SortingSearchResults

Sorting Search Results

Current Lucene implementation in DSpace does not specify any sort criteria for searches. I suppose "relevancy" is then used by default (right?)
This page wants to discuss how one could parameterize other search criteria.

Sorting, why?

Google is demonstrating the advantage of bringing the most useful pages first. "Usefulness" is different when looking in different types of repositories. In PoisonCentre.be case, most recent articles are preferred by our users. In other context, relevancy, title, first author, affiliation, and even compound keys (e. g. year of publication + title) may be preferred.

Implementing one basic sort, how?

A prototype patch has been submitted and may be examined on SourceForge to understand how this proposal could be implemented: Sorting simple /advanced Search results

Please do not hesitate to contact me for help implementing this patch: Christophe.Dupriez

This development is part of a bigger one I must do for managing links between records from DSpace (and non DSpace) repositories, a little bit like links in a Wiki. Preliminary specifications has been written.

Different kinds of Sort

- Relevancy: no Sort key must be specified to Lucene
- IssueDate: an UNTOKENIZED field must be added to Lucene, containing the issue timestamp (or just its first 10 characters for date only indexing). Sorting on Integer is more efficient: the date could be rewritten as YYYYMMDD (8 digits)
- Title: an UNTOKENIZED field must be added to Lucence. Depending of the language, articles may have to be removed from the beginning of the title. Case and punctuation have to be normalized. All this before storing in a Lucene field because this is an UNTOKENIZED field.
- First author: An UNTOKENIZED field is (by definition) single valued. Lucene is therefore unable to generate a sort result where multiple entries exist for a single record (e.g. one for each author). Punctuation normalization is important for author names.

Hopefully multiple sort criteria can be specified at search time: compound key will not be necessary. It will be possible to sort by the following combination of sort keys at search time: first author - reverse publication year - title. Publication year sort is in reversed order to insure more recent publications are listed first.

User Interface

- Localized sort names will have to be defined in the different Message.properties files.
- A menu of available sort orders could prefix all search results to allow users to choose the one they prefer.
- Browsing: Sort indexes could be used for a very efficient implementation of some of the current browsing types: title, date, first author. This could use Lucene class TermEnum. Other indexes would have to be created for fields with multiple values (TOKENIZED on a character delimiter which separate field occurrences)

Specifying a Sort Order at search time

This will be a supplementary parameter to the search servlet. The search form may include a drop down menu for this parameter.

It could be a string made of sort keys (e.g. title, author, issuedate) prefixed by a minus sign if the key must be in reverse order. The sort parameter in the HTML form (JSP) could be written:

by Publication date

by Author

by Title

by Relevance

Two special sort names would be predefined:

- DOC: Lucene internal document number
- SCORE: Score of the document toward the search expression

Specifying a Sort Key (basic proposal)

DSpace often proposes predefined processing for authors, subjects and titles. This basic proposal would be in line with this approach by predefining essential sort indexes types:

- date: extract an 8 digits integer YYYYMMDD from a field containing a date as a string
- name: put field content in upper case and normalize punctuation
- title: put field content in upper case, normalize punctuation and removes leading articles
- default: put field content in upper case

Proposal modeled on search index definitions (*search.index.1 = author:dc.contributor.**):

"sort.index.1 = dateissued: date(dc.date.issued)

"sort.index.2 = firstauthor: name(dc.contributor.author)

"sort.index.3 = sorttitle: title(dc.title)

This basic implementation can be a step before going to something more complex.

Specifying a Sort Key (more complex proposal)

*I propose not to follow this proposal of mine for now*Christophe.Dupriez 19:26, 2 January 2008 (EST)

The sort key (somewhat like a Lucene tokenization process) must be specified by defining a sometime complex "string" expression. From my point of view, EL (JSTL Expression Language) has the level of expressiveness required especially if an expansion mechanism allows to add new Java String functions. It is not ideal (no concatenation operator: a function is required; it is interpreted, not compiled) but it "fits the job" (better proposals???).

A different way: we can use OGNL. It say "...Most of what you can do in Java is possible in OGNL, plus other extras..." --Bollini 07:20, 2 August 2007 (EDT)

Core EL functions are documented under Sun JCP.

Proposal modeled on search index definitions (*search.index.1 = author:dc.contributor.**):

sort.index.1 = dateissued: fn:substring(dc.date.issued.1.value, 0, 10) #Keeps only 10 first characters of the date

sort.index.2 = firstauthor: fn:toUpperCase(fn:trim(dc.contributor.author.1.value)),'--',9999-substring(dc.date.issued.1.value, 0, 4) #Sort by author and then reverse publication year

sort.index.3 = sorttitle: ds:noLeadingArticle(fn:toUpperCase(fn:trim(dc.title.1.value)),dc.title.1.language) #Titles are sorted without leading article

Core EL functions are not all we need. Hopefully extensions mechanisms exist and can be used (how can this be done in line with extensions mechanisms currently designed?). In above example, we suppose a JSTL library "ds" has been defined and contains a public static Java function String noLeadingArticle(String title, String language).

Preprocessing may have to be done to convert above definitions to real EL:

- Commas at first level could indicated concatenation of multiple strings: it would be interpreted as a sequence of EL expressions to be concatenated
- dc... : It may be cumbersome to load a TreeMap of TreeMaps to represent the metadata of an Item (For instance: TreeMap dc containing TreeMap date, TreeMap contributor and TreeMap title; TreeMap date containing TreeMap issued; TreeMap issued containing TreeMap 1 containing String value, String language, etc.). Imagine the work for DSIndexer having to rebuild the whole thing! It may be much more interesting to define shorthand notations like *dc.date.issued* which could be automatically translated to ds:get('dc.date.issued')

N.B. Sort indexes and Search indexes would not have the same names: a Sort index should be usable for string searches (not word searches) like those needed for PersistentIdentifiers

Introducing EL in DSpace, could it open other doors?

