.
- **Presentation framework** for quick and flexible visual designs for various formats (html, xml, rss, vcard, etc). Organisation and inheritence/overload for configuration (properties & XML), CSS, javascript, and image resources. Templating (layout and decoration) system providing ajax support, search-engine specific macros and directives, link and click encodings and statistics, publishing system (static html or xml URLs inclusion) integration, and http protocol independence. The presentation layer renders off the java beans provided by the datamodel simplifying the templates and securing through isolation the view from the control and model layers.
- **Apache Velocity** is fully supported as "templates" in the presentation framework.

- **JavaServer Pages** are fully supported as "templates" in the presentation framework.
- An **http protocol layer** handling and encapsulating sessions, encodings (including broken clients), pretty URL variants and URL rewriting (like apache's mod_rewrite), user login persistence, parameter encryption, and resource caching.
- Custom **HttpClient library**, providing basic Quality-of-Service and statistics of outbound connections. Backgrounds and buffers outbound connections to avoid thread starvation or locking, and protect against misbehaving indexes.

Error formatting macro: toc: java.lang.NullPointerException

Product Description

Definitions of Terms

Term	Definition
SESAT SFC - Search Frontend Container	This is where all User Applications using SESAT are run.
SESAT SFC User Application (UA)	Auser-provided search application/portal with a unique vhost
SESAT SFC Domain	Root domain for vhost applications. Example: «sesam.no» is a root domain, whereas «sports.sesam.no» is a unique vhost belonging to a SFC User Application.
Magical Words	Special trigger keywords that invokes special ranking or index selection rules. Example: «weather singapore» may trigger a weather vertical to be displayed, or indicate that the weather search result should be displayed first in the result set.

Product components

- The SESAT Core application
- The SESAT Search Frontend Container (SFC)
- The SESAT Admin framework
- The SESAT API

These components are described in the following sections.

SESAT Core

SESAT Core runs as a web application within a Tomcat Web Server. All access to the search portal is handled by this application.
All access is through a recommended maximum number of vhosts (50), each of which specify which SFC user application is accessed.
A rules engine provide functionality to handle a wide variety of ranking models, ranking dependencies, magical words and analysis rules.
For a full feature list of what SESAT Core provides see http://sesat.no/Kernel+Feature+list

SESAT SFC (Search Frontend Container)

Deployed in the same directory and on the same Tomcat Web Server as the SESAT Core, all SFC User Applications are web archives utilising the services of the SESAT Core. SFC User Applications belong to a domain hierarchy, with configuration inheritance applied by default.

To define a User Application in the SFC, the following must be provided:

Element	Description
vhost-name	(for example, «vertical.mydomain.com», «mydomain.com» «modes.xml» configuration file, which specify: available modes, where a «mode» is defined as a portal tab or vertical. For example, «vg.sesam.no» may have two modes, one for «search within vg.no content only», and one for «search whole of norwegian internet». All modes within a User Application (vhost) have a unique name, that is later referred to from views.xml. A mode define which indexes should be used for a given tab or vertical. For example, the «search within vg.no content only» may also perform commandoes to get information from a PPC-system, from a yellow pages directory and from a stock market information index.
views.xml	configuration file, which specify: • For a given view or tab, which «mode» is used • For a given view or tab, which enrichments applied (enrichment access methods are usually listed in the modes.xml configuration) and how they should be applied/weighed. For example, for a financial newspaper, enrichments may be yellow pages, stock information and news. These enrichments are then ranked according to rules in the views.xml configuration. • Any ad-commands that should be applied to the search result
Images and templates	Optionally, the User Application can provide own images and templates. If these are not provided, the parent templates are used. For example, «vg.sesam.no» will by default inherit all images and templates from «vg.no», unless the application has defined its own.

SFC Directory Layout

From a *developers* perspective, the SFC are layed out in the following manner:

```
root-directory: domain.com/
sitesearch.domain.com/
 LICENSE.txt
 logs
 pom.xml
 war/
  src/
   conf
   css
   images
   javascript
   templates/
    defaultSearch.vm
    VM_site_library.vm
    enrichments/
```

fragments/
navigators/
pages/

From a *deployers* perspective, the SFC is layed out in the following manner:
webapp/
/ROOT.war (<-- this is sesat)
/generic.sesam.war (<-- this is sesat)
/sitesearch.domain.com.war
/sitesearch2.domain.com.war

SESAT Admin

SESAT Admin consists of portlets and applications running on JBoss Portals (server), which is a standards-compliant (JSR-168) portal. As of version 1.0 of SESAT, the following appliacions are available:

- User Administration Portlet (require usage of OpenLDAP). The administrator can define users, groups and roles, and ensure that access to other administrative systems or the search portal are prohibited and controlled.
- Statistics Portlet, providing simple statistics of queries and traffic.
- FAST Lists Editor (version 1.0 of SESAT). FAST Lists are lookup-tables within the FAST engine that are used when analysing search queries.
- Rules Engine Portlet (version 1.2 of SESAT)
- Personalisation Manager (version 1.2 of SESAT)

SESAT API

SESAT provide two APIs for communication with the SESAT Core.

The APIs are available as

- WS - Web Service.
- XML - XML feed API

Access control etc. is controlled through access control lists (ACLs).

	**Note** Link to more documentation about the API. This could also be documented in Continuum.

This page last changed on Aug 22, 2008 by mick.

Error formatting macro: toc: java.lang.NullPointerException

Product Description - Business Managers & Business Developers

With Sesat, your IT department can quickly merge content from a huge variety of different data sources, in line with your business requirements.

Executive Summary

Sesat is a federating search portal middleware, enabling developers to easy collect information from a variety of different sources and data stores, and present them in a configurable, intelligent and flexible way to the user.

Sesat includes a set of applications working together to provide a feature-rich search middleware. **Sesat**glues together information from underlying indexes and data sources, applies the relevant business logic, and passes the resulting collection of information on to the presentation layer. The toolkit has excellent support for all recent versions of the FAST search engine, as well as many other data sources. The APIs provide full flexibility for customers who easily want to create own connectors to underlying data layers and sources.

Sesat glues together information from multiple data sources and presents it uniformly to the developer, who now only need to concentrate on the presentation of the data. **Sesat** makes it very easy to develop, deploy and manage search portal applications.
Components used in **Sesat** are all Open Source, and the code to **Sesat** itself is freely available to the developers.

Sesat will:

- reduce development complexity
- enable a high degree of flexibility for the business
- lower total cost of ownership
- deliver superior value

What is it?

- **Sesat** is a search application toolkit
- **Sesat** is a search application framework
- **Sesat** is a search application tracking system

With the new Boomerang module it is easy to follow the users behavior on the site

Key Points

- Users of **Sesat** value the flexibility of not being tied to any particular search engine vendor.
- **Sesat**is not tied to any particular search engine or data storage. The customer may easily provide his own search interfaces and make them available to the **Sesat** Kernel.
- **Sesat** uses unicode (UTF-8) throughout, and is as such compatible with most languages and character sets.
- **Sesat**makes it possible to create search portals and search applications fast without worrying about internals of search engine APIs or writing display code, business logic etc. The developer's focus is entirely on the portal's functionality.
- **Sesat** is a mature search application framework that has been under heavy development since autumn 2005. It has an exiting roadmap which includes personalization, session-handling, Ajax-capabilities, rules engines, statistics tools and administrative tools.
- **Sesat** is an open architecture. The customer have access to all source code, and can submit code to the project.

- **Sesat** uses Open Source applications and libraries. No commercial software needs to be installed to run **Sesat**.
- Extensibility. Plugins and libraries may be sold as products in their own right.

What does it do?

- Handle the complex parts while giving the customer full flexibility to add own features
- Provide rapid development components not only for collecting information from various sources, but also for assembling your search portal
- Complements search applications

Why should I care?

- ROI
  When **Sesat** is in place, you can go to negotiate with your data source deliver, since **Sesat**gives you the opportunity to replace data sources without changing the front end. Your developers can even continuing develop when you are finding out which data sources you wnat to use. The developers can also focus on the front end and the functionality your company want to provide, not on integrate the search engines and other search applications.
- TCO - vanskelig å estimere, må baseres på tidsforbruk + kompetansekostnad
- TTM - time to market. eksempler med sitesearches (dra inn Magne T)
  When first **Sesat**is in place it will be very easy to continues develop your site (or add new sites). The reason for this is that *Sesat *also is a development framework with a lot of helpful default functionality. This default functionality is easy to extend to satisfy your requirements.
- Technical resources (mindre hardcore)
  Sesat will give your company a GUI framework with a well documented API for the search management, so your company's developers can care about the front end functionality, and not struggle with the complexity around the different data sources API etc. **Sesat** will translate the queries on an optimal way to the different supported data sources.
- Agility

Benefits

One great benefit with **Sesat** is that it is independent. It support a lot of the most common data sources today, and it is simple to add new data sources. There exists a well defined API to make a connector. If you want to add a new data source you don't have to worry about the front end. It will still use the same **Sesat** API. You can even replace a data source without change the front end.

Another great benefit with **Sesat** is that it is based and developed as open source. So even your developers can contribute, see the code and apply patches if necessary. By building **Sesat** as an open source, and get the community to work, the product in itself will live and all the customers of **Sesat** will get advantage of each others.

The business rules are easy defined and managed with an XML-file. (In the next version this will be included in the **Sesat** ADM tool)

Components and features

Sesat supports many multiple different data sources, so enterprises can easy use **Sesat** in their framework to extend their functionality.
Sesat can be scaled to unlimited number of instances, data sources.

Support

Norwegian companies Conduct, Comperio, and TRank, have worked as consultants on Sesat along the way.

Stories

See how easy it is to create a simple search website.

The traditional way of creating a search portal or a search application, is to develop a full stack of components from the lowest layer communicating directly with search apis, to the templating layer on top.

In effect, **Sesat** gives you most of these components, so the developer can focus on being productive with front-end coding.

- Cuts administrative overhead - developers can focus on presentation and functionaly

Overview of the **Sesat** Model: The developer focus on functionality rather than middleware and interface coding.

The main parts of Sesat, organizing all libraries, applications and tools, are:

- **Sesat** Kernel - the main application, handling the actual search analysis, content search and result presentation.
- **Sesat** API - Application Programmers Interface. **Sesat**exposes many of its functions as WebServices and XML interfaces. This is key to Ajax functionality and for client applications where the **Sesat** SFC is not an option.
- **Sesat**SFC - Search Front-end Container. This is where client applications reside, and were they utilize the functionality present in the **Sesat** Kernel.
- **Sesat** Adm - administrative plugins and applications. **Sesat** comes with a portal setup based on JBoss Portals.

Client applications reside within the Search Front-end Container, SFC. One **Sesat**-installation can handle multiple client applicatins, or "vhosts", virtual hosts. For example, one single kernel can control and support sesam.no, sesam.se, vg.sesam.no, svd.sesam.se, sports.svd.sesam.se...

This page last changed on Aug 22, 2008 by mick.

Sesat Whitepaper

Do you need help to convince your boss 

Error formatting macro: toc: java.lang.NullPointerException

Preface

Sesat - SEsam Search Application Toolkit - is a stack of software applications aiming to help efficient creation of feature-rich, stable and multi-source based search portals and search applications. Based on experience and learnings from the first installations of Sesam.no and Sesam.se, **Sesat** is now an extensive toolkit with one primary focus: reduce development complexity and dramatically improve quality and delivery time. In fact, **Sesat** lets customers create fully-functional portals in a few days rather than many months, as proved by the teams behind Sesam.no and Sesam.se, all using **Sesat** to speed up site and site search development.

In general, writing a search application from scratch given only one or more indexes providing search results as data objects or XML, the following steps are usual:

- Create an application where the user types a search query
- Create a search function for each of the data sources to be searched in, tailored to the index profile / index data model
- The application queries each of the data sources in turn
- Display the results to the user

But, this is **not at all flexible or scalable**, with code that needs to be rewritten and tailored to any change in the underlying data source models. **A much better approach would be to:**

- Define a dynamic data model for each of the indexes to be searched in
- Implement either a parallel or a sequential search function that takes the search string and queries the different data sources, collects the data and delivers it to a templating engine. For such a function to be stable, it should
  - Be able to analyse the search query to predict which indexes are worth searching in
  - Handle faulty indexes
  - Handle slow response time together with increased traffic, preventing resource starvation
  - Handle the different APIs for the different indexes
  - Have a model for unifying/normalising the search result
  - Be robust when the index model / index profile changes (there should be no need to alter the search function)
- Implement a template engine that gets data from the search function and display it to the user
- Implement a logger that can register all searches with details about the user client, search context, zero-hits, index response time etc.
- Implement some sort of rules engine that triggers on particular terms in the search query, to intelligently boost which data source should be displayed first
- Allow for the application to present search results differently depending on IP-range, user session id, personalisation settings.
- Design templates to display the result

With **Sesat**, you get all of the above by

- Set up your development environment,
- Define your data sources in a configuration file, and
- Hack the example templates that is included.

Introduction

The Sesam Search Application Toolkit, Sesat, is the heart of the http://sesam.no and http://sesam.se search portals. These portals show the power and flexibility of a platform enabling businesses to present both structured and un-structured information from many sources simultaneously, while honoring the user's search context and choices.

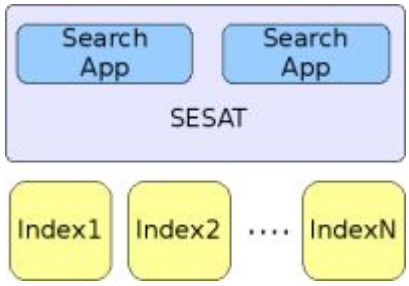
Sesat is built on experience and learnings from the first version of the sesam.no portal, and has grown both in maturity and functionality with the tasks given. It has all the time been an important goal to deliver as much flexibility and functionality to the user as possible, while keeping the "core" application stable. Sesam is today successfully leveraging the flexibility of this architecture, focusing on developing their portal rather than the underlying search technology.

Executive Summary

Sesat includes a set of applications working together to provide a feature-rich search middleware. Sesat glues together information from underlying indexes and data sources and passes it on to the presentation layer. The toolkit has excellent support for all recent versions of the FAST search engine, as well as many other data sources. It is easy to create own connectors to underlying data layers and sources.

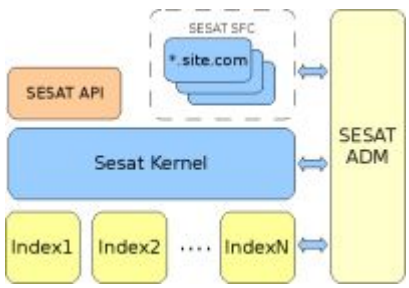
The **traditional** way of creating a search portal or a search application, is to develop a full stack of components from the lowest layer comminicating directly with search apis, to the templating layer on top.

In effect, Sesat gives you most of these components, so the developer can focus on being productive with front-end coding.

	Overview of the Sesat Model: The developer focus on functionality rather than middleware and interface coding.

The main parts of Sesat, organising all libaries, applications and tools, are:

- Sesat Kernel - the main application, handling the actual search analysis, content search and result presentation.
- Sesat API - Application Programmers Interface. Sesat exposes many of its functions as WebServices and XML interfaces. This is key to Ajax functionality and for client applications where the Sesat SFC is not an option.
- Sesat SFC - Search Front-end Container. This is where client applications reside, and were they utilise the functionality present in the Sesat Kernel.
- Sesat Adm - administrative plugins and applications. Sesat comes with a portal setup based on JBoss Portals.

	Client applications reside within the Search Front-end Container, SFC. One Sesat-installation can handle multiple client applicatins, or "vhosts", virtual hosts. For example, one single kernel can control and support sesam.no, sesam.se, vg.sesam.no, svd.sesam.se, sports.svd.sesam.se...

Key points:

- Sesat is not tied to any particular search engine or data storage. The customer may easily provide his own search interfaces and make them available to the Sesat Kernel.
- Sesat uses unicode (UTF-8) throughout, and is as such compatible with most languages and character sets.
- Sesat makes it possible to create search portals and search applications **fast** without worrying about internals of search engine APIs or writing display code, business logic etc. The developer's focus is entirely on the portal's functionality.
- Sesat is a mature search application framework that has been under heavy develpment since authumn 2005. It has an exiting roadmap which includes personalisation, session-handling, Ajax-capabilities, rules engines, statistics tools and administrative tools.
- Sesat is an open architecture. The customer have **access to all source code**, and can submit code to the project.
- Sesat uses Open Source applications and libraries. No commercial software needs to be installed to run Sesat.
- Extensibility. Extremely modular and pluggable architecture.

Architecture

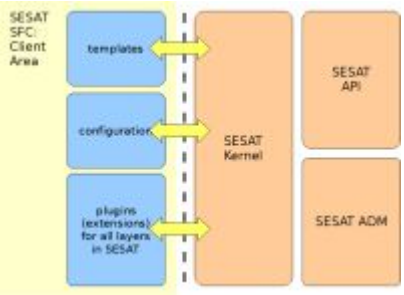
Overview

Sesat implements a search portal middeware. It lives in the layer between the application's presentation templates and one or more data sources. It is flexible enough to handle many different types of sources:

- FAST-indexes
- Lucene-indexes
- WebServices
- XML-RPC
  and many more. It is very easy to extend the engine to handle other search interfaces.

The power of Sesat is its ability to search in multiple indexes at the same time, and then create a java object containing all results. This object can then be accessed either via templates (eg Velocity or JSP), or by Ajax-calls. Sesat understands user sessions, and therefore the object can be kept ready for subsequent calls and transactions.

Sesat is controlled by configuration. Some of this configuration is done by means of administrative tools - Sesat ADM. The administrative tools are implemented to run on JBoss Portals, a portal framework compliant to JSR-168, running on JBoss application servers. This framework is also home of several administrative plugins, like personalisation and statistics.



On the left-hand side, in the Search Front-end Container, the client defines his templates. These are being run by the Sesat Kernel upon user requests. Also, the client provides information about which indexes is to be used, which analysis rules apply etc. In addition, most of the layers in the Kernel define a Service Provider Interface, SPI. The user can implement extensions to functions within the Kernel, ensuring flexibility.

Architecture of an example application

Core components

A full Product Description also exists.

Sesat Kernel

> Sesat Kernel runs as a web application within a web-server. The Sesat Kernel, the core to the search front, and the generic.sesam skin (SFC) serves as a search engine's controlling and presentation framework.

Currently, Tomcat and JBoss is supported. All access to a search application or portal, is managed by this part of Sesat. The Kernel is configured to work with a number of Virtual Hosts, vhosts. When starting the service, the Kernel loads information about all active vhosts. Each of these may have their own templates, indexes and configuration. Client code for any given vhost is initialised and loaded at this stage.

A full Kernel Feature list can be found here.

Search Lifecycle

Sesat SFC - Sesat Search Front-end Container

> The SFC is where client applications run. They are deployed and directed by the Sesat Kernel.

Applications run within the same JVM as the Kernel. They are all deployed as .war-files to the same directory, by default.
The appplication names are the same as the vhost they implement.

Features:

- Very quick development and deployment of search application based on the kernel
- Flexible and configurable:
  - skins, vhosts, ranking, indexes, analysis rules, navigators and enrichments
- Runs in the same JVM as the kernel for speed, but **can** run on a separate JVM for security.
- Defined logfiles for developer debugging and statistics.

