Discovery

What is DSpace Discovery

The Discovery Module for the XML user interface enables faceted searching & browsing for your repository.

Although these techniques are new in DSpace, they might feel familiar from other platforms like Aquabroser or Amazon, where facets help you to select the right product according to facets like price and brand. DSpace Discovery offers very powerful browse and search configurations that were only possible with code customization in the past.

Watch the DSpace Discovery introduction video

What is a Sidebar Facet

From the user perspective, faceted search (also called faceted navigation, guided navigation, or parametric search) breaks up search results into multiple categories, typically showing counts for each, and allows the user to "drill down" or further restrict their search results based on those facets.

When you have successfully enabled Discovery in your DSpace, you will notice that the different enabled facets are visualized in a "Discover" section in your sidebar, by default, right below the Browse options.

In this example, there are 3 Sidebar Facets, Author, Subject and Date Issued. It's important to know that multiple metadata fields can be included in one facet. For example, the Author facet above includes values from both dc.contributor.author as well as dc.creator.

Another important property of Sidebar Facets is that their contents are automatically updated to the context of the page. On collection homepages or community homepages it will include information about the items included in that particular collection or community.

What is a Search Filter

In a standard search operation, a user specifies his complete query prior to launching the operation. If the results are not satisfactory, the user starts over again with a (slightly) altered query.

In a faceted search, a user can modify the list of displayed search results by specifying additional "filters" that will be applied on the list of search results. In DSpace, a filter is a contain condition applied to specific facets. In the example below, a user started with the search term "approach", which yielded 15 results. By applying the filter "economics" on the facet "Subject". After applying this filter, only 6 results remain.

Another example would be the standard search operation [**wetland + "dc.author=Mitsch, William J" + dc.subject="water quality"**]. With filtered search, a user can start by searching for [**wetland**], and then filter the results by the other attributes, author and subject.

Discovery Features

- Configurable sidebar browse facets that can display contents from any metadata field
    - Dynamically generated timespans for dates
- Customizable recent submissions display on the repository homepage, collection and community pages
- Auto-complete on search terms

DSpace 1.8 Improvements

- Configuration moved from dspace.cfg into config/modules/discovery.cfg and config/spring/discovery/spring-dspace-addon-discovery-configuration-services.xml
- Individual communities and collections can have their own Discovery configuration.
- Tokenization for Auto-complete values (see SearchFilter)
- Alphanumeric sorting for Sidebarfacets
- Possibility to avoid indexation of specific metadata fields.
- Grouping of multiple metadata fields under the same SidebarFacet

Enabling Discovery

As with any upgrade procedure, it is highly recommend that you backup your existing data thoroughly. Although upgrades in versions of Solr/Lucene do tend to be forwards compatible for the data stored in the Lucene index, it is always a best practice to backup your [dspace]/solr/statistics cores to assure no data is lost.

1. Enable the Discovery Aspects in the XMLUI by changing the following settings in config/xmlui.xconf
    a. Comment out: SearchArtifacts
    b. Uncomment: Discovery

```
<xmlui>
    <aspects>
        <!--
            @deprecated: the Artifact Browser has been devided into ViewArtifacts,
            BrowseArtifacts, SearchArtifacts
            <aspect name="Artifact Browser" path="resource://aspects/ArtifactBrowser/" />
        -->
        <aspect name="Displaying Artifacts" path="resource://aspects/ViewArtifacts/" />
        <aspect name="Browsing Artifacts" path="resource://aspects/BrowseArtifacts/" />
        <!--<aspect name="Searching Artifacts" path="resource://aspects/SearchArtifacts/" />-->
        <aspect name="Administration" path="resource://aspects/Administrative/" />
        <aspect name="E-Person" path="resource://aspects/EPerson/" />
        <aspect name="Submission and Workflow" path="resource://aspects/Submission/" />
        <aspect name="Statistics" path="resource://aspects/Statistics/" />

        <!--
            To enable Discovery, uncomment this Aspect that will enable it
            within your existing XMLUI
            Also make sure to comment the SearchArtifacts aspect
            as leaving it on together with discovery will cause UI overlap issues-->
        <aspect name="Discovery" path="resource://aspects/Discovery/" />


        <!--
            This aspect tests the various possible DRI features,
            it helps a theme developer create themes
        -->
        <!-- <aspect name="XML Tests" path="resource://aspects/XMLTest/"/> -->
    </aspects>
```

