LinksManagement

Links Management

Authority lists, thesauri, serial records, other repositories with same authors, external information resources are all asking for validated links entry, recording, display and removal.

We have defined a "link markup language" (will be explained in a future publication) which enables users to write links within metadata fields. The goal is to allow the user to follow the links, from current record to other ones, but also to allow to search on the reverse ("what is linking to here?").

For instance, Arachnidismtherapylinks the current record with the subject "Arachnidism" by the relation "therapy"; Phenyl SalicylateCAS=118-55-8 links the current record with the substance with CAS#118-55-8.

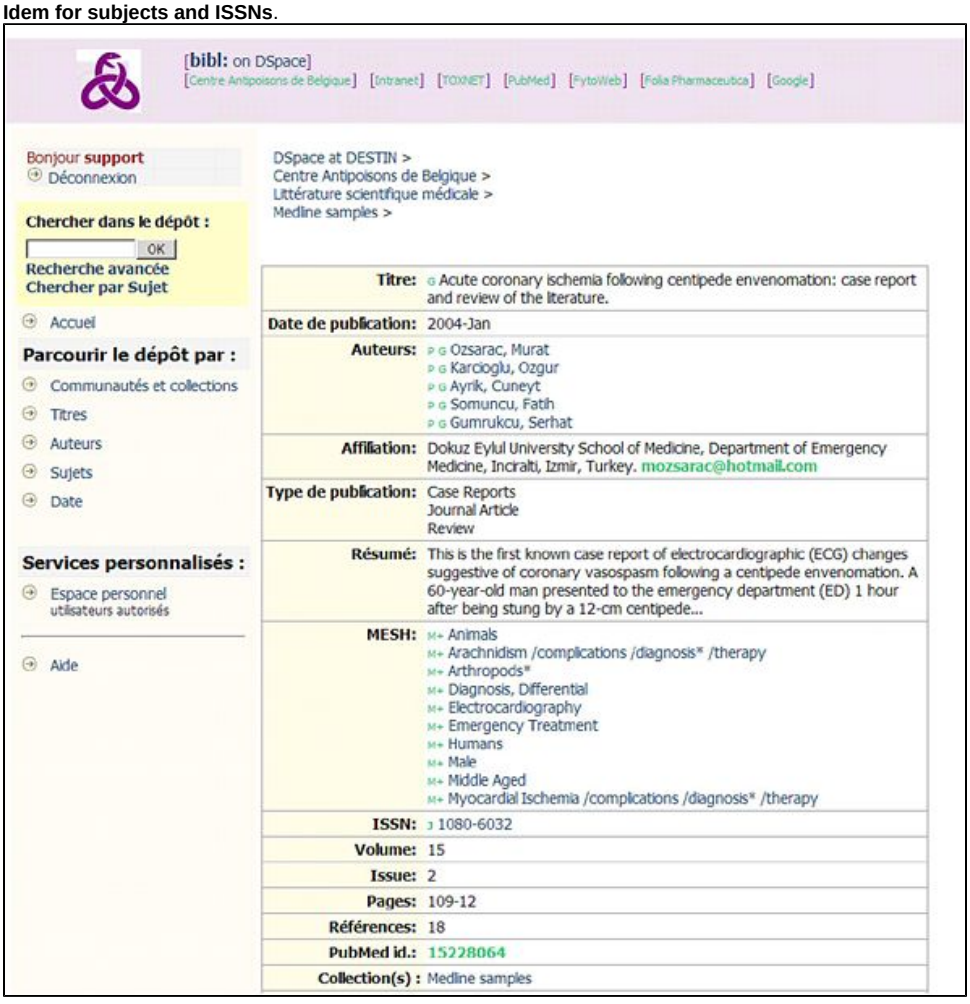
Each time a record is updated, this markup has to be interpreted to insure that if "A is related to B", "B is reverse-related to A" and therefore that DSpace can be queried to know what is linked to B.

In the above example, this means to be able to navigate from current record to a record giving more information about Phenyl Salicylate but also to be able to get the list of all documents linking to Phenyl Salicylate.