Sesat API

> Most Kernel functionality is accessible from WebServices and the XML-interface. This is important functionality external applications that needs to communicate with Sesat. For example, mobile applications have been developed to perform searches using the XML API.

Features:

- XML RESTful API.
  - RSS / Atom feeds automatically generated from any already configured query.
- XML-RPC:
  - XML-based API with full support for all services in the kernel. Provided to tightly integrated partners.
  - Output configurable on a per-partner basis
  - Security services
- Web Services (WS)
  - WS for loosely coupled client application

Please note that XML-RPC and Web Services to Sesat are not yet implemented - We had great plans for XML-RPC and/or a Web Service API but us, and all our partners, always come back to using RSS (or custom XML) feeds. That's because at the moment the API, like most search APIs, is all RESTful so URL requests and XML (or serialised datamodel) responses gives you the full API.

Sesat ADM

(source code to ADM not yet released)

> The Sesat admin is a collection of applications and tools that run in a JBoss Portals server. The applications interact with various parts of Sesat, for example the Statistics engine.

Sesat Admin consists of portlets and applications running on JBoss Portals (server), which is a standards-compliant (JSR-168) portal. As of version 1.0 of Sesat, the following appliacions are available:

- User Administration Portlet (require usage of OpenLDAP). The administrator can define users, groups and roles, and ensure that access to other administrative systems or the search portal are prohibited and controlled.
- Statistics Portlet, providing simple statistics of queries and traffic through parsing of the logfiles.
- FAST Lists Editor (version 1.0 of Sesat). FAST Lists are lookup-tables within the FAST engine that are used when analysing search queries.
- Rules Engine Portlet (version 1.2 of Sesat)
- Personalisation Manager (version 1.2 of Sesat)

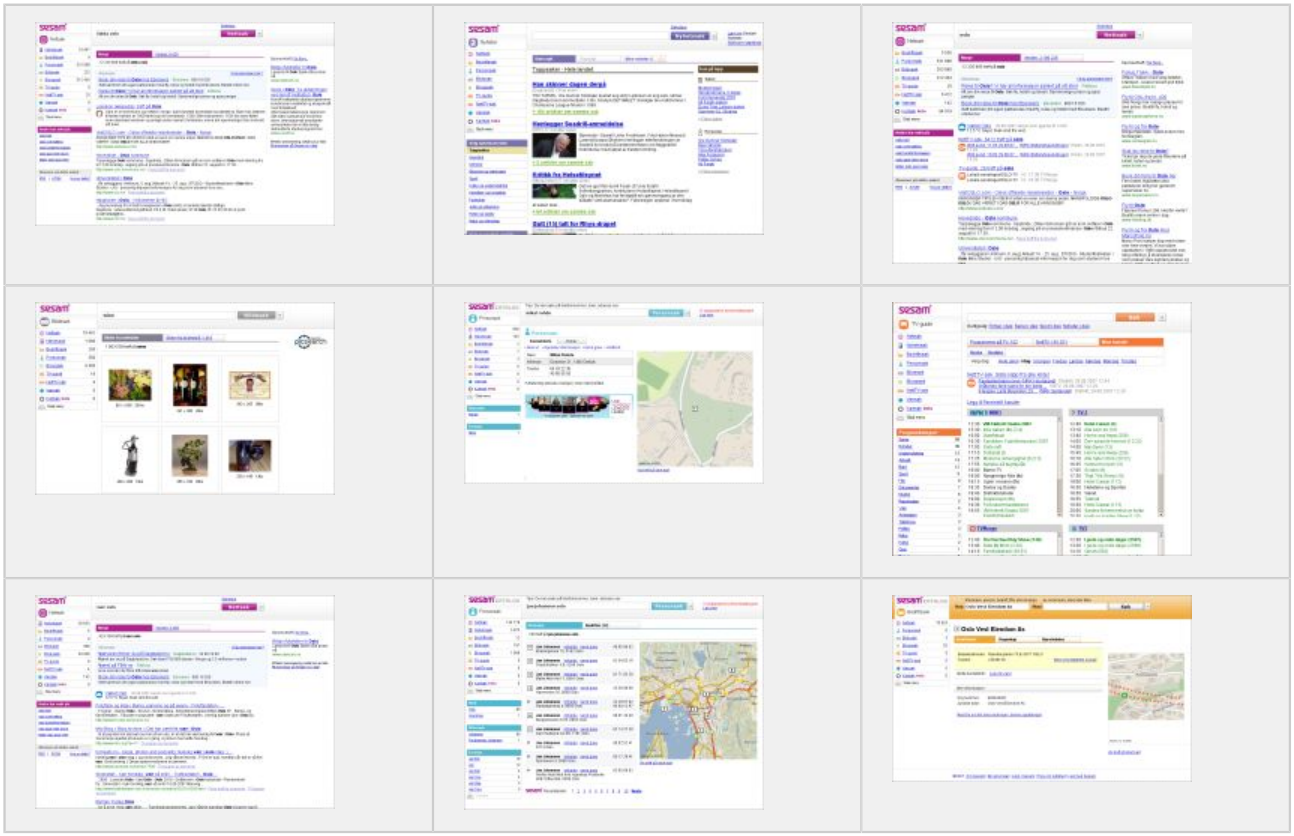
Sesat License

You may use Sesat under the Affero GPLv3 license. This is a Free Software and Open Source license. All your source code including that belonging to derivative works must be made freely available to any user that interacts with the software locally or via any network (eg the internet).

If this does not suit your business needs, for example if you need to write propriety and private derivative works, then please contact us to obtain a Propriety License.

This page last changed on Apr 28, 2008 by mick.

Showcase: Sesam.no



This page last changed on Sep 26, 2008 by mick.

Feature Matrix

We keep an up to date and detailed Kernel Feature list here.

Feature	Description
Installation and environment	
Linux flavours	All of course, but tested on CentOS, RedHat, Ubuntu and Debian. Works well on local Windows installations (not recommended for heavy production sites)
Application Servers Supported	Tomcat 6.0+, JBoss 4.2.0+
Apache front-end proxying supported with mod_cache and mod_jk	Yes
Runs on same server as client application	No need for separate server(s). SESAT is deployed directly on the Java Application Server
Supports Java 1.6	Yes
Multiple Sites in the same JVM	Yes, with properties and templates inheritance for (sub)sites (e.g. vg.sesam.no can inherit properties and settings from sesam.no)
Easy configuration of each defined site/ application	Yes, currently by XML-files, but with GUI-administration module under development.
Scalability	Scales linearly with number of servers using HW load balancing with sticky sessions.
Automated Deployment	Deployment scripts included for deployment of both SESAT and client applications.
Source Code	

Full access to Source Code	yes, from Subversion
Permissions to apply own patches	Yes
Special functionality built upon request	Yes, contribute it yourself or persuade the developers.
Site GUI Development	
Search Portal Framework	Yes. SESAT is both federating search middleware and a search application framework.
Development Environment	Anything you want. We recommend both NetBeans and Eclipse.
Development OS	Linux, Mac OS X and Windows
Client application building	Maven2-based generation of .war deployment files
Content formats supported	HTML, XHTML, XML, VCF, VML, RSS, etc.
Site inheritance	SESAT allows for sites to inherit layout and configuration from parent sites.
Example templates included	Yes, extensive stack of templates based on several real-life sites. (Velocity or JSP)
Easy development environment setup	Yes. Supports ssh-tunnels to data sources.
Search result access method	The Search Result Object is directly available from the template system.
Normalised search result	The Search Result Object presents the result in a normalised and uniform way, independent of the underlying index technology.
Templating System	Velocity and JavaServerPages are standard, support for adding any other

	Java-based templating systems.
Ajax-support	Search Result Object is connected to user sessionID to provide asynchronous information access.
Support for Java Faces	Not as standard yet, but can easily be implemented by client. Near-future versions of SESAT will have working examples of Java Faces.
Full flexibility for templating framework	Yes.
Query Analysis	
Configurable business logic	Yes, through the rules engine and administrative tools.
Rules-engine based Query Analysis	Yes, with rules managed by own module in SESAT ADM
Automatic search-phrase highlighting	Yes, for FAST-indexes. Highlightning independent of index type will be released in the near future.
Specification and administration of trigger-words	Yes, triggers may change ranking order of individual indexes in the search result. For example, "**fact** oslo" may boost content from a Wikipedia index before content from a global crawled web index.
Administrative back-end for rules engine	Yes, SESAT ADM Text Analysis providing automatic detection of full-names, geographical names etc.
Search term categorisation based on lookup-lists	yes
Phrase-search support	yes
Standard Search Operator support	yes, all functionality is inherited from the underlying index motor

Easter-eggs support	yes, of course!
In-built calculator	Yes, queries can be mathematically evaluated
Result Handling	
Result Object	Comprehensive, easily accessible object model
Simple Flat Result List	Search result represented as java object
Result persistence	Yes, attached to user SessionID, configurable duration
Handle both generic and specific index results	Yes - user configurable and extendable
Enrichments Support	
Enrichments ranking based on rules	yes
Easy creation and definition of enrichments	Yes, you can use any search command as an enrichment that will be displayed and ranked according to initial search analysis.
Unlimited number of enrichments	Yes
Ad Systems Support	
Yahoo PPC	yes
AdMomentum	yes
Google AdWords	yes
Easy to add new Ad Systems	Easy for user to implement Ad Systems by creating new, simple search commands
Data Stores Support	
Default Data Index Support	Default Search Command implementations for Fast-4 Simple search, Fast-4 Advanced search, Fast-5 ESP search, generic XML search, Web Services seach, Blending (FAST) search, Clustering

	(FAST) search, Correcting (FAST) search, Mathematical/ Calculator search, Overture PPC search, PicSearch search, Platefood PPC search, Sensis search, Solr (Lucene) search, Stock market search, Yahoo IDP search, Yahoo Media search, and Youtube search.		
Support for external data stores	Yes, through HTTP, WebServices, JDBC. Easy to add own data sources using the SPIs.		
Roadmap Data Source/ Index Support	Lucene, Google, SharePoint, other sources we may need to search in		
Reference Search Command implementations	Blinkx Video search, Blocket search, Finn search, GeoData Map search, HittaMap search, Hitta search, HittaWeather search, Prisjakt search, Storm Weather Blinkx Video search, Tasteline search, TV-guide (FAST) search, Whitepages (FAST) search, Yellow Pages (FAST) search		
Support for Custom Search Commands	 Easy to develop using inbuilt SPI		
Core Application Extensibility			
Available SPIs (Service Provider Interfaces) for extending core functionality	Yes		
Full access to source code	Yes		
Independent release-cycles of minor kernel upgrades vs. client code	Yes		
Backwards-compability	Complete Upgrade Guides kept up to date.		
APIs			

WebServices/XML support	Any search result is available through XML interfaces.
SOA-enabled interfaces	Yes, through WebServices
Search Logging Support	
Logging of requests to underlying search engines	Yes, everything can be logged with timestamp and result information (no. of hits)
Logging of client information	Yes, user environment, browser characteristics
Search context logging	Referer-domain, "skin" (client site from which the user is searching), navigational elements.
User navigation tracking	Yes, using SESAT's **Boomerang** component, you can track user behaviour on your site, and also any access to external sites. Also, any asynchronous access activities can be logged by Boomerang.
Administrative tool for reading logs	Yes, part of SESAT ADM
Automatic anonymisation of IP-address	Yes, configurable part of the statistics parser.
System behaviour logging	For each search, SESAT logs search execution times (on a per-index basis), zero-hitcount search results (on a per-index basis), enrichments, slow responding indexes
Monitoring/ Surveillance Logging Support	
Slow index/non-responding index detection	Yes, logged to own surveillance log
Personalisation	
Basic personalisation support	Yes

Persistence storages supported	MySql currently supported, but other databases available from JDBC
Security	user session keys generated at each request
Sample code	Yes, working example of personalisation provided
User credentials storage	OpenLDAP
Specific Index Functionality: FAST	
Full navigator support	Yes, through Result Object and template system
Full support for rank profiles	Yes, the profile can be specified in the search command.
Full support for collections	Yes
Robust Error-handling for common FAST-errors	Yes, for example query matching failures/ downtime

Docs + Support

This page last changed on May 12, 2008 by mick.

- Technical Feature Matrix shows the main SESAT features and how works together with search technologies from http://www.fast.no, such as FAST ESP and FAST Unity.
- Product Whitepaper
- Development Guidelines. Checking out the code and building the kernel.
- Tutorial - Building Sesam.com. Tutorial running you through how to build a skin (SFC) on top of the kernel.
- Debugging Tips and tricks for debuging Sesat.
- faq Frequenty Asked Questions.

This page last changed on Apr 17, 2008 by mick.

SESAT Architecture Overview

Overview

SESAT is composed of the following main components:

- SESAT Kernel - Main engine
- SESAT API - Application Programmers Interface
- SESAT SFC - Search Frontend Container
- SESAT ADM - Administrative Portal

The Components

SESAT Kernel

- Analyses queries
- Performs searches, handles timeouts
- Applies analysis and search rules
- Creates the «search result object»
- Handles personalisation

SESAT API – work in progress

- XML-RPC:
  - XML-based API with full support for all services in the kernel.
  - Output configurable on a per-partner basis
  - Security services
- Web Services (WS)
  - QoS governance
  - WS for loosely coupled consumers

SESAT SFC

- The Search Frontend Container allows quick deployment of search application based on the kernel
- Configurable:
  - skins, vhosts, ranking, indexes, analysis rules, navigators and enrichments
- Runs in the same JVM as the kernel for speed

SESAT ADM – source code not released yet

- Administration of analysis and search rules
- Administration of FAST-lists (wordlists)
- User/LDAP administration
- Statistics tools (query analysis, traffic analysis)
- Portal framework (JSR-168 compliant)

Design Proposals

This page last changed on Jul 10, 2008 by mick.

This page last changed on Aug 21, 2008 by mick.

Design proposal for SearchCommand and AbstractSearchCommand

Issue SKER2149: (Divide & Conquer AbstractSearchCommand to delegates)

Error formatting macro: toc: java.lang.NullPointerException

Current problems

- combination of various concern implementations leads to excessive subclass count or restrictive subclass implementation, eg simple or advanced filter combined with simple or advanced query gives four possible implementations with 100% duplication,
- poor parameter handling, offset, sortby, results-to-return, etc
- too many visitors, or unclear of various responsibilities involved,
- filter confusion: fielded clauses filters versus custom filters versus QueryTransformer.getFilter()
- ...

Current API

SearchCommand Context

ResourceContext
DataModelContext
SearchConfigurationContext
TokenEvaluationEngineContext

SearchCommand Concerns

Currently search commands have the methods:

SearchCommand	AbstractSearchCommand	AbstractXmlSearchComma	AbstractSimpleFastSearchCo	AbstractESPFastSearchCommand
getSearchConfiguration()	execute()	getResultsToReturn()	createnavigationFilterStrings()	getWord
call()	isCancelled()	getXmlResult()	setSearchEngineFactory(IF	anyOfFastSearchEngineFactory)
handleCancellation()	performQueryTransformation()	getHttpR	addNavigatedTo(String)	appendFilter(String,String)
isPaginated()	performExecution()	eader()	getNavigators()	getSortBy()
isUserSortable()	performResultHandling()		getResultsToReturn()	modifyQuery(IQuery)
	getFilter()		getSearchEngine()	createSearchResult(IQueryResul
	setFilter()		getSortBy()	getIQueryResult()
	getAdditionalFilter()		setAdditionalParameters(IS	isNavigatable()
	visitImpl		collectSpellingSuggestions(IQueryResult,	NavigatableESPFastCommand
	visitXorClause(..)		FastSearchResult)	createNavigationFilterStrings()
	getSearchResult(..)		createSpellingSuggestion(String)	addNavigatedTo(String)
	getOffset()		collectResults(IQueryResult)	getNavigatedTo(String)
	isPaginated()		createResultItem(IDocumentSummary)	getNavigatedValues()
	geUserSortBy()		createQuery()	getNavigatedValue(String)
	isUserSortable()		getDynmicParams(Map,String,String)	getNavigatedTo()
	getParameter(String)		getNavigatorsString()	collectModifiers(IQueryResult,
	getSingleParameter(String)		flattenNavigators(Collection<Navigator>,	FastSearchResult)
	getQuery()		Navigator)	collectModifier(String,IQueryRes
	getQueryRepresentation(Query)		collectModifiers(IQueryResult,FastSearch	FastSearchResult)
	getTransformedTerm(Clause)		collectModifier(String,IQueryResult,FastSearchResult)	getModifierComparator(Navigato
	getTransformedTerms()		getModifierComparator(Navigator)	
	initialiseTransformedTerms()		createProperNameSuggestion(String)	

	appendToQueryRepresentation(String) insertToQueryRepresentation(String) getQueryRepresentationLength() escapeFieldedLeaf(LeafClause) isEmptyLeaf(Clause) getFieldFilter(LeafClause) getTransformedQuery() getTransformedQuerySesamSyntax() setTransformedQuerySesamSyntax() newSesamSyntaxQueryBuilder() updateTransformedQuerySesamSyntax() getEngine() statisticInfo(String) MapVisitor FilterVisitor SesamSyntaxQueryBuilder	collectRelevantQueries(IQueryResult,FastSearchResult	**NewsEspCommand** addMedium(Clause) createCollapsedResults(NewsEsp addResult(NewsEspCommandCo **NewsClusteringESPFastCommand** createClusteringSearchResult(Cl

A number of clear concerns can be seen:

- Execution handling `call() execute() handleCancellation() isCancelled() performQueryTransformation() performExecution() performResultHandling()`
- Parameters `getParameter(String) getSingleParameter(String)`
  - Result list size `getResultsToReturn()`
  - Pagination & Offset `isPaginated() getOffset()`
  - Sorting `isUserSortable() getUserSortBy() getSortBy()`
- Command applicable Query `getQuery()`
- Command Query construction --> internal visitor `visitImpl(*) getTransformedTerms() getTransformedTerm(Clause) initialiseTransformedTerms() appendToQueryRepresentation(String) insertToQueryRepresentation(String) getQueryRepresentationLength() escapeFieldedLeaf(LeafClause) isEmptyLeaf(Clause) getTransformedQuery() MapVisitor`
- Command XorClause handling `visitXorClause(XorClause)`
- Filter (fielded clauses) `getAdditionalFilter() FilterVisitor`
- Custom Filters `getFilter() setFilter() QueryTransformer.getFilter()`
- Modifiers (Domain Navigators)
- Stream handling/manipulating
- Reserved words
- Collapsing (& Clustering)
- Suggestions: Spelling, ProperName, Relevant queries
- Result Construction
- Utility (almost static methods  `getSearchResult(..)`

Solutions

Execution handling

Can remain as is.
This is in fact a concrete implementation of Callable's "ResultList<ResultItem> call()" method by AbstractSearchCommand, and it introduces the concepts of QueryTransformation, Execution, and Result Handling. AbstractSearchCommand express solely this, everything else it does should be moved out.