EL Expressions could:

- streamline many aspects of JSP (JSP UI without Java, only JSTL tags and EL !) and could help the transition toward Manakin (comments???)
- specify any custom information to display using DSpace data without modifying JSP
- specify individual fields display (I will make a presentation of the LinkOut concept I developped for the PoisonCentre)
- help specify pre-processing of metadata before indexing

Complete configuration example

(not using EL (Expression Language))

We would have 3 kinds of Lucene indexes:

1. index: value words are indexed (tokenization)
2. identifier: each value is indexed as one token (as with KeywordTokenizer)
3. sortindex: only one value is indexed and as is (not tokenized)

Different new configurable options:

1. A cleanup function (java class) could be called upon each value to normalize dates, remove leading articles, etc. before sending the value to be indexed to Lucene.
2. A suffix to DSpace field name (-en is proposed) could permit to specify that an index is built starting from a specific linguistic version of a value.
3. The Analyzer (and then the Filters and the Tokenizer) could vary from one field to another.
4. A configuration option would provide the list of the indexes combined for a "simple" search.

Another would provide those proposed in the Advanced Search "index selection box".

```
# Each index / identifier index / sort index below can be used
# in a search equation.
# The name of each should be defined in Message.properties
search.analyzer = org.dspace.search.DSAnalyzer
search.index.author=dc.contributor.*,dc.creator.*, dc.description.statementofresponsibility
search.index.title=dc.title.*
search.index.keyword=dc.subject.*
search.index.abstract=dc.description.abstract, dc.description.tableofcontents
search.index.series=dc.relation.ispartofseries
search.index.sponsor=dc.description.sponsorship
# We change "index" by "identifier" to indicate
# that each "dc.identifier" fields values are one identifier
#(KeywordAnalyzer: no separation of words)
search.identifier.mime=dc.format.mimetype
search.identifier.identifier=dc.identifier.*
search.identifier.language=dc.language.iso
# One may wish to not search "fulltext" in the simple search
search.combined=author,title,keyword,abstract,sponsor,identifier,series
# Only untokenized fields (single valued) can be used for sorting:
search.sortindex.dateissued=dc.date.issued
# Untokenized fields may have to be cleaned up before indexing.
# Here, we remove time (and possibly days) to reduce
# the potential problem with "Range" searches:
search.cleanup.dateissued=be.destin.dspace.dateCleanUp
# Only the first existing occurrence will be used
# Note "-en" used to indicate the desired language occurrence
search.sortindex.sorttitle=dc.title-en,dc.title-fr,dc.title
search.cleanup.sorttitle=be.destin.dspace.titleCleanUp
search.sortindex.lc=dc.subject.lc
search.sortindex.mention=dc.description.statementofresponsibility
search.identifier.issn=dc.identifier.issn
search.sortindex.author1=dc.contributor
# Index differently Arabic texts:
search.index.arabtitle=dc.title-ar
search.analyzer.arabtitle=gpl.pierrick.brihaye.aramorph.lucene.ArabicStemAnalyzer
# List of advanced search indexes to proposes in Web UI (no JSP modification anymore)
search.advanced=fulltext,title,arabtitle,author,keyword,abstract,sponsor,series,mime,language,identifier
# Some prefers not to have full text search when making a "simple" search:
search.combined=title,arabtitle,author,keyword,abstract,sponsor,series, identifier
# Sort type may combine more than one sort field:
# (Note the minus sign for decreasing order applied on one of the sort keys)
search.sort.date=-dateissued,sorttitle
search.sort.title=sorttitle,mention,-dateissued
search.sort.journal=issn,dateissued,sorttitle
search.sort.lc=lc,sorttitle
search.sort.author1=author1,-dateissued
# Order of the sort options proposed to the user (Advanced search, result browsing)
search.sort=date,journal,author1,title,lc,SCORE
```

Message.fr.properties (example for any other languages)

```
# The following would NOT be used anymore:
jsp.search.advanced.type.abstract   = R\u00E9sum\u00E9s
jsp.search.advanced.type.author     = Auteurs
jsp.search.advanced.type.id         = Identificateur
jsp.search.advanced.type.keyword    = Tous les index
jsp.search.advanced.type.language   = Langue (ISO)
jsp.search.advanced.type.series     = Collections
jsp.search.advanced.type.sponsor    = Organismes subventionnaires
jsp.search.advanced.type.subject    = Sujets
jsp.search.advanced.type.title      = Titres

# The following are general names for defined indexes,
# used for all places needing them for user display
#(including Advanced Search JSP):
search.index.fulltext=Texte intégral
search.index.combined=Partout
search.index.author=Auteurs
search.index.title=Titres
search.index.subject=Sujets
search.index.abstract=Résumés
search.index.series=Collections
search.index.sponsor=Sponsor
search.index.identifier=Identifiant
search.index.language=Langue
search.index.dateissued=Publication
search.index.sorttitle=Titre
search.index.lc=Cote LC
search.index.mention=Mention de responsabilité
search.index.issn=ISSN
search.index.arabtitle=Titre en Arabe
# Display names for possible sorts
search.sort.date=par date
search.sort.title=par titre
search.sort.author1=par 1er auteur
search.sort.lc=par cote LC
search.sort.issn=par périodique
search.sort.SCORE=par intérèt
search.sort.DOC=par clé d'accès
```

Christophe.Dupriez 19:21, 2 January 2008 (EST)