Link Management will be an integration "roof" for all applications (DSpace and others) and it will use the "object" keys extensively to represent relations. We will double the current DSpace item indexer (DSIndexer*) with a "link indexer" to parse the metadata fields and to store links within Lucene* or a database: any application will then be able to query the links between objects, being managed by DSpace or not.

A "LinkOut* like" module has already been added to DSpace to display metadata fields together with HTML links giving access to external databases or to "horizontal search" within the DSpace application. In the example below:

- The green links are going toward external applications
- G triggers a Google search
  P triggers a search using PubMed/Entrez.
  M links to PubMed/Entrez and requests a search using MeSH Thesaurus.
- + do the same thing but restricted to MeSH Major Topics*
  The blue links trigger "horizontal searches": clicking an author names searches every document in the repository for the same author. Idem for subjects and ISSNs.



Indexing :

Within a given application O, Item A.field F may contain one or more "wiki like" links:

```
... S[Rw=K] ...
```

This field content has to be parsed to extract one or more assertions stating that:
Resource O, Item A has a field F containing a string S has
a relation R (always linking to, an external resource X) with access key K, weight w

Functional needs:

1. Configuration data to define resources, fields, relations, assumptions for each different field, etc. (details below): How do we state configuration?
2. Parsing of all fields of item A (within DSIndexer?) to identify deleted and added assertions. This parsing will have to fill in "assumed" information (see below).
3. Added/Deleted assertions must be communicated to the "add/delete assertion" method(s) applied on object identified by string K of external resource X in the context of citing relation R (weight w)
4. Added assertions should be locally indexed in DSpace (Lucene) with a string which may be retrieved either by "S R", by "R K" or S alone or K alone.
5. Deleted assertions should be locally de-indexed.

Configuration

Apart from the new indexing parameters explained in SortingSearchResults, we would also have the following relations (indexes) definitions in dspace.cfg:

```
# Relations definition
relation.default.cas=dc.subject.substances
relation.default.compositeur=dc.contributor
relation.direct.issn=dc.identifier.issn
relation.mandatory.cas=dc.subject.substance
```

Relations targets

Those could be in a "central" configuration file accessible to all applications. Its format would be compatible with INI files to be directly readable also by Microsoft applications.

A relation links a source and a target resource. A resource as an "application name": this name must be translated by each application to a full path to the database (in Java, this can be done using JNDI).

```
[triplestore]
server=application-name-for-assertions-database-server
type=SQL    (default for now)
current=SELECT a AS RESOURCE, b AS KEY, r AS RELATION, x AS TARGETRESOURCE, y AS TARGETKEY, w AS WEIGHT, l AS
LABEL FROM t WHERE a=? AND b=? ...
check=SELECT a AS RESOURCE, b AS KEY, r AS RELATION, x AS TARGETRESOURCE, y AS TARGETKEY, w AS WEIGHT, l AS
LABEL FROM t WHERE a=? AND b=? AND r=? AND x=? AND y=? ...
delete=DELETE FROM t WHERE a=? AND b=? AND r=? AND x=? AND y=? ...
add=INSERT INTO t ( a, b, r, x, y, w, l ) VALUES ( ... )
search=SELECT a AS RESOURCE, b AS KEY, r AS RELATION, x AS TARGETRESOURCE, y AS TARGETKEY, w AS WEIGHT, l AS
LABEL FROM t WHERE a=? AND b=? AND r=? ...
reversesearch=SELECT a AS RESOURCE, b AS KEY, r AS RELATION, x AS TARGETRESOURCE, y AS TARGETKEY, w AS WEIGHT,
l AS LABEL FROM t WHERE r=? AND x=? AND y=? ...
[resources]
bibl=application-name-for-bibl-application-database-server
prod=application-name-for-prod-application-database-server
[relations]
resource.issn=bibl
search.issn=SELECT x AS CODE, y AS TITLE, k AS KEY, z AS COMMENT ... WHERE ... y LIKE ?% ...
get.issn=SELECT x AS CODE, y AS TITLE, k AS KEY, z AS COMMENT ... WHERE ... x=? ...
getkey.issn=SELECT x AS CODE, y AS TITLE, k AS KEY, z AS COMMENT ... WHERE ... k=? ...
display.issn=http://infor3:8080/dspace/search??query=issn:?
```

Christophe.Dupriez 14:50, 3 January 2008 (EST)