ToDo The SearchConfiguration classes SearchConfiguration, AbstractSearchConfiguration, and CommandConfig do not match the behaviour separation defined between SearchCommand and AbstractSearchCommand. For example, a brand new fresh SearchCommand implementation requires a SearchConfiguration with little of the properties actually defined in SearchConfiguration.

Execution handling generics

Although the use of generics requires a review. Using generics in method signatures intended to be subclassed is something a little more complicated to design properly and IMHO was not done correctly the first time.
For example:

```
public interface SearchCommand extends Callable<ResultList<? extends ResultItem>>
public abstract ResultList<? extends ResultItem> execute()
public ResultList<? extends ResultItem> call()
protected final ResultList<? extends ResultItem> getSearchResult(String, DataModel)
protected final ResultList<? extends ResultItem> performExecution()
protected final void performResultHandling(final ResultList<? extends ResultItem> result)
```

No subclass can actually subclass these methods and provide exact generics to the signatures because each command, for example, returns a BasicResultList upon cancellation or handled internal checked exception.

Parameters

Parameters typically have three sources, and they first found used: a url parameter, a user parameter, the command's configured parameter.
Sometimes (eg userSortBy and pagination) the configuration actually comes from the presentation layer. The command's configuration here must simply point to where in the presentation layer this configuration can be found. Strictly speaking the domain ayer should be isolated from the presentation layer but here we access only the presentation layer's configuration through the datamodel.

ResultToReturn is an interesting example. It should be both overridable from url and user parameters. But the configuration exist in both the presentation layer and the domain layer. The domain layer's only responsibility is to ensure **at least** the amount of results are returned that the presentation layer wants. Up until now its just been presumed that the command's configuration is hardcoded to a value larger than any possible presentation value.

An example implementation can be viewed here: SearchCommandParameter code example.

Command applicable Query

Command Query construction

Magnus Eklund's solution follows:

> Introduce a new interface called QueryBuilder. Query Builders are used to transform the query object into a string representation.
> - Remove "extends AbstractReflectionVisitor" from AbstractSearchCommand.
> - New interface QueryBuilder
> - Add property queryBuilder to SearchConfiguration to hold the FQN of a QueryBuilder class. This property will correspond to a <queryBuilder>- tag in modes.xml to allow for configuration of the query builder itself. (e.g. <queryBuilder class="no.sesat.search.mode.command.query.PrefixQueryBuilder" supportsAndNot="false" />)
> - Create QueryBuilder implementations:
>   - PrefixQueryBuilder
>   - SesamQueryBuilder
>   - InfixQueryBuilder.
>   - Fast4QueryBuilder (if needed, should be possible to configure one of the above to produce suitable syntax)
>   - Fast5QueryBuilder (if needed, should be possible to configure one of the above to produce suitable syntax)

These would still visitors over the query but transforming from the transformedMap into a string representation.
The transformedMap is the mutable state of each clause through the search command's query transformations.
QueryBuilder also needs to have an API to deal with XorClauses, ReservedWords, and supported filter clauses. The definition of these things would be supplied through the context.

Backward Compatibility could be helped by, when no QueryBuilder is specified, creating a proxy against the search command and using it presuming it contains all the QueryBuilder methods required. The alternative backard compatibility would be to refactor AbstractSearchCommand to DeprecatedSearchCommand, so that existing search commands work as is. Problem here is that we'd end

up with a score of DeprecatedXyzSearchCommand classes as many of the subclasses are rewritten to the new AbstractSearchCommand implementation.

Example starting implementation can be read QueryBuilder code example.

Command XorClause handling

This can stay how it is. It must be provided to any QueryBuilder's context.

Reserved Words

Only exists in AbstractESPFastSearchCommand. Should be formalised and brought down to the SearchCommand interface.
Since the enumeration belongs to a particular SearchCommand implementation the enumeration should also return how each item is to be escaped in any query.
It must be provided to any QueryBuilder's context.

Fielded clause (& Custom) filters

Where filters (included custom filters) go is a prickly trick. Some command's have the filter simply appended to the query, while others have no notion of a filter at all.
Worse yet is that some filter must go into the query and others defined as separate parameters.

Should the transformedMap state be made available to multiple QueryBuilder within any one request, so that a command that request filters in a separate parameter to the query could run two separate QueryBuilders. The query QueryBuilder would ignore filters, the filter QueryBuilder would ignore non-filter terms. Exaggerating it, commands that requested *each* filter in a separate parameter could have multiple filter QueryBuilders each ignoring everything but the one filter it is responsible for.
But how does this deal with custom filters that come from the configuration, not the query. Should each custom filter be deserialised into a Query tree/object and QueryBuilders be able to visitor multiple query trees???

Custom filters

Stream manipulation & Result Construction

Stream manipulation exists in AbstractXmlSearchCommand. Should be formalised. A little awkward, for example FAST commands work against a java api.
Could be defined in auxiliary interfaces, eg RestfulSearchCommand, StreamReaderSearchCommand, that could be mixed and matched.
Example implementation can be read Stream manipulation code examples.
Result Construction is a higher level concern than stream manipulation. Likely will be the user of stream manipulation where applicable.
While the return values of such methods are easy to define, the parameters are not. Some commands parse streams, some local files, some use a webservice or local library api.
Example can be seen within the Stream manipulation example here

Collapsing (& Clustering)

Modifiers

Defined in an auxiliary interface, eg NavigatableSearchCommand.
Modifiers today are generic, built from the Navigator which holds the configuration, and the search command's results.
Defined behavior by an interface would result in the FastNavigationController becoming a ModifierNavigationController.

I do not see a need for delegation in the design as in most cases the parsing of the information required to construct the modifiers is specific to the search command.
Keep in mind that this is a subset to the Result Construction concern.

Utility behavior

Stay as is. Actual static methods can be moved to a SearchCommandUtility class.

QueryBuilder code example

This page last changed on Aug 21, 2008 by mick.

Error formatting macro: toc: java.lang.NullPointerException

QueryBuilder interface

```
/** QueryBuilder provides a string representation of a Query Tree against of map of "transformed
terms".
*
* It is similar in functionality to the QueryTransformer interface
*  except that it does not transform terms but uses them to build the final string representation.
*/
public interface QueryBuilder extends Visitor {

interface Context extends QueryContext, ResourceContext, SiteContext, DataModelContext{

/**
* Get the terms with their current transformed representations.
* @return
*/
Map<Clause, String> getTransformedTerms();

/** Get the unescaped transformed term for the clause.
*
* @param clause
* @return unescaped transformed term
*/
String getTransformedTerm(Clause clause);

/**
* For evaluation acitions on individual (or the whole query) terms.
* @return
*/
TokenEvaluationEngine getTokenEvaluationEngine();

/**
* QueryTransformers must follow the same XorClause hints as the search command. *
* @param visitor
* @param clause
*/
void visitXorClause(Visitor visitor, XorClause clause);

/**
* QueryTransformers needs information about supported field filters. *
* @param clause
* @return
*/
String getFieldFilter(LeafClause clause);

/** The collection of words that have special meaning/function within the query string
*
* @return collection of reserved words
*/
Collection<String> getReservedWords();

/** Escape the word.
* The word need not be reserved or require escaping but should be escaped anyway.
*
* @param word
* @return escaped version of the word
*/
String escape(String word);
}
```

```
/** The Query String built from the Query's transformed clauses
*
* @param query
* @return string built from the Query's transformed clauses
*/
String getQueryString(Query query);
}
```

Abstract QueryBuilder

```
/** Helper abstract implementation handling stringBuilder functionality behind the visitor pattern.
**/
public abstract class AbstractQueryBuilder extends AbstractReflectionVisitor implements
QueryBuilder {

// Attributes ------------------------------------------------

private final Context context;
private final QueryBuilderConfig config;
private final StringBuilder sb = new StringBuilder(128);

// Constructors ----------------------------------------------

public AbstractQueryBuilder(final Context cxt, QueryBuilderConfig config) {

context = cxt;
this.config = config;
}

// Public ----------------------------------------------------

public String getQueryString() {

final Clause root = context.getQuery().getRootClause();
sb.setLength(0);
visit(root);
return sb.toString().trim();
}

// Protected -------------------------------------------------

protected final Context getContext(){
return context;
}

protected QueryBuilderConfig getConfig(){
return config;
}

/** Gets the transformed term, escaping any reserved words.
*
* @param clause
* @return
*/
protected String getEscapedTransformedTerm(final Clause clause){

return escape(context.getTransformedTerm(clause).toLowerCase());

}

/** Escapes any reserved words (including those fielded).
* Case-insensitive.
*
* How to actually escape any matching words is left to the context to define via
context.escape(word)
```

```
*
* @param string
* @return possibilly escaped string
*/
protected String escape(final String string){

for (String word : getWordsToEscape()) {

// Case-insensitive check against word.
// Term might already be prefixed by the TermPrefixTransformer.
if (string.toLowerCase().endsWith(':' + word.toLowerCase()) ||
string.equalsIgnoreCase(word)) {

final Pattern p = Pattern.compile(
Matcher.quoteReplacement(word),
Pattern.UNICODE_CASE | Pattern.CASE_INSENSITIVE);

return p.matcher(word).replaceAll(context.escape(string));
}
}

return string;
}

protected Collection<String> getWordsToEscape(){
return context.getReservedWords();
}

protected final void appendToQueryRepresentation(final CharSequence addition) {
sb.append(addition);
}

protected final void appendToQueryRepresentation(final char addition) {
sb.append(addition);
}

protected final int getQueryRepresentationLength() {
return sb.length();
}

protected final void insertToQueryRepresentation(final int offset, final CharSequence addition) {
sb.insert(offset, addition);
}

protected boolean isEmptyLeaf(final Clause clause) {
return false;
}

protected boolean isEmptyLeaf(final LeafClause clause) {

final String tt = 0 == context.getTransformedTerm(clause).length()
? null
: context.getTransformedTerm(clause);

return
// no field and a valid term
null == clause.getField() && null == tt
// or, a field that is an accepted filter
|| null != clause.getField() && null != context.getFieldFilter(clause);

}

protected boolean isEmptyLeaf(final DoubleOperatorClause clause) {

return isEmptyLeaf(clause.getFirstClause()) && isEmptyLeaf(clause.getSecondClause());
```

```
}

protected void visitImpl(final XorClause clause) {
getContext().visitXorClause(this, clause);
}
}
```

Infix QueryBuilder implementation

```
/**
* Largely mimics the Query tree layout replacing OperatorClauses with the RESERVED_WORDS.
*/
public class InfixQueryBuilder extends AbstractQueryBuilder{

// Constructors -------------------------------------------------

public InfixQueryBuilder(final Context cxt, final QueryBuilderConfig config) {
super(cxt, config);
}

// protected ----------------------------------------------


@Override
protected InfixQueryBuilderConfig getConfig() {
return (InfixQueryBuilderConfig) super.getConfig();
}

@Override
protected Collection<String> getWordsToEscape() {

final Collection<String> words = new HashSet<String>(super.getWordsToEscape());

words.add(getConfig().getAndInfix());
words.add(getConfig().getNotPrefix());
words.add(getConfig().getOrInfix());

return words;
}


protected void visitImpl(final LeafClause clause) {

appendToQueryRepresentation(getEscapedTransformedTerm(clause));
}

protected void visitImpl(final OperationClause clause) {

clause.getFirstClause().accept(this);
}

protected void visitImpl(final AndClause clause) {

clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ' + getConfig().getAndInfix() + ' ');
clause.getSecondClause().accept(this);
}

protected void visitImpl(final OrClause clause) {

clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ' + getConfig().getOrInfix() + ' ');
clause.getSecondClause().accept(this);
}
```

```
protected void visitImpl(final DefaultOperatorClause clause) {

clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ');
clause.getSecondClause().accept(this);
}

protected void visitImpl(final NotClause clause) {

final String childsTerm = getEscapedTransformedTerm(clause.getFirstClause());
if (childsTerm != null && childsTerm.length() > 0) {
appendToQueryRepresentation(getConfig().getNotPrefix());
clause.getFirstClause().accept(this);
}
}

protected void visitImpl(final AndNotClause clause) {

final String childsTerm = getEscapedTransformedTerm(clause.getFirstClause());
if (childsTerm != null && childsTerm.length() > 0) {
appendToQueryRepresentation(getConfig().getNotPrefix());
clause.getFirstClause().accept(this);
}
}
}
```

Prefix QueryBuilder implementation

```
/** QueryBuilder prefixing terms depending on their inclusion/exclusion.
*/
public class PrefixQueryBuilder extends AbstractQueryBuilder{

// Attributes ------------------------------------------------

// third state variable. TRUE --> must have clause, FALSE --> must not have clause, null -->
optional clause.
private Boolean clauseState = Boolean.TRUE;

// Constructors ----------------------------------------------

public PrefixQueryBuilder(final Context cxt, final QueryBuilderConfig config) {
super(cxt, config);
}

// Protected -------------------------------------------------


@Override
protected PrefixQueryBuilderConfig getConfig() {
return (PrefixQueryBuilderConfig) super.getConfig();
}

@Override
protected Collection<String> getWordsToEscape() {

final Collection<String> words = new HashSet<String>(super.getWordsToEscape());

words.add(getConfig().getAndPrefix());
words.add(getConfig().getNotPrefix());
words.add(getConfig().getOrPrefix());

return words;
}

protected void visitImpl(final LeafClause clause) {
```

```
insertClauseStatePrefix(clause);
super.visitImpl(clause);
}

protected void visitImpl(final PhraseClause clause) {

if (clause.getField() == null) {
insertClauseStatePrefix(clause);
appendToQueryRepresentation(getEscapedTransformedTerm(clause));
}
}

protected void visitImpl(final AndClause clause) {

clauseState = Boolean.TRUE;
clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ');
clauseState = Boolean.TRUE;
clause.getSecondClause().accept(this);
}

protected void visitImpl(final OrClause clause) {

clauseState = null;
clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ');
clauseState = null;
clause.getSecondClause().accept(this);
}

protected void visitImpl(final DefaultOperatorClause clause) {

clauseState = Boolean.TRUE;
clause.getFirstClause().accept(this);
appendToQueryRepresentation(' ');
clauseState = Boolean.TRUE;
clause.getSecondClause().accept(this);
}

protected void visitImpl(final NotClause clause) {

if(getConfig().getSupportsNot()){
final String childsTerm = getEscapedTransformedTerm(clause.getFirstClause());
if (childsTerm != null && childsTerm.length() > 0) {
clauseState = Boolean.FALSE;
clause.getFirstClause().accept(this);
}
}
}

protected void visitImpl(final AndNotClause clause) {

if(getConfig().getSupportsNot()){
final String childsTerm = getEscapedTransformedTerm(clause.getFirstClause());
if (childsTerm != null && childsTerm.length() > 0) {
clauseState = Boolean.FALSE;
clause.getFirstClause().accept(this);
}
}
}

private void insertClauseStatePrefix(final Clause clause){

if(!isEmptyLeaf(clause)){
if(Boolean.TRUE == clauseState){
```

```
appendToQueryRepresentation(getConfig().getAndPrefix());
}else if(Boolean.FALSE == clauseState){
appendToQueryRepresentation(getConfig().getNotPrefix());
}else{
appendToQueryRepresentation(getConfig().getOrPrefix());
}
}
}
}
```

Sesam Syntax QueryBuilder implementation

```
/** Query builder for creating a query syntax similar to sesam's own.
* Currently is basically a PrefixQueryBuilder with OrClauses wrapped in () parenthesis.
*/
public class SesamSyntaxQueryBuilder extends PrefixQueryBuilder{

// Constructors -------------------------------------------------

public SesamSyntaxQueryBuilder(final Context cxt, final QueryBuilderConfig config) {
super(cxt, config);
}

// AbstractReflectionVisitor implementation ---------------------------------------------

private boolean insideOr = false;

@Override
protected void visitImpl(final OrClause clause) {

boolean wasInside = insideOr;
if (!insideOr) {
appendToQueryRepresentation('(');
}
insideOr = true;
super.visitImpl(clause);
insideOr = wasInside;
if (!insideOr) {
appendToQueryRepresentation(')');
}
}

/** Overridden so to avoid visiting any FULLNAME_ON_LEFT.
*/
@Override
protected void visitImpl(final XorClause clause) {

switch (clause.getHint()) {
case FULLNAME_ON_LEFT:
clause.getSecondClause().accept(this);
break;
default:
super.visitImpl( clause);
}
}

}
```

SearchCommandParameter code example

This page last changed on Aug 15, 2008 by mick.

EXAMPLE implementation of a more dynamic Parameter handling for SearchCommands.

```
interface SearchCommandParameter{

enum PARAMETER_ORIGIN {REQUEST,USER,CONFIGURATION};

enum PARAMETER_TYPE {PLAIN,NAVIGATION_MAP,TAB};

String getName();

boolean isActiveParameter();

PARAMETER_ORIGIN getParametersOrigin();

String getParameter();
}

class BaseSearchCommandParameter implements SearchCommandParameter{

private final String name;
private final PARAMETER_ORIGIN[] lookupOrder;
private PARAMETER_ORIGIN origin = null;

public BaseSearchCommandParameter(
final String name,
final PARAMETER_ORIGIN[] lookupOrder){

this.name = name;
this.lookupOrder = Arrays.copyOf(lookupOrder, 0);
}

public String getName(){
return name;
}

public boolean isActiveParameter() {
return true;
}

public String getParameter() {

String result = null;
if(isActiveParameter()){

for(PARAMETER_ORIGIN origin: lookupOrder){

if(null != this.origin){
// shortcut to the origin found last time.
origin = this.origin;
}
switch(origin){

case REQUEST:
final StringDataObject sdo = context.getDataModel().getParameters().getValue(name);

if(null != sdo){
result = sdo.getString();
}
break;

case USER:
result = context.getUser().getUser().getUserPropertiesMap().get(name);
break;
```

```
case CONFIGURATION:
final PropertyDescriptor[] properties =
Introspector.getBeanInfo(getSearchConfiguration().getClass()).getPropertyDescriptors();
for(PropertyDescriptor property : properties){
if (name.equals(property.getName()){

if( null != property.getReadMethod() ){
result = (String)invoke(property.getReadMethod(), getSearchConfiguration(), new Object[0]);
break;
}
}
}
}
if(null != result){
this.origin = origin;
break;
}
}
}
return result;

}

public PARAMETER_ORIGIN getParametersOrigin() {

if(null == origin){
getParameter();
}
return origin;
}

}

class NavigationSearchCommandParameter extends BaseSearchCommandParameter{

private final String navigationMapKey;

public NavigationSearchCommandParameter(
final String name,
final String navigationMapKey,
final PARAMETER_ORIGIN[] lookupOrder){

super(name, lookupOrder);
this.navigationMapKey = navigationMapKey;
}

@Override
public boolean isActiveParameter() {

final boolean navMapExists = null != context.getDataModel().getNavigation()
&& null != context.getDataModel().getNavigation().getConfiguration();

final Nav nav = navMapExists
? context.getDataModel().getNavigation().getConfiguration().getNavMap().get(navigationMapKey)
: null;

return null != nav && getSearchConfiguration().getId().equals(nav.getCommandName());
}


}
```

Stream manipulation code examples

This page last changed on Aug 21, 2008 by mick.

Error formatting macro: toc: java.lang.NullPointerException

Restful

This example defines an interface outside of the SearchCommand class/interface hierarchy.
This approach permits delegation to occur from SearchCommand class to a Restful instance.

```
/** SearchCommands that are RESTful should implement this behaviour.
* http://en.wikipedia.org/wiki/Representational_State_Transfer
*/
public interface Restful {

/** A RESTful service requires a URL.
* The resource part of the URL is usually constant to the particular command instance,
*  but each search creates a seperate URL.
*
* @return the URL to use to the RESTful service.
*/
String createRequestURL();

/** Obtain a BufferedReader, in the given encoding, of the RESTful result.
* Makes the presumption that the RESTful service returns an ascii and not binary response.
*
* @param encoding
* @return
* @throws java.io.IOException
*/
BufferedReader getHttpReader(String encoding) throws IOException;

}
```

XmlResultful

An extension to the Restful interface when it is known the response will be in xml format.

```
/** SearchCommands that are RESTful and return XML response.
* http://en.wikipedia.org/wiki/Representational_State_Transfer
*
* It makes the presumption of working with JAXP's DOM.
*/
public interface XmlRestful extends Restful{

Document getXmlResult() throws IOException, SAXException;
}
```

Example abstract XmlRestful implementation

```
/**
* Base implementation for search commands that are RESTful and have XML responses.
*
* The RESTful server is defined through:
* host: AbstractXmlSearchConfiguration.getHost()
* port: AbstractXmlSearchConfiguration.getPort()
*
* @version $Id: AbstractXmlSearchCommand.java 6596 2008-05-10 10:05:48Z ssmiweve $
*/
public abstract class AbstractXmlRestful implements XmlRestful{

// Attributes ----------------------------------------------------
```

```
private final transient HTTPClient client;

// Constructors -------------------------------------------------


/**
* Create new xml based command.
*/
public AbstractXmlRestful(final Context cxt) {

final AbstractXmlSearchConfiguration conf =
(AbstractXmlSearchConfiguration)cxt.getSearchConfiguration();

final SiteConfiguration siteConf = cxt.getDataModel().getSite().getSiteConfiguration();
final String host = siteConf.getProperty(conf.getHost());
final int port = Integer.parseInt(siteConf.getProperty(conf.getPort()));

client = HTTPClient.instance(conf.getHostHeader().length() > 0 ? conf.getHostHeader() : host, port,
host);
}

// Public -------------------------------------------------------

public final Document getXmlResult() throws IOException, SAXException {
final String url = createRequestURL();
return client.getXmlDocument(url);
}

public final BufferedReader getHttpReader(final String encoding) throws IOException {
final String url = createRequestURL();
return client.getBufferedReader(url, encoding);
}
}
```

Example Usecase

AbstractXmlSearchCommand here is largely refactored (simplified) and now holds an Restful instance and delegates to it as needed

AbstractXmlSearchCommand also introduces the createItem behaviour requirement as it's the norm that each individual result is defined within a separate element within the xml response.

```
/**
* Helper base implementation for search commands that are RESTful and have XML responses.
*
* The RESTful server is defined through:
* host: AbstractXmlSearchConfiguration.getHost()
* port: AbstractXmlSearchConfiguration.getPort()
*/
public abstract class AbstractXmlSearchCommand extends AbstractSearchCommand{

// Attributes ---------------------------------------------------

private XmlRestful restful;

// Constructors -------------------------------------------------


/**
* Create new xml based command.
*
* @param cxt The context to execute in.
*/
protected AbstractXmlSearchCommand(final Context cxt) {
super(cxt);
```

```
}

// Public ------------------------------------------------------

public final String createRequestURL() {

return restful.createRequestURL();
}

// Protected ---------------------------------------------------


/** Each individual result is usually defined within one given Element.
*
* @param result the w3c element
* @return the ResultItem
*/
protected abstract ResultItem createItem(final Element result);

protected final XmlRestful getXmlRestful(){
return restful;
}

protected final void setXmlRestful(final XmlRestful restful){
this.restful = restful;
}
}
```

PicSearchCommand extends a final implementation on AbstractXmlSearchCommand displaying the delegation pattern at whole.

```
/**
* A search command that uses the picsearch API.
*/
public class PicSearchCommand extends AbstractXmlSearchCommand {

private static final Logger LOG = Logger.getLogger(PicSearchCommand.class);
private final transient HTTPClient client;
private final int port;
private static final String REQ_URL_FMT = "/query?
ie=UTF-8&tldb={0}&filter={1}&custid={2}&version=2.6"
+ "&thumbs={3}&q={4}&start={5}&site={6}&color={7}&size={8}";

/**
* Creates a new command in given context.
*
* @param cxt Context to run in.
*/
public PicSearchCommand(final Context cxt) {

super(cxt);

setXmlRestful(
new AbstractXmlRestful(cxt) {
public String createRequestURL() {


final PictureCommandConfig cfg = (PictureCommandConfig) cxt.getSearchConfiguration();

try {

final String query = URLEncoder.encode(PicSearchCommand.this.getTransformedQuery(), "utf-8");

final String color = null != PicSearchCommand.this.getParameter("color")
? PicSearchCommand.this.getParameter("color")
: "";
```

```
final String size = null != PicSearchCommand.this.getParameter("size")
? PicSearchCommand.this.getParameter("size")
: "";

final String urlBoost = PicSearchCommand.this.getFilterBuilder().getFilter("tldb")
.replace('=', ':')
.replace(' ', ',');

if(null != cfg.getSite() && cfg.getSite().length() > 0){
PicSearchCommand.this.getFilterBuilder().addFilter("site", cfg.getSite());
}

// The boost can eiter be from the URL or from the configuration.
final String boost = URLEncoder.encode(urlBoost.length() > 0 ?
urlBoost : cfg.getDomainBoost(), "utf-8");

return MessageFormat.format(REQ_URL_FMT,
boost,
cfg.getFilter(),
cfg.getCustomerId(),
cfg.getResultsToReturn(),
query,
PicSearchCommand.this.getOffset()+1,
PicSearchCommand.this.getFilterBuilder().getFilter("site").replace(' ', ','),
color,
size);

} catch (UnsupportedEncodingException e) {
throw new SearchCommandException(e);
}

}
});

final SiteConfiguration siteConfig = datamodel.getSite().getSiteConfiguration();
final PictureCommandConfig psConfig = (PictureCommandConfig) context.getSearchConfiguration();

final String host = siteConfig.getProperty(psConfig.getQueryServerHost());
port = Integer.parseInt(siteConfig.getProperty(psConfig.getQueryServerPort()));
client = HTTPClient.instance(host, port);

}

public ResultList<ResultItem> execute() {

final BasicResultList<ResultItem> searchResult = new BasicResultList<ResultItem>();

if (port > 0){
try {

final Document doc = getXmlRestful().getXmlResult();
if (doc != null) {
final Element resultElement = doc.getDocumentElement();
searchResult.setHitCount(Integer.parseInt(resultElement.getAttribute("hits")));
final NodeList list = resultElement.getElementsByTagName("image");
for (int i = 0; i < list.getLength(); i++) {
searchResult.addResult(createItem((Element) list.item(i)));
}
}

} catch (IOException ex) {
throw new SearchCommandException(ex);
} catch (SAXException ex) {
throw new SearchCommandException(ex);
}
```

```
}

return searchResult;

}

@Override
protected ResultItem createItem(final Element picture) {

final BasicResultItem item = new BasicResultItem();
for (final Map.Entry<String, String> entry :
getSearchConfiguration().getResultFieldMap().entrySet()) {
item.addField(entry.getValue(), picture.getAttribute(entry.getKey()));
}
return item;

}
}
```

This page last changed on Jan 29, 2009 by sshafroi.

SESAT Personalization architecture

This is a document describing the design of and how the personalization architecture works in Sesam.

Components / API

The architecture for personalization is build on three systems:

- the search front
- the user admin application
- the RMI backend, the service layer, for handling lookup and persistens of user data.

Both the RMI backend and the user admin application runs on our HA-JBoss environment using the HA-mysql database. The search front and the user admin application must share domain so they can share login cookies.

Search Front

The search front should be as clean as possible. That's why we want to separate the user admin into it's own application. Then we don't have to handle validation and stuff like that in the search front.

The search front will contain a user filter that is used to check if there exists a known user and populate the data model with the user object and the belonging properties. The search front can also set and remove single properties for a user, to make it possible handling i.e. news cases that is implemented as drag-n-drop. And a logout method. For more advanced editing of properties and creating user accounts etc, a link will bring the user over to the admin application.

The user filter will work something like this.

UserFilter.java

```
public void doFilter(..) {

if (null == datamodel.getUser()) {

// Look for login cookie

if (null != cookie) {

// Lookup user data by using cookie

if (null == user) {

// Do nothing, end filter

}

if (legalToken) {

// Populate data model with user data and belonging properties
// Update login cookie

} else {

// Invalidate all logins, warn user of theft

}
}
}
```

```
}
```

User Admin Application

The user admin application should have the following functionality:

- **Create User**
  Form for creating a new user. Submit will create a non-confirmed user object and send confirm mail to the email address that is submitted.
- **Confirm User**
  Just a direct access page for confirming a new user. This page will confirm the user in the database and redirect the user to the login page.
- **Login User**
  Page where the user can log in. After login, get links back to Sesam or update user data.
- **Edit Properties**
  Pages where the user can edit all the properties for his user.
- **Edit User**
  Page where the user can update the user info. If the user is based on an ldap user, the update will happen automatically, and give a message if an update was done.
- **Delete User**
  Users should have the possiblility to delete their user object and all the belonging information.
- **Master Logout**
  This action erases all remembered logins for the user in one action.

RMI backend

The RMI backend should contain the implementation for the EJB3 entity beans, and a service layer that is used by the different applications. The service layer should contain all needed functionality, in an easy accessible way.

The basic user service is a lightweight service class for authenticating users, and an easy way of getting/setting properties.

BasicUserService.java

```
public interface BacisUserService {

BasicUser authenticateByLoginKey(final String loginKey) throws InvalidTokenException;

BasicUser authenticateByUsername(final String username, final String password);

BasicUser findBasicUserByUsername(final String username);

BasicUser refreshUser(final BasicUser user);

void invalidateLogin(final String loginKey);

void invalidateAllLogins(final String loginKey);

void invalidateAllLogins(final BasicUser user);

BasicUser createUser(final String email, final String firstName, final String lastName,
final String password, final String confirmUrl) throws UserAlreadyExistsException,
MailException;

boolean confirmUser(final BasicUser user, final String confirmKey);

void deleteUser(final BasicUser user);

BasicUser setUserProperty(final BasicUser user, final PropertyKey propertyKey,
final String propertyValue);

BasicUser removeUserProperty(final BasicUser user, final PropertyKey propertyKey);

boolean isLegalLoginKey(final String loginKey);
```

```
}
```

The basic user class is a lightweight value class that just wraps in the needed information, totally stripped for entity magic. Remoting and serialization is also easy since it just contains simple Java attributes.

BasicUser.java

```
public interface BasicUser extends Serializable {

Long getUserId();

String getUsername();

String getFirstName();

String getLastName();

Date getCreated();

Date getLastLogin();

boolean isExternal();

Map<PropertyKey, String> getUserPropertiesMap();

String getNextLoginKey();

void setNextLoginKey(final String nextLoginKey);

Date getUpdateTimestamp();

void setUpdateTimestamp(final Date updateTimestamp);

boolean isDirty(final Date timestamp);

}
```

Persistent Login Cookie

Here is a summary of how we're handling the login cookies for Sesam.

Our solution is based on:
Persistent Login Cookie Best Practice

With a posted improvement:
Improved Persistent Login Cookie Best Practice

The summary:

1. When the user successfully logs in with Remember Me checked, a login cookie is issued in addition to the standard session management cookie.
2. The login cookie contains the user's username and a random number (the "token" from here on) from a suitably large space. The username and token are stored as a pair in a database table.
3. When a non-logged-in user visits the site and presents a login cookie, the username and token are looked up in the database.
    a. If the pair is present, the user is considered authenticated. The used token is removed from the database. A new token is generated, stored in database with the username, and issued to the user via a new login cookie.
    b. If the pair is not present, the login cookie is ignored.
4. Users that are only authenticated via this mechanism are not permitted to access certain protected information or functions such as changing a password, viewing personally identifying information, or spending money. To perform those operations, the user must first successfully submit a normal username/password login form.

5. Since this approach allows the user to have multiple remembered logins from different browsers or computers, a mechanism is provided for the user to erase all remembered logins in a single operation.

The problem:

One disadvantage, however, is that if an attacker successfully steals a victim's login cookie and uses it before the victim next accesses the site, the cookie will work and the site will issue a new valid login cookie to the attacker (this disadvantage is far from unique to Miller's design). The attacker will be able to continue accessing the site as the victim until the remembered login session expires. When the victim next accesses the site his remembered login will not work (because each token can only be used one time) but he's much more likely to think that "something broke" and just log in again than to realize that his credentials were stolen.

The improvement:

1. When the user successfully logs in with Remember Me checked, a login cookie is issued in addition to the standard session management cookie.
2. The login cookie contains the user's username, a series identifier, and a token. The series and token are unguessable random numbers from a suitably large space. All three are stored together in a database table.
3. When a non-logged-in user visits the site and presents a login cookie, the username, series, and token are looked up in the database.
    a. If the triplet is present, the user is considered authenticated. The used token is removed from the database. A new token is generated, stored in database with the username and the same series identifier, and a new login cookie containing all three is issued to the user.
    b. If the username and series are present but the token does not match, a theft is assumed. The user receives a strongly worded warning and all of the user's remembered sessions are deleted.
    c. If the username and series are not present, the login cookie is ignored.

Comments

1. The cookie values would be on the form **userId###series###token** (specification below) Example: **1234###550E8400-E29B-11D4-A716-446655440000###3051a8d7-aea7-1801-e0bf-bc539dd60cf3**
2. Users can have several user cookies, so they can be logged in on different machines at the same time.
3. There should be a service that will delete and clean up old data. Login cookies not used within 3 months, and users not accessed within 1 year should be deleted? Could be implemented by a scheduled service in the backend.

Database

Below follow the SQL for the entities we use for personalization.

USER

```
create table USER (
USER_ID bigint(20) primary key auto_increment,
USERNAME varchar(50) not null,
PASSWORD_HASH varchar(50) not null,
PASSWORD_SALT varchar(10) not null,
FIRST_NAME varchar(50) not null,
LAST_NAME varchar(50) not null,
CREATED timestamp not null,
CONFIRMED timestamp null,
LAST_UPDATED timestamp not null,
LAST_LOGIN timestamp null,
EXTERNAL char(1) not null  /* T or F */
) engine=InnoDB charset=utf8;

/* The username should be a mail address for internal users, and the ldap username or email for
external users. */
/* Passwords should not be saved in clear text, but as a hash. */
```

```
/* For ldap users, the password could be just set to i.e. "external", not hashed. */

create index USER_ID on USER(USER_ID);

create unique index USERNAME on USER(USERNAME);

create index USERNAME_PASSWORD_HASH on USER(USERNAME, PASSWORD_HASH);

create index FIRST_NAME on USER(FIRST_NAME);

create index LAST_NAME on USER(LAST_NAME);
```

USER_PROPERTY

```
create table USER_PROPERTY (
USER_PROPERTY_ID bigint(20) primary key auto_increment,
USER_ID bigint(20) not null,
PROPERTY_KEY varchar(50) not null,
PROPERTY_VALUE varchar(2048)
) engine=InnoDB charset=utf8;

create index USER_ID on USER_PROPERTY(USER_ID);

create index PROPERTY_KEY on USER_PROPERTY(PROPERTY_KEY);

create index USER_ID_PROPERTY_KEY on USER_PROPERTY(USER_ID, PROPERTY_KEY);

alter table USER_PROPERTY
add foreign key (USER_ID) references USER(USER_ID);
```

USER_COOKIE

```
create table USER_COOKIE (
USER_COOKIE_ID bigint(20) primary key auto_increment,
USER_ID bigint(20) not null,
SERIES varchar(50) not null,
TOKEN varchar(50) not null,
CREATED timestamp not null
) engine=InnoDB charset=utf8;

create index USER_ID on USER_COOKIE(USER_ID);

create unique index SERIES on USER_COOKIE(SERIES);

create index USER_ID_SERIES on USER_COOKIE(USER_ID, SERIES);

create index CREATED on USER_COOKIE(CREATED);

alter table USER_COOKIE
add foreign key (USER_ID) references USER(USER_ID);
```

Comments

1. Legal property keys is implemented by using a Java enum. Can also be implemented as a code value entity.
2. When creating cookie series, must check for duplicated series id's. Must be done synchronized?

Debugging

This page last changed on Apr 10, 2008 by mick.

- Debugging Velocity Templates

Debugging Velocity Templates

This page last changed on Apr 10, 2008 by sshafroi.

Example on how to set up velocity debug and local editing of files:
(Debug can always be switched on/off by accessing http://localhost:8080/servlet/VelocityDebug)

Issues: Be sure to specify file.encoding: Export JAVA_OPTS=-Dfile.encoding=utf-8

The velocity debug loader takes the following arguments:

-DVELOCITY_DEBUG=true|false Enable velocity debug(disabling cache etc), default is false. (required)
-DVELOCITY_DEBUG_ON=true|false Show debug by default, default is false.
-DVELOCITY_DEBUG_STYLE=standard|onmouseover|silent
standard is full debug showing every template parsed empty or not
onmouseover shows templates when holding mouse over the different templates
silent shows only filename and editable in the div title.
Default is standard.
-DVELOCITY_DEVELOP_BASEDIR=/some/local/filesystem/path/search-main/generic.sesam/war/src/main/webapp commaseparated loadpath, default is nothing

See attached files for example on how to set up CATALINA_OPTS and screenshots.

catalina_opts.sh

```
SRC=$HOME/Source/TRUNK/minimal-search-main

# This defines the loadpath. Files residing in genericno.sesam.no will be loaded before the
generic.sesam.no

TEMPLATES="$SRC/genericno.sesam.no/war/src/main,$SRC/generic.sesam/war/src/main"
export CATALINA_OPTS="-DVELOCITY_DEBUG=true \
-DVELOCITY_DEBUG_ON=true \
-DVELOCITY_DEBUG_STYLE=standard \
-DVELOCITY_DEVELOPERBAR_HIDDEN=false \
-DVELOCITY_DEVELOP_BASEDIR=$TEMPLATES"
```

catalina_opts_minimal.sh

```
SRC=$HOME/Source/TRUNK/minimal-search-main

# This defines the loadpath. Files residing in genericno.sesam.no will be loaded before the
generic.sesam.no

TEMPLATES="$SRC/genericno.sesam.no/war/src/main,$SRC/generic.sesam/war/src/main"
export CATALINA_OPTS="-DVELOCITY_DEBUG=true \
-DVELOCITY_DEBUG_ON=true \
-DVELOCITY_DEBUG_STYLE=onmouseover \
-DVELOCITY_DEVELOPERBAR_HIDDEN=true \
-DVELOCITY_DEVELOP_BASEDIR=$TEMPLATES"
```

Logging, Logfiles, and Statistics

This page last changed on Sep 29, 2008 by bernt.

Error formatting macro: toc: java.lang.NullPointerException

Logging

Skin logfile, eg generic.sesam.log

There exists a logfile for each individual skin. The logging here is from the Skin's ResourceServlet. Upon initialisation of the ResourceServlet there's a series of useful WARN statements written listing exactly

- modification timestamp,
- allowed ipaddresses to restricted resources,
- what is a restricted resource by extension type,
- context path to resource type (by extension type), and
- all WEB-INF/lib jar libraries.

Default daily rotation.

Sesat logfiles

> An important part of SESAT is the logging functionality. All SESAT Kernel actions, choices, search results etc. are extensively logged for later parsing. An important part of SESAT future development is integration of statistics into the administrative portal.

Sesat's logfiles are written by Log4j and defined by sesat-kernel/war/src/main/conf/log4j.xml

The default configuration is daily rotation (sesam.access is the exception).

Two logfiles, sesam.marketing and sesam.sales, are not listed here as they are not yet in use.

SESAT produces several log files targeted to different users.

- sesam.access
- sesam.product
- sesam.analysis
- sesam.statistics
- sesam.initialisations
- sesam.dump

Default location of these logfiles is in your container's log directory.

Types of logfiles

sesam.access

This logfile contains the first log entry for all search requests, thus a good starting point for tracing a search request through the Search application. It partly mirrors the Apache access_log but splits the request up in two or three stages:
1. Initial log entry. This entry contains the actual http request parameters, in the <request>-element. Such as <url>, <http-referer>, <user> and <browser>.
2. Real-url. In case a pretty URL was submitted to the Search application, a second entry is logged. This time with a <real-url>-element.
3. Final log entry. Is logged after the search request has been handled by the Search application, when a response has been returned to the caller, and basically just list the http <response code="">.

Information to look for:

- request_id: The unique request Id that is used in all the logfiles to identify the search request.

- time: Timestamp for when the request was received and when it was done with.
- skin: Which virtual host received this request.
- IP address: Only used for geographical lookup, to tag searches with Geo location.
- HTTP session Id: Only used for counting unique users, can be used for tracking user movements.
- http referer: The HTTP-referer string.
- user-agent: The user-agent string (identifying browser and platform).
- response_code: The http response code.
- **query parameters**: q (phrase), c (index; news, catalog, person etc), offset (>0 for browsing resultsets), output (rss, xml etc), newscase etc.
- **boomerang**: Boomerang-ed events are logged here (newspaper view, company view, Ajax messages).
- **site-search**: The site-search parameters (ss_ss, ss_lt etc) must be parsed from <real-url>.
- **mobile search**: The mobile searches (originator, ua) must be parsed from <real-url>.
- **retriever**: Ajax solution for logging paper archive clicks/reads (/search/writeLog.do?paper=...).

sesam.product

This logfile type is logged to when the Search application is through with the query server communication and knows how many hits the search got and which enrichment types should be returned. The following information can be read out:

- request_id
- timestamp
- mode: The tab (aka vertical aka service aka tjeneste) for which the search request was submitted, e.g. mode="p" for picture search.
- query: The search phrase.
- enrichment size: The number of enrichments returned for this search request.
- no-hits: If this appears in the log entry, it signals that no results were found for the given query and mode.
- **enrichment type**: The actual enrichment types can also be read out from the log entry, not just the number.

sesam.analysis

Related to sesam.product but logs the underlying analysis scores for each query. Breaks the scores down to individual positive and negative hits for each predicate explicit in each analysis score.

sesam.statistics

This logfile type contains more technical information, such as a servlet's execution time and envoked search commands. The following data can be found:

- request_id
- timestamp
- skin
- query
- servlet execution time: How long did the servlet take in processing the search request.
- servlet output: If a special output was requested (e.g. output=rss).
- **search-command**: One or more fields, containing the type of search-command that got executed by the servlet, the number of hits it returned and the time spent in executing the command.

sesam.initialisations

Loggers involved in core initialisation of the engine, typically SiteKeyedFactories, also log to this file. Read this log file for startup and configuration deserialisation errors.
This logfile has no supported format.

sesam.dump

Logs a summary of every outbound http request made from sesat. This includes search commands, publishing fragments, and query evaluations.

This page last changed on Nov 20, 2008 by mick.

The Sesat-Interpreter

Basic usage

The Sesat-Interpreter is a simple interpreter that let's you add commands with some arbitrary code assosiated with it.

To enable the Sesat-Interpreter set/export SESAT_INTERPRETER=true

an example of how to add a command:

Bar.java

```
import no.sesat.Interpreter;

static {
Interpreter.addFunction("gc", new Function() {
public String execute(final Context ctx) {
System.gc();
return "GC requested";
}
public String describe() {
return "Request a gc run.";
}
});
}
```

When you start catelina from the commandline, and the interpreter has been enabled by setting the SESAT_INTERPRETER enironment variable to true, then you will see the functions as they are added.

Console output

```
..
Added function: test-arguments
Added function: help
Added function: gc
Added function: all-stacktraces
Added function: loggers
Added function: properties
Added function: save
Added function: read
Added function: macro
Added function: quit
Added function: sites
..
..
```

Now you can invoke a method by typing in it's name.

Console output

```
help
Functions available:
all-stacktraces
Look at all stacktraces.
echo
Echo arguments.
gc
Request a gc run.
help
```

```
Print this help message
loggers
Print active loggers, and set level if specified. 'loggers [regexp] [level]'
macro
Make a macro
properties
List System.getProperties()
..
..

test-arguments 1 2 query='en fisk'
Raw argument string:
1 2 query='en fisk'
Argument array: 2
1
2
Keywords:
QUERY --> en fisk
```

Most of the functions added, like the one in Bar.java at the top, are in static blocks. This means that the function will be added when the class is loaded. So all functions might not be available at once.

Aspectj

Some times you do not want to introduce code permanently, then you might want to use AspectJ to instrument the classes. Have a look at this aspectj class.

http://sesat.no/svn/sesat-commons/commons-interpreter-aspectj/trunk/src/main/aspect/no/sesat/interpreter/aspectj/QueryParserInspector.aj

If you want your aspect code to be active you need to include the interpreter-aspectj profile when building.

I use this:
-P include-fast,development,interpreter-aspectj

Javaagent

Enable this by setting JAVA_OPTS=-javaagent:/home/haavard/.m2/repository/sesat/commons-interpreter/2.0/commons-interpreter-2.0.jar

where home/haavard.... should be changed to wherever you have it.

Development Guidelines

This page last changed on Jan 20, 2009 by mick.

This page goes through building the Sesat-Kernel. It by itself is not enough to have a functioning search application. A complimentary tutorial exists on building the sesam.com skin.

Error formatting macro: toc: java.lang.NullPointerException

Get Sesat source

Check out the latest version of Sesat from the Subversion repository:

```
svn co http://sesat.no/svn/sesat-kernel/trunk sesat-kernel
```

Build Sesat

From the working copy of sesat-kernel run the Maven as follows to build the artifacts.
Make sure you have installed all the requirements.
Read Building with FAST to enable FAST APIs within Sesat

```
mvn install
```

java.lang.NullPointerException at com.sun.tools.javac.comp.Check.checkCompatibleConcretes(..)
When maven is building data-model-javabean-impl it fails with the javac crash:

```
[INFO] Compilation failure
Failure executing javac,  ...
java.lang.NullPointerException at
com.sun.tools.javac.comp.Check.checkCompatibleConcretes(..)
```

try checking out sesat-kernel again with a new name (eg sesat-kernel1) and building again. repeat until it works.
Or use Java7. It's a known bug http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6218229 related to inodes.

Generating schema files failed due to error: Java heap space
When maven is building generic.sesam/war it fails with:

```
[INFO] [sesat-mojo:searchModesSchemaGenerator {execution: default}]
[INFO] no.sesat.mojo.SearchModesSchemaGenerator
[INFO] Using: classpath =[snip]
Generating schema files failed due to error: Java heap space
```

Increase the heap size by adding parameters to MAVEN_OPTS like (for example)

```
export MAVEN_OPTS="-Xms512m -Xmx1024m -XX:MaxPermSize=256m"
```

build time
On a dual 2.2GHz machine w/ 4Gb ram & T10 connection the initial build took 6 and a half minutes 27. Subsequent full rebuilds took 1 minute.

Deploy to Container

Tomcat

By defining CATALINA_BASE the maven builds automatically deploys all warfiles into CATALINA_BASE/webapps/

Configure Container

Tomcat

UTF-8 Encoding

If your machine is not natively running unicode you have to set the CATALINA_OPTS environment variable: `export CATALINA_OPTS="-Dfile.encoding=UTF-8"`
You also have to make a change in the server.xml file as tomcat runs by default ISO-88591.
From:

```
<Connector port="8080" maxHttpHeaderSize="8192" maxThreads="150"
minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
redirectPort="8443" acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true"/>
```

To:

```
<Connector port="8080" maxHttpHeaderSize="8192" maxThreads="150"
minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
redirectPort="8443" acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" URIEncoding="UTF-8"/>
```

unpackWARs="true"

This setting, found in server.xml->Server->Host, cannot be set to false as Sesat requires write access into the deployed application.

port 8080

Note: You need to run your development tomcat on port 8080! (Or make your own maven development profile for your setup).

Browse

Deploy your skins to your container, start your container, and browse http://localhost:8080

** Every skin deployed must have its name added into your hosts file pointing to 127.0.0.1

This page last changed on Sep 11, 2008 by mick.

Building with FAST libraries

The FAST libraries are proprietary and so not part of the default Sesat build and not hosted in Sesat's maven repository.
To build the version of Sesat to be used against FAST libraries use instead:

```
mvn install -Pinclude-fast
```

You need your own access to the FAST jar libraries and must deploy them to your own maven repository

The FAST libraries require a few dependencies which cannot be bundled (due to classloading issues) into Sesat. Therefore it is required that the following jar's are copied (eg from your local maven repository) to $CATALINA_HOME/lib:

- commons-cli-1.0,
- commons-codec-1.3,
- commons-httpclient-3.0.1,
- commons-lang-2.3,
- commons-logging-1.1,
- dsapi-2.0.70,
- esp-searchapi-5.0.14,
- jscience-3.3,
- msearch-client-4.2,
- spring-1.2.1

The following script "deploy-fast-libraries-from-local-repository.sh" can be used on *nix:

```
#!/bin/bash
#
# Deploys the fast libraries and its dependencies into the tomcat lib directory, finding them from
the local repository.
#

if test -z "$1" ; then
echo "Usage: deploy-fast-libraries-from-local-repository.sh <path to tomcat-home/lib>"
exit 1;
fi

find ~/.m2/repository/ -name commons-cli-1.0.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name commons-codec-1.3.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name commons-httpclient-2.3.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name commons-httpclient-3.0.1.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name commons-lang-2.3.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name commons-logging-1.1.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name dsapi-2.0.70.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name esp-searchapi-5.0.14.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name jscience-3.3.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name msearch-client-4.2.jar -exec cp {} $1/ \;
find ~/.m2/repository/ -name spring-1.2.1.jar -exec cp {} $1/ \;
```

Kernel Operations

This page last changed on Nov 20, 2008 by mick.

"Feature Freeze". and creating the production branch.

Coming up to any release the code goes into a "feature freeze".
At this point all development improvements and new features must be completed (resolved) and tested (verified),
or remaining improvements and new features be bumped up to the next version so that the feature freeze can progress.

The code is then branched, with quality assurance testing and bug fixing continuing in this branch until the release, with this branch being referred to as the "production branch", while trunk becomes the development for the next version.

This is done using the maven release plugin like:

```
cd sesat-kernel
svn up
mvn release:branch -DbranchName=<current-version-without-SNAPSHOT-suffix>
```

Accepting all default prompt values.
It's wise to initialise svnmerge at this time.

It's typical to run the trunk code in the alpha environment (using the alpha profile), and the "feature freeze" or "production branch" in the beta environment (using the beta profile).
Accordingly please re-configure the Sesat-Kernel.productionBranch... projects in Hudson after any new production branch.

Updating a (alpha|nuclei|beta|electron|gamma|production) environment.

```
cd sesat-kernel
svn up
mvn -P<environment>,include-fast clean install sesat-mojo:deploy -DserverDeployLocation=<wagon-
style-remote-location> -Duser.name=<remote-location-username>
```

Sesat Kernel comes with seven "environment" profiles: development, alpha, nuclei, beta, electron, gamma, production.

Development is intended for use on a developer's own machine.
Alpha is used on a server cluster for trunk testing.
Nuclei is used on a server cluster for secondary trunk testing.
Beta is used on a server cluster for production branch testing.
Electron is used on a server cluster for secondary production testing.
Gamma is used on a server cluster for tertiary production branch testing.
Production is used on a server cluster that hosts the real production.
These are only recommendations, they can be used how you wish.

The default serverDeployLocation settings for each "environment" profile are to sesam's own server farm.

This page last changed on Feb 29, 2008 by mick.

Requirements

To build Sesat you need the following tools installed:

- Java SE Development Kit (JDK) >= 1.6 http://java.sun.com/
- Maven >= 2.0.6 http://maven.apache.org/
- Subversion >= 1.4.6 http://subversion.tigris.org/

To run Sesat you need a servlet container. Sesat has been tested on the following containers:

- Apache Tomcat 6.0 http://jakarta.apache.org/tomcat/

We do not currently support older versions than 6.0 out of the box. (SEARCH-4296)

This page last changed on Jan 25, 2008 by mick.

Development Platform Architecture

Sesat Software Requirements

See Software Requirements.
See Hardware Requirements.

Server Type Legend

Type	Description	RAM	Disk
Generic	Standard 2-CPU pizza-box	2Gb	3x72Gb disk-drives, 10krpm
FAST	Standard 2-CPU pizza-box	2Gb	Disc-cabinet with 10x72Gb disk-drives, 15krpm
DB	Standard 2-CPU pizza-box	4 Gb	3x144Gb disk-drives, 10krpm

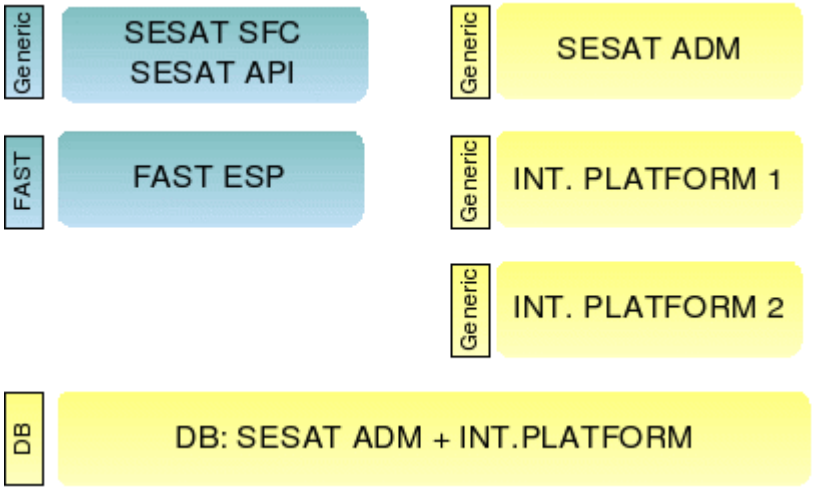
Minimal configuration for Proof-of-Concept

The hardware platform must match the following requirements, including "external" services used by SESAT (but not part of SESAT).

Servers	Type	Description	Software	Role
1	Generic	Search portal server running the search-application part of SESAT.	OS: CentOS, RedHat, Debian or other. App-server: Tomcat 5.5. Web-server: Apache2 (not required). SESAT is deployed on Tomcat as one or more .war-files.	Required in all SESAT installations.
1	FAST	FAST Search Engine for test indices.	FAST version 4.1 or 5.0, the latter is preferred. Sufficient for 5-10 mill. documents.	Required if not using only external content.
1	DB	Database-server for storing and preparing data to be fed to FAST, and also used if installing SESAT ADM	DB: MySql 5.0 or 5.1	**Not required if data is fed from the filesystem**, for example, or if all data is gathered externally.

1	Generic	Application Server for SESAT ADM	OS: CentOS. App-server: JBoss 4.1	Required if installing SESAT ADM.
1 or 2	Generic/DB	1 server: Integration server for collecting & washing external content.	Suggestion: Mule (Open Source)	Simple and powerful integration platform. We have no experience with Mule.
		2 servers: Business Objects Data Integrator.	FAST 5.0 and BODI	Powerful, commercial integration platform, used by Sesam.no.

The following illustrates a minimal set of hardware for performing PoC-development:



The blue servers are absolutely mandatory, while the yellow servers are optional depending on your requirements.

Single Site, Permanent Development Platform

The main difference between the "minimal" configuration and this configuration is the introduction of more FAST-servers and load-balanced servers for the search front and SESAT ADM servers. It is important to architecturally match the production platform, hence the load-balanced servers. Also, the number of FAST-servers will grow with:

- The amount of data to be indexed
- The number of different indices (for example, it is wise to have one server for Yellow and White pages data, and another server for Wikipedia indexing).

Configuration for multiple sites (Sesam.no, Sesam.se, etc.)

(Use Sesam.no/Sesam.se as example)

- In this example, we will look at the setup used by Sesam.no/Sesam.se. This includes different components for statistics, documentation, issue-tracking etc. Many of these components are not part of SESAT, but are included for convenience.

Hardware Requirements

This page last changed on Jan 16, 2008 by mick.

Common hardware recommendations

- Application servers: Dual-Core Dual-64bit-CPU, 8 gigs of RAM, RAID 5 setup, 72-144 gig local disks.

This page last changed on Jan 16, 2008 by mick.

SESAT Software Requirements

Note: SESAT supports FAST versions 4.1 and 5.0.

Main application software

Server role	Software-type	Software	Required
Webserver	Operative system	Linux RH 4 or Linux CentOS 4	yes
Webserver	Java application server	Tomcat 5.5	yes
Webserver	Proxying webserver	Apache 2.0 or Apache 2.2	no
Webserver	Java Virtual Machine	Sun JDK 1.6	yes
Devserver	Software Management	Subversion (latest version)	yes
Devserver	Software Management	Continuum (latest version)	yes

Software required for personalisation

Software-type	Software	Required
Directory service	OpenLDAP	yes
Persistence storage	MySql	yes

Other databases will be supported through JDBC.

This page last changed on Jan 25, 2008 by mick.

Development Using the APIs

Coming soon

Development using the SFC

This page last changed on May 12, 2008 by mick.

work in progress

For newcomers there's a tutorial on how to build sesam.com. This may prove more useful to begin with than the following documentation.

Development Using the SFC

Error formatting macro: toc: java.lang.NullPointerException

General Information

The SFC - SESAT Frontend Container - is the place to create vhosts that are using the full SESAT functionality.
All sites or applications running within the SFC also runs on the same JVM as the SESAT Kernel.

All code, configuration, images etc. related to a vhost are termed **resources**. These resources may be on the same file system as SESAT Kernel (in the Tomcat deployment catalogue), **or** they can be on some other server reachable by http.

In effect, SFC is just the notion of the JVM where we deploy and run applications using SESAT Kernel.

The following text will expose the details of how to create a site within the SFC.

Sesam examples

Documentation for starting a new sitesearch project: SiteSearch

Example documentation applicable to setting up Sesam Setting up Sesam

Structure of an SFC application

Catalogue hierarchy

The resources are found under the main catalogue. As for now we are using the following:

- /conf - here is the config files, with basic files as modes.xml and views.xml
- /css - the css files mainly reside in the /css/tabs folder which are default mapped from the id attribute for each tab in views.xml
- /images
- /javascript
- /templates - velocity templates and jsp files.

The hierarchy for where a sitesearch is placed is: generic.sesam.no -> genericno.sesam.no -> genericsitesearch.sesam.no -> ***sitesearch***.sesam.no (vg.sesam.no)

It's important to remember that every project inherits all the template-, css-, imagefiles and so on from the level above. That's a reason why we use the generic genericsitesearch.sesam.no to keep the sitesearch spesific stuff for itself.

Versioning

- versioning of css-templates, versioning of images etc., cache-control

Configuration files

- detailed description of the configuration files

HTML Templates - Velocity - JSP

> While not a requirement, most users would want to use HTML to encode search results. SESAT does not pose any strict requirements on the user, but we strongly recommend following international standards.

While SESAT support any "style" of HTML-coding, we recommend using XHTML 1.0 with the TRANSITIONAL doctype.
We recommend using the STRICT doctype, but with the extensively use of javascript today it would be impossible to make it valid.

The templates are defined through a custom templating system defined via the layout element in views.xml. Using these definitions work with the search:include tag or the searchTabInclude velocity macro.

Guidelines

We recommend the following:

- All element- and attribute names must be in lower case. All attribute values must be quoted.
- Close all elements. Example: ,
- Don't use deprecated elements and attributes http://www.html-reference.com/depreciated.htm
- Avoid using tables for layout
- Keep number of elements as low as possible (overuse of div's)
- Allways add alt-attribute in <img> element
- Allways set width and height attributes for images inline. Example:
- Encode ampersands in url's
- Use relative path instead of absolute where this is possible
- Make javascript unobtrusive where it's possible

- www subdomains - for example, we recommend http://sesam.no instead of http://www.sesam.no. All internal, and external links, should use the same way to link to sesam, not mixing the two. The Site class, responsible for handling the vhost always strips out any www. prefix.

The benefits by doing this are:

- Simpler development and maintenance
- Compatibility with futures web browsers
- Faster download and rendering of pages
- Better accessibility (for screenreaders and alternative browsing)

CSS

See Developing using CSS

Standard Taglibs

Navigators

See Navigation documentation

Enrichments

Simple tutorial Developing a quick and simple Enrichment tutorial

Developing a quick and simple Enrichment tutorial

This page last changed on Aug 30, 2007 by mick.

This tutorial will give an overview on the most basic implementation of an enrichment.
It will refer to the Idol enrichment implemented for sesam.no in 2007.

The Search Command

Each enrichment is just a result list in a condensed format presented like a single result hit within the pages main result list.
Therefore the enrichment requires a result list and this must come from a search command. We could use the StaticCommand which always returns a single dummy result item in the result list, but this example will use the PropertiesCommand that performs a simple search within a specified .properties file.

PropertiesCommand and the properties file

Configure modes.xml for the PropertiesCommand in your vertical's mode like:

```
<properties-command id="idol" properties-filename="idol"/>
```

Create a new file idol.properties, for example:

```
Idol=
Jan\ Fredrik\ Karlsen=
Benedicte\ Adrian=
Asbjørn\ Slettemark=
Mariann\ Thomassen=
```

The PropertiesCommand will return hits if any of the keys are found within the user's query.

Triggering the execution of the search command

For the vertical/mode to include the enrichment's search command to be run it must be specified in views.xml within the corresponding vertical/tab like:

```
<enrichment base-score="1000" threshold="0" weight="6" command="idol"/>
```

This enrichment does

- not refer to a rule since there is no analysis rules-based engine to determine the display, that is a result from the search command is the only determining factor,
- specify a base score and a threshold with the base score above the threshold to ensure its inclusion,
- specify a weight to ensure top placement of the enrichment over other enrichments, and
- specify the associated command to run.

> The term **mode** is used for the business definition of a vertical, and the term **tab** used for the view definition of a vertical

Visual Design of the enrichment

Create a velocity template in the skin's template/enrichments folder giving it a name equal to the command name.
For example:

```
<div class="enrichment">
<div class="enrich_img"><img src="/images/star.png" alt="stjerne" /></div>
<div class="enrich_text">
<b>Idol 2007</b><br/>
#foreach ( $item in $datamodel.getSearch($commandName).results.results )
So you think <strong>$item.getField('key')</strong> is going to win?
#end
</div>
<div class="clearFloat"> </div>
```

```
</div>
```

This page last changed on Jan 16, 2008 by mick.

Developing using CSS

General information

Css files are mainly created dynamically from the <tab> element in the views.xml file. The name of the css file is taken from the id attribute, and is imported in the code in head.vm. This means that every vertical/tab can have it's own style avoiding too much unnecessary style. It's also possible to add a css file manually, besides adding it directly in the head.vm, with inserting a css attribute in the <tab> element (without the css file extension), and to omit the stylesheet to be used you can add attribute called "display-css" and set this to "false". The principle that every tab inherits from the level above also applies for css. Another thing is that a style can be overidden by the more spesific a stylesheet.

Practical use

I will take the person-infopage as a quick example for how it works. Here it's three stylesheets, default, whitepages and whitepages-info. The default stylesheet is included in every tab and should contain styles that are in common, i.e link color, font-size, footer style and so on. (An exception for this is the individual frontpages for the tabs where the style is included directly in the vm file). Starting at the bottom level in the views.xml file we find the tab "whitepages-info". Here's the style for the role-tab which just exist for the person infopage and not the result page. As we see, this inherits data from "whitepages", where i.e the common "person things" like color and searchbar are. (this file will also contain result spesific things but you can't please all...). This tab inherits from "default-template", but here you will find the display-css attribute set to false so this file isn't included. At last this tab inherits from "default" and this file is added.

(At the time writing, the ps.css also exist, but I plan to remove this...)

Project independant css

There's a challenge on how we got things structured and with the inheritance which follows, and how we manage the css files (and templates accordingly). The thing we want to avoid is ofcourse making a stylechange one place, making an impact in another project that's not meant to be. We've come to the conclusion that projects in different countries should be independant of each other so they don't share css at all. But sitesearchprojects for a given country inherits stylesheets from the main project. Further there is way more similarities between the sitesearches and that's the reason why we have the genericsitesearch project which serves as a common central for the sitesearches. So sitesearches have a sitesearch.css that's acts like the default.css mentioned earlier. Common styles for both the main project and sitesearches, like i.e style for advertisement, should be in default.css.

This is how we try to keep things for now, but there may of course be other and better solutions in the future.

Guidelines

- Try to avoid inline css
- Naming css elements should be: word1Word2 (not word1:word2)
- Try minimizing style classes/id where it's practical i.e: text text Here 'class1', and really the whole element is excessive. Use ' #id1 span {} ' in stylesheet instead.

...and remember to remove style when codechanges make style redundant.

This page last changed on Jun 30, 2008 by mick.

HOWTO implement a RunHandler

POJO to hold configuration: AbcRunHandlerConfig.java

Create AbcRunHandlerConfig.java in run-handler-config project.
Implement getter and setters to store configurable options.

Controlling class: AbcRunHandler.java

Create AbcRunHandler.java in run-handler-control project.
Implement the handleRunningQuery method.

Configuration: modes.xml//modes/mode/run-handlers

Add the run-handler to the mode you wish it to be applicable to.
For example:

```
<modes>
<mode>
<run-handlers>
<abc bean-property-1="someValue" bean-property-2="someValue"/>
</run-handlers>
</mode>
</modes>
```

The name of the element inside <run-handlers> is the name of the configuration POJO minux the RunHandlerConfig suffix, and with CamelCase translated to typical xml format camel-case.

TroubleShooting

1. Make sure both run-handler-config & run-handler-control jars are bundled into your skin. For example inside the war like

   ```
   genericno.sesam.no!WEB-INF/lib/genericno.sesam.no-run-handler-config-2.18-SNAPSHOT.jar
   genericno.sesam.no!WEB-INF/lib/genericno.sesam.no-run-handler-control-2.18-SNAPSHOT.jar
   ```

Navigation documentation

This page last changed on Jan 16, 2008 by mick.

Contents

Error formatting macro: toc: java.lang.NullPointerException

Concepts

The navigation model is built around the object ExtendedNavigator. A ExtendedNavigator has basically four aspects viewed from the view generation side.

The core aspects of a navigators:

- **Selected**: Is this navigator selected? This is a indication if this navigator has been chosen by the user. In some cases a navigator can be chosen by default. This is controlled by configuration.
- **HitCount**: In many cases a navigator is a representation of a search. The hitcount indicates the expected number of hits you will get when you do this search. (At the time of writing this functionality only works for fast navigators)
- **Url**: The url to this navigation. To "select" this navigator, go to this url.
- **Title**: The title of this navigator. This should be used in the display text of the navigator.

Optimally all navigation on a page should be represented as ExtendedNavigators. If you do, you will be able to write more generic velocity code, and you don't need to worry about url generation in the view layer.

Configuration

Navigation is configured in xml. The configuration describes how searcResults are used to generate the hitCounters, and how the URL's should be built to navigate in and out of the page.

The navigation configuration concept was written in answer to all the view code made to solve navigation in a page.

If you want to add a navigation to a mode you need to add the navigation element to that mode. An navigation may contain one or more navigation elements. And that navigation in turn must contain a nav element, to define the navigator.

A nav element can be linked to a searchResult (by commandName), contain predefined options or even a combination.

Nav predefined options

If you want to add a pure predefined navigation, you just supply those option in the <nav> element. For a pure predefined navigator, hitCount is not set.

Example under shows how you can add a sort navigation.

```
<navigation>
<navigation>
<nav id="sort">
<option value="descending" display-name="nyest først"/>
<option value="ascending" display-name="eldst først"/>
</nav>
</navigation>
</navigation>
```

This configuration will present two navigators, on the key "sort". In velocity to access the defined navigators:

```
#set ($sort = $datamodel.navigation.getNavigation('sort'))
#foreach($sortType in $sort)
<a href="/search/?$sortType.url" >$sortType.title</a>
#end
```

Will result in the following html:

```
<a href="/search/?c=m&sort=descending"  >nyest først</a>
<a href="/search/?c=m&sort=ascending"  >eldst først</a>
```

The default behavior is that the "c" parameter will stick to the one active for the page you are using. You can override this by setting tab="something" in the nav and/or option.

```
...
<nav id="sort" tab="b">
<option value="descending" display-name="nyest først" default-select="true"/>
<!-- Tab on a option "overrides" the one in nav -->
<option value="ascending" tab="a" display-name="eldst først"/>
</nav>
...
```

The above velocity code will then result in the following html:

```
<a href="/search/?c=b&sort=descending" >nyest først</a>
<a href="/search/?c=a&sort=ascending" >eldst først</a>
```

Now we want to show that the default sorting is descending. So we need to set a default-select. This requires that the searchCommand actually uses descending sort if none was supplied.

```
...
<navigation>
<nav id="sort">
<option value="descending" display-name="nyest først" default-select="true"/>
<option value="ascending" display-name="eldst først"/>
</nav>
</navigation>
...
```

We change the velocity code to not make a link if a navigation is selected.

```
#set ($sort = $datamodel.navigation.getNavigation('sort'))
#foreach($sortType in $sort)
#if($sortType.selected)
<!-- Displaying the selected sort -->
$sortType.title
#else
<a href="/search/?$sortType.url" >$sortType.title</a>
#end
#end
```

Results in the following html:

```
<!-- Displaying the selected sort -->
nyest først
<a href="/search/?c=m&sort=ascending" >eldst først</a>
```

Linking navigation to a searchCommand

In many cases you want navigation to be dynamic, depending on the search that was run. At the time of writing, only fast navigators are supported from the searchCommands.

You need to define the navigators in the search command to make the navigators able to access them.

For a typical news search you, may want to display navigators per year. A typical configuration for this could be:

```
<mode>
```

```
<navigation>
<navigation command-name="newsSearch">
<nav id="year"/>
</navigation>
</navigation>

<clustering-esp-fast-command id="newsSearch">
<navigators>
<navigator id="year" field="year" name="yearnavigator" display-name="År" sort="YEAR"/>
</navigators>
</clustering-esp-fast-command>
</mode>
```

You need to "link" the navigation element to a command by supplying the command-name in the navigation element. When generating navigators, the navigation system will use the results from this search. It will look for a navigator in the searchResult, with id equal to the id of the navigation element.

Velocity code for displaying the navigator:

```
#set ($years = $datamodel.navigation.getNavigation('year'))
#foreach($year in $years)
#if($year.selected)
<!-- Displaying the selected year -->
<p>$year.title</p>
#else
<p><a href="/search/?$year.url" >$year.title</a> $year.hitCount</p>
#end
#end
```

Depends on the search, but an example result could be:

```
<p><a href="/search/?c=m&year=2007&nav_year=yearnavigator"  >2007</a> 583</p>
<p><a href="/search/?c=m&year=2006&nav_year=yearnavigator"  >2006</a> 1786</p>
<!-- Displaying the selected year -->
<p>2005</p>
```

As you see here, when you link navigation to a fast navigator you get two parameters. The year= and the nav_year=. If the navigators are for some reason available from the search command, and you want to suppress the nav_year=, you can set the parameter real-navigator=false on the nav element.

```
...
<navigation command-name="newsSearch">
<nav id="year" real-navigator="false"/>
</navigation>
...
```

Linking navigation with predefined options

In some cases it will make sense to only display chosen navigator elements with counters. This can be done if you link navigation to a Search, but supply the options that you want to display.

```
...
<navigation id="newscategory">
<nav tab="m" id="medium" command-name="newsSearchNavigator" real-navigator="false" out="true">
<option value="webnewsarticle" display-name="Norske nyheter"/>
<option value="printnewsarticle" display-name="Norske papiraviser"/>
</nav>
</navigation>
...
```

Combining navigation and the <reset-nav> element

So, if you want to combine both a navigation on year, and want the results be be sortable you just combine the configuration from the above examples:

```
<mode>
<navigation>
<navigation>
<nav id="sort">
<option value="descending" display-name="nyest først" default-select="true"/>
<option value="ascending" display-name="eldst først"/>
</nav>
</navigation>
<navigation command-name="newsSearch">
<nav id="year"/>
</navigation>
</navigation>

<clustering-esp-fast-command id="newsSearch">
<navigators>
<navigator id="year" field="year" name="yearnavigator" display-name="År" sort="YEAR"/>
</navigators>
</clustering-esp-fast-command>
</mode>
```

If you use this, navigation will "stick" on all urls. That mean that if you click on sort, that sort whill stick on all year urls generated for the page.

In some cases you don't want all navigation to stick on certain clicks. To support this behavior you need to use the <reset-nav> element.

In the example over we want the sort, to always be descending when you display a new result. So we want sort to reset when you click year. This can be done by adding a <reset-nav> to the year navigation.

```
...
<navigation command-name="newsSearch">
<nav id="year"/>
<reset-nav id="sort"/>
</navigation>
...
```

This page last changed on Nov 20, 2008 by mick.

The Search modes schema generator

To keep the modes.xml files correct we can use different schema files, and validate against them when writing the modes.xml files. In the sesat-mojo we have introdusced the searchModeSchemaGenerator which is capable of generating schema files for RelaxNG, XML Schema and DTD's.

Generating the schema files.

The use of this is done through the build script. See an example below.

pom.xml part-1

```
<plugin>
<groupId>sesat</groupId>
<artifactId>sesat-mojo</artifactId>
<executions>
<execution>
<configuration>
<outputDir>
src/main/conf/
</outputDir>
<classpaths>
<classpath>../query-transform-config/src/main/java/</classpath>
<classpath>../search-command-config/src/main/java/</classpath>
<classpath>../result-handler-config/src/main/java/</classpath>

<classpath>../../query-transform-config-spi/src/main/java/</classpath>
<classpath>../../search-command-config-spi/src/main/java/</classpath>
<classpath>../../result-handler-config-spi/src/main/java/</classpath>
</classpaths>
</configuration>
<phase>compile</phase>
<goals>
<goal>searchModesSchemaGenerator</goal>
</goals>
</execution>
</executions>
</plugin>
```

There is two ting to note in this configuration.

The first thing is where the generated files should be placed.

```
<outputDir>
src/main/conf/
</outputDir>
```

And the second thing is where the source files are located.

```
<classpaths>
<classpath>../query-transform-config/src/main/java/</classpath>
....
```

Validating during a normal build.

This is the a snippet showing how to turn validation on.

pom.xml part-2

```
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>xml-maven-plugin</artifactId>
<executions>
<execution>
<phase>test</phase>
<goals>
<goal>validate</goal>
</goals>
</execution>
</executions>
<configuration>
<validationSets>
<validationSet>
<dir>.</dir>
<includes>
<include>**/modes.xml</include>
</includes>
<excludes>
<exclude>**/target/**</exclude>
</excludes>
<validating>true</validating>
</validationSet>
</validationSets>
</configuration>
</plugin>
```

Validating in your favorite editor.

Normally this will just work.

This page last changed on May 08, 2008 by ssmathyr.

Tutorial, and best practices, on using Ajax in templates

The simple way

The RunningQueryImpl has a number of supported parameters that, in combination with using various layouts per tab (which can also be called a "vertical"), can be used to build a rich ajax client.

Each tab/vertical has a connection to one mode. In this mode there is a list of commands to run.
By default all commands are run unless they are associated to enrichments from the tab, and the enrichment's rule achieved a score meeting the enrichment's threshold. Enough about enrichments, that's a story for another day.

The *commands* parameter

This default behavior of which commands to run within the RunningQueryImpl can be explicitly, and per request, defined via the "commands" parameter.
For example, this search asks for *only* the defaultSearch command to run on sesam.no which is why you see "?" for the hit count on all the other services. The commands parameter takes a comma separate list when you want to request multiple commands within the mode to be run.

The *waitFor* parameter

By default the RunningQueryImpl waits until all commands have been executed. This need not be the case though if you give, in modes.xml, the command the attribute *asynchronous="true"*. If a command is attributed such and you want to be guaranteed to see the results you must explicitly state that you are willing to wait for the command to finished by adding the parameter for example *waitFor=defaultSearch*.

The waitFor parameter can be used in a number of ways:

- *commands=defaultSearch&waitFor=defaultSearch* explicitly asks the command to be run, and presuming that it is attributed as an asynchronous command, and to wait until it is finished.
- *waitFor=defaultSearch* asks to wait for the command to finish. This presumes that the command is included in the default list of commands to run but is attributed as asynchronous.
- *commands=&waitFor=defaultSearch* disables all commands from running but asks to wait for the command to finish. This presumes that a previous request started the command. This demonstrates how commands are stored in the datamodel between requests and how asynchronous commands can be initiated by the original request, but not used in that request's output, and their results used via a second request, eg an ajax call, later on.

The *layout* parameter

Finally all this can be used with alternative layouts by suppling the *layout* parameter.
Each tab contains a default layout that is in views.xml the tab without an id attribute. With the layout parameter defined a layout with a matching id is looked for and used instead. If it can't be found the default is still used.
This allows a pseudo "portals" approach to the templating system. For example the request http://sesam.no/search/?q=sesat&commands=defaultSearch&layout=defaultSearchlayout
with the additional layout

```
<layout id="defaultSearchLayout" main="/fragments/layout/results/defaultSearch"/>
```

would render only the main results list on the page.

This above given layout doesn't have any embedded include templates. Since the result list is all rendered with just defaultSearch.vm template it can be supplied directly, in an absolute manner (see the initial / in the path), as the *main* template.

Examples and usages

This section won't include much code, but will explain in more detail how AJAX can be used. For these examples we'll use the following configuration:

views.xml

```
<view id="ajax-test" key="at" mode="ajax-test" inherit="default-mode">
<!-- default layout for normal requests -->
<layout main="main">
<include ...>
[...]
</layout>

<!-- layout for the ajax request using the ajaxtest template-->
<layout id="ajax" main="ajaxtest"/>
</view>
```

modes.xml

```
<mode id="ajax-test" inherit="default-mode">
<yahoo-web-command id="defaultSearch" inherit="default-yahoo-web-command" asynchronous="true"/>
<yet-another-search-command id="yasc" inherhit="default-yasc"/>
</mode>
```

There's a few different ways AJAX/SESAT can be used to enhance a web page. What's probably most useful is the ability to fetch slow results using a second request.

Note: This is currently broken and probably won't work as expected until SESAT 2.18. See http://sesat.no/scarab/issues/id/SKER4737

The following steps will show a typical use of this:

1. The users browser makes a request to /search/?c=at&q=sesat
2. SESAT executes all search commands. Since defaultSearch is marked as asynchronous it's not waited for by SESAT, so if it' not finished by the time the other search commands are it's not included in the results returned to the browser.
3. The users browser executes javascript code that was returned in the first response and does an AJAX request to /search/?c=at&q=sesat&commands=&waitFor=defaultSearch&layout=ajax
4. SESAT fetches the search result for defaultSearch that was executed by the previous request and returns it to the browser using the templates associated with the layout with id ajax.
5. The users browser dynamically adds the defaultSearch result to the web page.

Another use would be to only fetch the results of specific search commands:

1. The users browser makes a request to a given web page
2. The users browser executes javascript code that was returned in the first response and does an AJAX request to to /search/?c=at&q=sesat&commands=yasc&layout=ajax
3. SESAT executes only the yasc command and wait for it to finish before returning the result to the browser.
4. The users browser dynamically adds the yasc result to the web page

The hard way

Enable DWR (the dependency and the servlet). This serialises the whole datamodel through the request so it is available in the client's javascript. This is a far more advanced approach when much more logic is required on the client side. DWR is disabled by default in Sesat as we suspect the simpler approach is well, simpler.

Comments

I've been asked by Steinar to add a comment here, and after performing a lexographical analyzis (counting occurneces of the string 'ajax') it strikes me that this is probably the least ajax-centric ajax-tutorial I've read (and I've only read a few so that says close to nothing, really  However, the

reference to RunningQueryImpl was quite useful. But for me who doesn't yet see the big picture (tutorial readers seldom do) it'd be useful with an explanation of how ajax actually is used to tie it all together.

Thx 

Posted by smjonms at May 06, 2008.

Tutorial - Building Sesam.com

This page last changed on Mar 12, 2009 by mick.

This tutorial will go through building a Sesat Skin from scratch to completion.
It repeats the basic building of the "sesam.com" skin found at sesat-kernel/generic.sesam/sesam.com

6 easy steps to build your own search engine website.

Error formatting macro: toc: java.lang.NullPointerException

Create the project, a la the "Skin"

We will create the new skin from an existing archetype found within sesat-kernel. So we must checkout and build sesat-kernel & its skin-archetype first.

```
svn co http://sesat.no/svn/sesat-kernel/trunk sesat-kernel
cd sesat-kernel
mvn install -DskipTests=true
cd skin-archetype
mvn install -DskipTests=true
```

Now we can create our new skin:

```
cd ../..
mkdir test-skin
```

You should now have:

```
\-|
|-sesat-kernel
\-skin-archetype
|-test-skin
```

Then do (you should replace *-Dversion=2.18-SNAPSHOT* with the version you see in sesat-kernel/pom.xml):

```
cd test-skin
mvn archetype:generate -DarchetypeGroupId=sesat -DarchetypeArtifactId=sesat-skin-archetype \\
-DarchetypeCatalog=local -DgroupId=sesat -DartifactId=pom.sesam.com \\
-Dversion=<current sesat-kernel version>-SNAPSHOT -Dpackage=no.search.sesat
```

From Author: make sure you are using maven-archetype-plugin:2.0-alpha-3 or greater. Versions before that have some confusion between archetype:generate and archetype:create goals. Take a peek in ~/.m2/repository/org/apache/maven/plugins/maven-archetype-plugin/ if unsure. Add "-up -U" to the mvn archetype:generate command above to update to a later version.

You should now have under test-skin a directory "pom.sesam.com" and under that the files:

```
LICENSE.txt  pom.xml  query-transform-config  query-transform-control  result  result-handler-
config  result-handler-control  search-command-config  search-command-control  velocity-directives
view-config  view-control  war
```

Troubleshooting
If these directories do not exist then "mvn archetype:generate ..." didn't work properly. You can workaround this by extracting the appropriate pom.sesam.com-X.Y.tar into the test-skin directory.

You can now cd into this new maven project that has all the possible skin components set up ready to go.
This tutorial will really only deal with the "war" submodule.
Try building the new maven project:

```
cd pom.sesam.com; mvn install -DskipTests=true
```

Some users report that *-DskipTests=true* doesn't work, if so revert to the older and deprecated *-Dmaven.test.skip=true*.

Create Business definition: modes.xml

We are going to use yahoo's search for our main result list. This is purely business logic so it all goes into modes.xml
Add to test-skin/pom.sesam.com/war/src/main/conf/modes.xml (inside the <modes></mode>):

```
<mode id="default-mode" analysis="false" inherit="default-magic">
<yahoo-web-command id="default-yahoo-web-command"
appid="YahooDemo"
field-filters="site"
host="yahooWebHost"
language="en"
port="yahooWebPort"
result-fields="Title AS title,Summary AS body,Url AS url,ClickUrl AS clickurl"
results-to-return="10">
<result-handlers>
<field-chooser target="title" fields="title,url"/>
<regexp field="url" target="site" regexp="http://([^/]*)/?"/>
</result-handlers>
</yahoo-web-command>
</mode>
<mode id="international" evaluation="false" inherit="default-mode">
<yahoo-web-command id="globalSearch" inherit="default-yahoo-web-command"/>
</mode>
```

This defines two "modes". The first "default-mode" is both the template to build off and the guts of the configuration to the yahoo search command.
The second, "international", is the mode we'll actually be working with but it just inherits everything from "default-mode" for now. This layering approach is often a good idea for good house-keeping.

There are three result-handlers configured in this example. Result-handlers are used to manipulate (post-process) results once they have been fetched.
find-file-format adds a field to each result whose value is the mimetype guessed from the result's URL (which comes from the *clickurl* field).
field-chooser will assign the *title* field if it is null the value of the *url* field. A useful fallback mechanism.
regexp will assign the *site* field the regular expression capturing match against the *clickurl* field.

Note: in our repository version of generic.sesam/sesam.com/war/src/main/conf/modes.xml you'll see the yahoo-web-search elements commented out and replaced with similar yahoo-idp-search elements. The YahooIdpSearchCommand is a more powerful search interface to yahoo's indexes than their "Contextual Web Service", but you need to enter a contract with Yahoo to use it.

Create Presentation definition: views.xml

The configuration for how these results from yahoo will appear on the page comes from views.xml.
A "tab" does not have to be a tab in UI terms, here it simply refers to a page, tab, or vertical that layers on top of one of our modes.
Add to war/src/main/conf/views.xml (inside the <views></views>):

```
<tab id="default-sesam-com" inherit="default"/> <!-- placement for a global skin css -->
<tab id="international" inherit="default-sesam-com"
key="g"
mode="international"
page-size="10"
rss-result-name="globalSearch">
<navigation>
<navigation id="offset">
<result-paging id="offset" command-name="globalSearch" page-size="10" number-of-pages="10"
hitcount-source="totalhits"/>
</navigation>
</navigation>
<layout main="sesam.com/main.jsp" front="sesam.com/index.jsp">
<include key="header-element" template="sesam.com/head.jsp"/>
<include key="header-element-extra" template=""/>
<include key="top-col-one" template="sesam.com/searchbar-top.jsp"/>
<include key="main-col-three" template="sesam.com/globalSearch.jsp"/>
```

```
<include key="bottom-col-one" template="sesam.com/offsetPager.jsp"/>
<include key="bottom-col-three" template="sesam.com/footer.jsp"/>
<include key="bottom-col-two" template="sesam.com/searchbar-bottom.jsp"/>
</layout>
</tab>
```

Skin properties

war/src/main/conf/configuration.properties contains skin defining properties:

```
# Site attributes
site.parent=@sesam.site.parent@
site.locale.default=en
site.issitesearch=false
site.defaultTab=g
yahooWebHost=search.yahooapis.com
yahooWebPort=80
# disable fast token evaluation. HACK while we wait for SEARCH-3540 Anonymous TokenPredicates &
Token Evaluator SPI.
tokenevaluator.port=0
```

The filtered properties come from the pom.xml
There six different profiles (in addition to the default "development" profile) that can be used for server testing and production environments as you wish.
The property here *sesam.site.parent* is crucial and must match the name of the web application, as deployed, that is the parent skin.
The yahoo properties relate to the host and port attributes previously mentioned in the yahoo-web-search element in the modes.xml.
The tokenevaluator.port property is set so to disable the FAST token evaluation while SEARCH.3540 is still in progress.

These default values the archetype created can be used in this tutorial.

Design templates

Each of the templates specified in views.xml must be created, along with the main.jsp and index.jsp. The main template's path is relevant war/src/main/webapp/pages/. The templates are all relative to war/src/main/webapp/fragments/layout.

These templates are rather self explanatory so links are given to each:
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/pages/sesam.com/index.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/pages/sesam.com/main.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/fragments/layout/sesam.com/searchbar-top.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/fragments/layout/sesam.com/globalSearch.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/fragments/layout/sesam.com/offsetPager.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/fragments/layout/sesam.com/footer.jsp
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/webapp/WEB-INF/classes/fragments/layout/sesam.com/searchbar-bottom.jsp

These templates are internationalised, the message values (from <search:text.. tags) can be found in war/src/main/conf/messages_en.properties:
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/conf/messages_en.properties

Design css

Some of the css definitions come from the existing css in generic.sesam:
http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/war/src/main/css/tab/default.css

Everything specific to the sesam.com skin is found in the default-sesam-com.css, which is the reason there exists the "default-sesam-com" tab in views.xml as each ancestor in the tab's ancestry imports a matching css:

http://sesat.no/source/xref/Sesat.Kernel..trunk/generic.sesam/sesam.com/war/src/main/css/tab/default-sesam-com.css

Finished!

Yes it's all that easy.
If you define the system environment variable CATALINA_BASE when you build ("mvn install") the warfiles should have automatically been copied into $CATALINA_BASE/webapps/
So, after building, inside the webapps directory there should exist: ROOT.war, generic.sesam.war, localhost.com.war
Given that you have something like "127.0.0.1 localhost.com" in your /etc/hosts file you should now be able to fire up your container (eg tomcat) and browse http://localhost.com:8080/

For more documentation on skin development see Development using the SFC.

This page last changed on May 06, 2008 by mick.

FAQ

Questions and answers on our mailing lists

Error formatting macro: toc: java.lang.NullPointerException

License

How to keep private code, or 'Trade Secrets', while using the Free Software License?

Only code directly using the Sesat APIs (or SPIs) is considered derivative work. So for example code that is using an rss/xml/ajax interface to Sesat is not a covered work but be an interactive client. Skins, for example java classes within the skins, will be covered work.
Backend systems, from administration portals to databases and indexes, and any code layering on top of this of course may remain private, and can be used within the search front end in a manner similar to other propriety/private libraries that the Sesat code uses, for example the FAST java libraries.
The most practical approach to keep private code often would be to keep propriety/private code in a separate project and include it in the Sesat skin as a jar library dependency. This maintains a clean design between framework code and the underlying data and/or schema details. This is described as an aggregated work in §5 of the license.
You are also free to inquire about a Propriety License, aka a contract with Schibsted Søk AS to waiver the free license. We expect bigger companies will take this approach just out of convenience (rather than any real objections to the free license).

Documentation

Where is the "monolithic" or "all-in-one" documentation?

We do not provide a monolithic documentation page. The main reason we see for such a page is for searching for something when you know it is in the documentation but not what section. Since the website has a search form at the top of every page just for searching the website, and we are a search technology project, we'd rather you use that.

Versioning

Where is kernel version 1.0?

Versions up to 2.15 where done under private development. Version 3.0 of the sesat-kernel can be considered the initial api-stable release as open source projects usually consider the 1.0 release to be. See http://www.producingoss.com/en/development-cycle.html#release-number-simple-strategy

HowTo Solr Query Evaluation

This page last changed on Jan 31, 2009 by mick.

associated to
http://sesat.no/scarab/issues/id/SKER4952

Setting up Sesat's Query Evaluation with a Solr index

Preface

Sesat comes with a query evaluation that occurs during the query parsing stage. Its purpose is to evaluate metadata against each clause (leaf and operator) within the currently constructed Query tree. The evaluation has various types.

Current implementations are clauses matching regular expressions (RegExpTokenEvaluator), mathematical expressions (JepTokenEvaluator), hits against a FAST Query Matching server (VeryFastTokenEvaluator), and hits against a Solr server (SolrTokenEvaluator).

Some examples of metadata used at sesam.no are first_name, last_name, full_name, company_name, english_words, geological_province, geological_city, geological_suburb, and geological_street. All of these metadata examples are stored in a Solr index as some of them are very large, eg the full_name list contains the national register of the norwegian population - roughly 5 million names.

After the Query tree has been constructed, and the metadata associated to each clause, this can be used to maximise the efficiently of which searches to execute and end up federating. For example there is no need for us to initiate a search against our 'white pages' (or people catalogue) if neither a full_name, or first_name + last_name combination, metadata exists within the query. Another example is our 'yellow pages' (or company catalogue) searches can be enhanced when we know which clauses within query are geological terms.

It can be seen why we, at sesam.no, see this query evaluation as a crucial part of a federating search engine.

Introduction

Our query evaluation against large data lists used to be via the (VeryFastTokenEvaluator) that works against a FAST Query Matching server. In a desire to move away from a proprietary closed solution that left us at the mercy of FAST consultants and towards an open solution that we fully owned and were free to share we decided to re-implement all this functionality with a Solr index. What follows is our installation and setup of a Solr index to successfully work with our (SolrTokenEvaluator) using the english_words metadata as the example.

Configuration

In Sesat's base skin "generic.sesam" you'll find war/src/main/conf/SolrEvaluators.xml containing:

```
<solr-evaluators>
<list token="ENGLISHWORDS" list-name="common_english"/>
</solr-evaluators>
```

This declares the metadata (also referred to as a Token or TokenPredicate) ENGLISHWORDS, and connects it to hits in the solr index with list_name=common_english.
This is all that is required to setup evaluation for english_words metadata.

We also need to configure where the Solr index can be found.
In the same skin you'll find war/src/main/conf/configuration.properties with the line:

```
tokenevaluator.solr.serverUrl=@tokenevaluator.solr.serverUrl@
```

The @tokenevaluator.solr.serverUrl@ is filtered from values defined in the skin's pom.xml according to the current profile in action.

For the development profile this reads:

```
<tokenevaluator.solr.serverUrl>http://localhost:16000/solr</tokenevaluator.solr.serverUrl>
```

So you can either run Solr locally on port 16000 or create a ssh tunnel on your port 16000 to another Solr server.
Note: the values for the other profiles point to a host "sch-solr-test01.dev.osl.basefarm.net". This is our own Solr server and so naturally won't work for you - you'll need to override this setting.

Along with this configuration it's now presumed that you have sesat skin up and running. If you don't read Tutorial - Building Sesam.com for help on how to.

Solr Installation

You'll need a recent version of Solr, 1.4, or latter, so that the two patches found in https://issues.apache.org/jira/browse/LUCENE-1380 and https://issues.apache.org/jira/browse/SOLR-763 are included.

Deploy the solr.war to your container. Before starting it you'll need to configure your Solr Home.
It is fine to use the example Solr home found in example/solr but replacing schema.xml with:

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema name="example" version="1.1">
<types>
<fieldType name="string" class="solr.StrField" sortMissingLast="true" omitNorms="true"/>
<fieldType name="date" class="solr.DateField" sortMissingLast="true" omitNorms="true"/>
<fieldType name="shingleString" class="solr.TextField" positionIncrementGap="100" omitNorms="true">
<analyzer type="index">
<tokenizer class="solr.KeywordTokenizerFactory"/>
</analyzer>
<analyzer type="query">
<tokenizer class="solr.WhitespaceTokenizerFactory"/>
<filter class="solr.ShingleFilterFactory" outputUnigrams="true" outputUnigramIfNoNgram="true"
maxShingleSize="99" enablePositions="false" />
</analyzer>
</fieldType>
<fieldtype name="ignored" stored="false" indexed="false" class="solr.StrField" />
</types>


<fields>
<field name="id" type="string" stored="true" required="true" />
<field name="list_name" type="string" indexed="true" stored="true"/>
<field name="list_entry" type="string" indexed="true" stored="true"/>
<field name="list_entry_shingle" type="shingleString" indexed="true" stored="true"/>
<field name="list_entry_synonym" type="string" indexed="true" stored="true"/>
<field name="timestamp" type="date" indexed="true" stored="true" default="NOW" multiValued="false"/
>
</fields>

<uniqueKey>id</uniqueKey>
<defaultSearchField>list_entry_shingle</defaultSearchField>
<solrQueryParser defaultOperator="OR"/>
</schema>
```

Now start the container and unzip and feed in the solr xml document add_english_words.xml.gz that contains all the english words.

Coding with the metadata: TokenPredicates

Everything should now work.

When a query is parsed WordClauses within the Query that match one of the english words with contain within the clause's knownPredicates list the TokenPredicate ENGLISHWORDS. For example:

```
true == clause.containsKnownPredicate(Categories.ENGLISHWORDS)
```

Known and Possible Predicates?

What is a possible predicate?

Sometimes metadata is position dependent, that is the position of the term within the query has the final say whether the metadata is really applicable. This can be used with the regular expression evaluators, but also every TokenPredicate has an "exactPeer" which is only ever true if the whole query has the metadata. We cannot assign such metadata definitely to the root clause of any query because clauses are immutable and used in a fly-weight pattern across multiple Query trees.

Upgrade Guides

This page last changed on Jan 25, 2008 by mick.

2.16 Upgrade

This page last changed on Apr 03, 2008 by mick.

Error formatting macro: toc: java.lang.NullPointerException

Compatibility changes

Enrichment placements (14 Feb 2008)

SEARCH-3654 Generalized enrichment handling

Removed APIs:

- SearchTab.getEnrichmentLimit()
- SearchTab.getEnrichmentOnTop()
- SearchTab.getEnrichmentOnTopScore()
- EnrichmentHint.isAlwaysVisible()

Added APIs:

- EnrichmentPlacementHint class
- EnrichmentHint.getProperties()

Changed API:

- AbstractEnrichmentDirective.placementCorrect(..)

The logic on how many enrichments to show and enrichment thresholds have been moved, from being static to a `SearchTab`, into the `EnrichmentPlacementHint` class.
The class is configured with the `<enrichment-placement>` element in views.xml.
Any vertical, or `SearchTab`, may have as many `<enrichment-placement>` as desired. It is intended that `EnrichmentPlacementHint` is used primarily by `EnrichmentDirective` implementation.
Both `EnrichmentHint` and `EnrichmentPlacementHint` may contain arbitrary properties. The properties for `EnrichmentHint` are passed into the corresponding search command result list as fields.
`EnrichmentHint.isAlwaysVisible()` was removed, it's /reverse/ functionality re-implemented in genericno.sesam.no/velocity-directives's `EnrichmentDirective` as this was the sole place where it was false, ie `always-visable="false"`.
`AbstractEnrichmentDirective.placementCorrect(..)` method signature changed to take for the first parameter a `DataModel` instance instead of a `SearchTab` instance. This gave the implementation of the method more flexibility as `SearchTab` was easily derived from the `DataModel`.

New methods for offset / SearchCommand.isPaginated() (3 Mar 2008)

SEARCH-3733 New methods for offset
SEARCH-4264 Nyhetssøk, nettsøkberikelsen tar i bruk offset fra URL-en
Offset methods moved away from junkyard to using 'offset' navigator.
No longer any need for SearchConfiguration's isPaging() or isIgnoreOffset() methods.

Added APIs:

- boolean SearchCommand.isPaginated()
- int AbstractSearchCommand.getOffset()

Deprecated APIs:

- int AbstractSearchCommand.getCurrentOffset(int i) :: Use instead getOffset() +i

Removed APIs:

- SearchConfiguration.isPaging()
- CommandConfig.setPaging()
- NewsEspCommandConfig.isIgnoreOffset()
- NewsEspCommandConfig.setIgnoreOffset()

AbstractSearchCommand.getParameter(..) no longer uses junkyard (5 Mar 2008)

Removed APIs:

- AbstractSearchCommand.getParameters()
  Use AbstractSearchCommand.getParameter(parameterName) or datamodel.getParameters().getValues() instead.

New Features

Requirements

Minimum Java required bumped to Java6 (28 Feb 2008)

Sesat-kernel/sitemap-generator uses ServiceLoader to load PageProviders from skin sitemap jarfiles.

Minimum Tomcat required bumped to Tomcat-6 (28 Feb 2008)

META-INF/context.xml unpacking during deployment on tomcat-5.5 kills the application. See SEARCH-4296 sesat r6123 incompatible with tomcat-5.5

2.17 Upgrade

This page last changed on Sep 26, 2008 by mick.

Error formatting macro: toc: java.lang.NullPointerException

Compatibility changes

Enable sesat templating system to accept JSP templates (17 Feb 2008)

SEARCH-4294 Enable sesat templating system to accept JSP templates
Changed API:

- AbstractVelocityTemplateTag.importTemplate(..) renamed to importVelocity(..). method parameters unchanged.

Anonymous TokenPredicates (2 April 2008)

SEARCH-4421 Anonymous TokenPredicates
TokenPredicate is now a interface. It's inner classes provide all functionality: Categories for the existing hardcoded enumerations, Static for static-utility methods, and TokenPredicateImpl for anonymous TokenPredicates.
Changed API:

- All of TokenPredicate's existing enums have been moved into TokenPredicate.Categories
- All of TokenPredicate's static methods have been moved to TokenPredicate.Static

Added APIs:

- TokenPredicate.Static.getAnonymousTokenPredicate(String) :: finds an existing anonymous TokenPredicate
- TokenPredicate.Static.createAnonymousTokenPredicate(String, Type) :: create a anonymous TokenPredicate
- TokenPredicate.Static.getTokenPredicates(Type) :: retrieve all tokens by type

Removed APIs:

- TokenPredicate.ALWAYSTRUE :: had been deprecated for quite some time already.
- TokenPredicate.getFastTokenPredicates() :: use TokenPredicate.Static.getTokenPredicates(Type) instead

Deprecated APIs:

- TokenPredicate.getMagicTokenPredicates() :: norway specific logic
- TokenPredicate.getTriggerTokenPredicates() :: norway specific logic

index.jsp removed

Any root request (ie "/") is now automatically forwarded to "/search/?".
So instead of using the index.jsp -> index.vm mechanism the layout's front defined template is used.
If your front page is broken you can most likely fix it just by adding *front="index"* to that tab's layout element in views.xml
For the "/" case, where neither the c nor q paramter is defined make sure you've set site.defaultTab in configuration.properties.

Base skin now called "generic.sesam"

Whereas before it was "generic.sesam.no".
The same goes for the profile counter parts: generic.alpha.test.sesam, generic.nuclei.test.sesam, generic.beta.test.sesam, generic.electron.test.sesam.
It is wise now to explicitly define the port in the sesam.site.parent property in any skin extending generic.sesam, since generic.sesam isn't a valid domain and might not be successfully handled by any front end proxy, eg apache, handling the default port 80.

Gamma profile now an almost identical copy of the Production profile

Allows multiple tomcat instances to be run behind the one proxy front making it easy to switch between profiles. Useful for supporting multiple version of sesat-kernel under the one domain.

ControlLevel.RUNNING_QUERY_RESULT_HANDLING renamed to RUNNING_QUERY_HANDLING (5 Jul 2008)

Changed API:

- ControlLevel.RUNNING_QUERY_RESULT_HANDLING now ControlLevel.RUNNING_QUERY_HANDLING

Change the syntax of XML written to the logfile sesam.access (07 Jul 2008)

Commit r6724
Instead of using the parameter's key, as the name of the tag, we now use the fixed tag <parameter> for this. The key and value are added as parameters inside the parameter tag.
So we have gone from this:
<1_wonkey_name>123</1_wonkey_name>
to
<parameter key="1_wonkey_name" value="123" />
Changed API:

- sesam.access logfile

New Features

Using JSPs in skins (16 Feb 2008)

SEARCH-4290 Design and code with JSPs in skins.
It is now possible to use both JSPs and Velocity templates in the skin's templates/ folder.
The full documentation on this feature is in the javadoc for SiteJspLoaderFilter

AllParametersDirective added (2 Jun 2008)

SEARCH-4760 Replace usages of ${request.requestURL} in velocity templates with AllParametersDirective.
Velocity templates are encouraged to be $request free and so http-implementation independent.
So now $request.queryString can be replaced with #allParameters()
It is also an encoding safe way of writing out the current URL's query string in correct UTF-8 encoding if the original request was ISO encoded.

SearchMode.isAutoBroadening() added (16 Sep 2008)

By adding the attribute auto-broadening="false" the query will not be wrapped and re-executed when all commands return zero hits. Default value is true leaving functionality as is today.
See SKER5004 Create option to disable the automatic rewrite of <multi-word query> to (<multi-word query>) when there are zero hits across all verticals

Rss mode can executed multiple commands (5 Jul 2008)

In views.xml/views/tab@rss-result-name multiple commands can be entered each separated by commas. The first will be used by the SyndicationGenerator and will continue to be the return result from SearchTab.getRssResultName(). An extra method has been added SearchTab.getRssCommands() that returns the whole list. This method is used by RunningQueryImpl to ensure all specified commands are executed in rss mode.

CopySearchRunHandler (5 Jul 2008)

A whole SearchDataObject in the datamodel can be copied with a new name.

FederatorRunHandler (5 Jul 2008)

Easily configurable run handler to merge results from multiple search command together into one ResultList.

Requirements

Renaming of news case facade (16 Sept 2008)

SEARCH-4987 RMI doesn't work in alpha or beta
Changed API in newsadmin-services:

- NewsCaseFacadeInterface renamed to NewsCaseFacade

The sesat revision r6808 requires an upgrade of newsadmin-services from v1.4 to v1.5 on the RMI backend server.

2.18 Upgrade

This page last changed on Oct 30, 2008 by sshafroi.

Error formatting macro: toc: java.lang.NullPointerException

There also exits a presentation running through the new features in 2.18.

Compatibility changes

Sorting parameter only applicable to search command with sorting navigator (26th May)

SKER4750 - Only main search should use sort order set in the URL
Implemented behavior similar to SearchCommand.isPaginated() & AbstractSearchCommand.getOffset()
Added API:

- SearchCommand.isUserSortable() returns true for search command's that have an applicable navigator, ie id="sort"
- AbstractSearchCommand.getUserSortBy() returns the userSortBy parameter if true == isUserSortable()
  It's now mandatory that the sort navigator defines the search command applicable via the command-name attribute.

SearchConfiguration subclasses must meet javabean specification (26th June)

Issue SKER4404: (Automatically assign config settings in readSearchConfiguration where there is a setter)
Changed API:

- no.sesat.search.query.transform.AgefilterQueryTransformerConfig's config attribute changed from field to age-field.
- no.sesat.search.mode.config.SearchConfiguration's getResultFields changed to getResultFieldMap.
- no.sesat.search.mode.config.SearchConfiguration's getFieldFilters changed to getFieldFilterMap.

Token Evaluator SPI (7th July)

SKER3540 - Token Evaluator SPI
TokenPredicate exploded into more stable objects. Evaluators no longer hardcoded into kernel. Evaluators are loaded through their corresponding AbstractEvaluatorFactory implementation. Factories are registered via configuration.properties, and answer whether they are responsible for a TokenPredicate through isResponsibleFor(tokenPredicate).
The TokenEvaluator is fetched through the factory as well through getEvaluator(tokenPredicate). The factory encapsulates whether the one evaluator can be used for all queries, and/or for all tokens.
The skin's project for evaluator implementations is query-evaluation, and the kernel loads classes through Spi.QUERY_EVALUATION.
Added API:

- new class AbstractEvaluatorFactory
- factory implementations JepEvaluatorFactory, FastQueryMatchingEvaluatorFactory, and RegexpEvaluatorFactory added
- new class AbstractTokenPredicate (solid equals and hashCode overrides to satisfy interface constraint that name is unique across all implememtations - allowing exactPeer() to be used more exactly)
- AbstractEvaluatorFactory.Context requires method "String getUniqueId()" as datamodel.parameters.uniqueId is not available. This contextual addition creeps back through TokenEvaluationEngine.Context and AnalysisRulesFactory.Context

Changed API:

- TokenPredicate.Categories moved out to own class as Categories
- TokenPredicate.Type moved out to own class as EvaluatorType
- TokenPredicate.TokenPredicateImpl moved out to own class as TokenPredicateImpl
- TokenPredicate.ExactTokenPredicateImpl moved out to own class as ExactTokenPredicateImpl

- TokenPredicate.EvaluationException moved out to own class as EvaluationRuntimeException (remaining an unchecked exception)
- TokenPredicate.Static moved out to own class as TokenPredicateUtility
- Categories.EXACT_WIKI renamed to EXACT_WIKIPEDIA (to match non-exact peer)
- Categories.EXACT_FIRST renamed to EXACT_FIRSTNAME (to match non-exact peer)
- VeryFastEvaluationException renamed to EvaluationException (remaining a checked exception)

Removed API:

- TokenPredicate.getType() [TokenPredicate no longer references TokenEvaluator in any manner]
- TokenPredicate.Static.getTokenPredicates(type) [AbstractEvaluatorFactory responsible now for relationship]
- TokenPredicate.Static.getMagicTokenPredicates() [magic tokens a sesam.no concept, not sesat]
- TokenPredicate.Static.getTriggerTokenPredicates() [trigger tokens a sesam.no concept, not sesat]

New Features

Selecting content type in layout element in views.xml (2nd July)

SKER4919 - Configurable content types
In views.xml//views/tab/layout you can now define the attribute content-type.
Example: <layout main="main" front="index" content-type="text/xml; charset=utf-8"/>
For further flexibility use a jsp page with setContentType(..) in scriptlet code.
Added API:

- SearchTab.Layout.contentType & SearchTab.Layout.getContentType()

Sesat-ise and standardise decorators (2nd July)

SKER4182 - Sesat-ise and standardise decorators
The output parameter was used to select decorators like the rss, og vcard. This is a bit more configurable now. The output parameter has been renamed to layout and refers to the id attribute in the layout element in views.xml.
Example: <layout id="opensearch" main="opensearchDecorator.jsp"/> This will let you use layout=opensearch, and you get forwarded to the opensearchDecorator. If you put this in a views.xml early in the chain, then all pages will have this ability.
For backward compatibility output=xyz rewrites to layout=xyz (see urlrewrite.xml)

Solr Search Command (15th September)

SKER4949: (Solr SearchCommand implementation)
Searching can now be configured against a Solr index. Example configuration available commented out in generic.sesam/sesam.com/war/src/main/src/main/conf/modes.xml
Added API:

- SolrCommandConfig
- SolrSearchCommand

Solr Token Evaluation (15th September)

SKER4952: (Solr TokenEvaluator)
Query evaluation (or Query matching) can now be configured against a Solr index.
Full tutorial on setup and configing exists at Setting up Sesat's Query Evaluation with a Solr index
Added API:

- SolrEvaluatorFactory
- SolrTokenEvaluator

Youtube Search Command (26th September)

SKER4753: (Promote YoutubeSearchCommand (from Schibsted Søk AS))
Searching can now be configured against Youtube.
Added API:

- YoutubeCommandConfig
- YoutubeSearchCommand

Field Splitter result handler (27th September)

SKER4752: (Promote FieldSplitter result handler (from Schibsted Søk AS))
Searching can now be configured against Youtube.
Added API:

- FieldSplitterResultHandlerConfig
- FieldSplitter

Clustering hit count result handler (27th September)

SKER4781: (Promote ClusteringHitCountResultHandler (from Schibsted Søk AS))
Searching can now be configured against Youtube.
Added API:

- ClusteringHitCountResultHandlerConfig
- ClusteringHitCountResultHandler

An Interpreter to help debug and inspect Sesat.

See the page below for info.
Sesat-Interpreter

Generating XML Schema files for the modes.xml configuration files

We are now generating XML Schema files for the modes.xml files. And validating against these files during build.
See the page below for info.
Search modes schema generator

Requirements

SiteMesh is no longer a requirement (2nd July)

SiteMesh was removed by work done in SKER4182 - Sesat-ise and standardise decorators.

Writing Upgrade Guide guidelines

This page last changed on May 26, 2008 by mick.

1. Prefix page with {toc:indent=5}
2. The page should contain three h1. headings: Compatibility changes, New Features, and Requirements
3. Each "change" should be a h6. heading and include a date stamp
4. Each "change" should refer to a scarab issue
5. Each "compatibility change" has it's body text quoted
6. Each "compatibility change" should list separately: Changed API, Added APIs, Removed APIs, Deprecated APIs, and equivalents for SPIs.

About

This page last changed on Jan 25, 2008 by mick.

Free Software License

This page last changed on Jan 25, 2008 by mick.

Sesat is licensed under the Affero GPLv3 license as follows

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively

on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under

those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not

accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in

connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

This page last changed on May 06, 2008 by mick.

The Sesat.no Project

Sesat.no is hosted over three machines.
The first is for the website, subversion, mailing lists.
The second machine is dedicated to scarab + mysql.
The third machine is dedicated to hudson and opengrok.

Web Site

Webpages are exported from confluence using the autoexport plugin.

Version Control

Subversion located at http://sesat.no/svn/
Browsable history through Opengrok at http://sesat.no/source/

Mailing Lists

Running mailman at http://sesat.no/mailman/listinfo
with postfix and postgreyfilter.

Issue Tracker

Running scarab at http://sesat.no/scarab/

This page last changed on Jan 25, 2008 by mick.

SESAT Propriety License

SESAT is free and licensed under Affero GPL V3. We may consider providing SESAT under a proprietary license in order that derivative works could be kept privately, if you absolutely cannot use the free license.

Sitemap

This page last changed on Jul 09, 2008 by mick.

The following is a sitemap of the wiki pages in this site. It excludes content from hudson/, mailman/, maven2/, projects/, source/, and svn/.

- Home
  - Apache
  - download
- Community
  - Getting Help
  - Guide Helping
  - Kernel Roadmap
  - Team List
- External
  - External - FAST
  - External - Integration Platforms
- Features
  - Kernel Feature list
  - Product Description
    - Product Description - Business Managers & Business Developers
  - Product Whitepaper
  - Showcases - Sesam.no
  - Technical Feature Matrix
- Docs + Support
  - Architecture Overview
    - Design Proposals
      - New design proposal for SearchCommand and AbstractSearchCommand
        - QueryBuilder code example
        - SearchCommandParameter code example
        - Stream manipulation code examples
    - Personalization architecture
  - Debugging
    - Debugging Velocity Templates
    - Logging, Logfiles, and Statistics
    - Sesat-Interpreter
  - Development Guidelines
    - Building with FAST
    - Kernel Operations
    - Requirements
  - Development platform architecture
    - Hardware Requirements
    - Software Requirements
  - Development using the APIs
  - Development using the SFC
    - Developing a quick and simple Enrichment tutorial
    - Developing using CSS
    - HOWTO develop a RunHandler
    - Navigation documentation
    - Search modes schema generator
    - Tutorial - Ajax examples
    - Tutorial - Building Sesam.com
  - faq
  - HowTo Solr Query Evaluation
  - Upgrade Guides
    - 2.16 Upgrade
    - 2.17 Upgrade
    - 2.18 Upgrade
    - Writing Upgrade Guide guidelines
- About
  - Free Software License
  - Project
  - Propriety License
  - Sitemap

News items

Title	Author	Date Posted
Lucene and Solr patches accepted	Mck Semb Wever	Jan 31, 2009
Sesat 2.18 - "The acquisition of wealth is no longer the driving force in our lives."	Mck Semb Wever	Dec 03, 2008
Sesat 2.18 - "The acquisition of wealth is no longer the driving force in our lives."	Mck Semb Wever	Nov 20, 2008
federatedsearchblog.com - "New open source federated search middleware released"	Mck Semb Wever	Jul 09, 2008
Sesat on Sourceforge	Mck Semb Wever	Apr 28, 2008
Sesat on Freshmeat	Mck Semb Wever	Apr 28, 2008
Sesat 2.17 - "Are we human because we gaze at the stars or do we gaze at the stars because we are human?"	Mck Semb Wever	Apr 22, 2008
Sesat 2.17 - "Are we human because we gaze at the stars or do we gaze at the stars because we are human?"	Mck Semb Wever	Apr 08, 2008
jaxmag.com - Search Middleware Portal Generally Available To Access Data Source	Mck Semb Wever	Apr 03, 2008
Sesat, a federated search solution middleware, goes open source	Mck Semb Wever	Mar 31, 2008
Cominvent - Norweigan search portal Sesam.no releases middleware as GPL	Mck Semb Wever	Mar 30, 2008
DIGG - Build your own search engine with SESAT	Mck Semb Wever	Mar 30, 2008
Sesambloggen - Sesam Search Application Toolkit (SESAT) available as Free Software	Mck Semb Wever	Mar 21, 2008
Digi.no - Sesam deler ut kildekoden	Mck Semb Wever	Mar 21, 2008