2. Enable the Discovery Indexing Consumer that will update Discovery Indexes on changes to content in XMLUI, JSPUI, SWORD, and LNI in config /dspace.cfg
    a. Add discovery to the list of event.dispatcher.default.consumers

```
# default synchronous dispatcher (same behavior as traditional DSpace)
event.dispatcher.default.class = org.dspace.event.BasicDispatcher
#event.dispatcher.default.consumers = search, browse, eperson, harvester
event.dispatcher.default.consumers = search, browse, discovery, eperson, harvester
```

    b. Change recent.submissions.count to zero

```
#Put the recent submissions count to 0 so that discovery can use it's recent submissions,
# not doing this when discovery is enabled will cause UI overlap issues
#How many recent submissions should be displayed at any one time
#recent.submissions.count = 5
recent.submissions.count = 0
```

3. Check that the port is correct for solr.search.server in config/modules/discovery.cfg
    a. If all of your traffic runs over port 80, then you need to remove the port from the URL

```
##### Search Indexing #####
solr.search.server = http://localhost/solr/search
```

4. From the command line, navigate to the dspace directory and run the command below to index the content of your DSpace instance into Discovery.

```
./bin/dspace update-discovery-index
```

NOTE: This step may take some time if you have a large number of items in your repository.

5. Verify if you now see the Sidebar Facets on your DSpace homepage. Note that these are **only visible when you have items in your repository**.

Configuration files

The configuration for discovery is located in 2 separate files.

- General settings: The **discovery.cfg** file located in the [dspace]/config/modules directory.
- User Interface Configuration: The **spring-dspace-addon-discovery-configuration-services.xml** file is located in [dspace]/config/spring /discovery/ directory.

General Discovery settings (config/modules/discovery.cfg)

The discovery.cfg file is located in the [dspace]/config/modules directory and contains following properties:

Property:	**search.server**
Example Value:	search.server=http://localhost:8080/solr/search
Informational Note:	Discovery relies on a SOLR index for storage and retrieval of its information. This parameter determines the location of the SOLR index.
Property:	**index.ignore**
Example Value:	index.ignore=dc.description.provenance,dc.language
Informational Note:	By default, Discovery will include all of the DSpace metadata in its search index. In cases where specific metadata is confidential, repository managers can include those fields by adding them to this comma separated list.

Modifying the Discovery User Interface (config/spring/spring-dspace-addon-discovery-configuration-services.xml)

The spring-dspace-addon-discovery-configuration-services.xml file is located in the [dspace]/config/spring directory.

Structure Summary

Because this file is in XML format, you should be familiar with XML before editing this file. The configurations are organized together in beans, depending on the purpose these properties are used for.
This purpose can be derived from the class of the beans. Here's a short summaries of classes you will encounter throughout the file and what the corresponding properties in the bean are used for.

Download the configuration file and review it together with the following parameters

Class:	**DiscoveryConfigurationService**
Purpose:	Defines the mapping between separate Discovery configurations and individual collections /communities
Default:	All communities, collections and the homepage (key=default) are mapped to defaultConfiguration
Class:	**DiscoveryConfiguration**
Purpose:	Groups configurations for sidebar facets, search filters, search sort options and recent submissions
Default:	There is one configuration by default called defaultConfiguration
Class:	**DiscoverySearchFilter**
Purpose:	Defines that specific metadata fields should be enabled as a search filter
Default:	dc.title, dc.contributor.author, dc.creator, dc.subject.* and dc.date.issued are defined as search filters
Class:	**DiscoverySidebarFacetConfiguration**
Purpose:	Defines which metadata fields should be offered as a contextual sidebar browse option
Default:	dc.contributor.author, dc.creator, dc.subject.* and dc.date.issued
Class:	**DiscoverySortConfiguration**
Purpose:	Further specifies the sort options to which a DiscoveryConfiguration refers
Default:	dc.title and dc.date.issued are defined as alternatives for sorting, other than Relevance (hard coded)

Default settings

In addition to the summarized descriptions of the default values, following details help you to better understand these defaults. If you haven't yet, download the configuration file and review it together with the following parameters.
The file contains one default configuration that defines following sidebar facets, search filters, sort fields and recent submissions display:

- Sidebar facets
  - **sidebarFacetAuthor:** groups the metadata fields dc.contributor.author & dc.creator with a facet limit of 10, sorted by occurrence count
  - **sidebarFacetSubject:** groups all subject metadata fields (dc.subject.*) with a facet limit of 10, sorted by occurrence count
  - **sidebarFacetDateIssued:** contains the dc.date.issued metadata field, which is identified with the type "date" and sorted by specific date values
- Search filters
  - **searchFilterTitle:** contains the dc.title metadata field and has a tokenized autocomplete
  - **searchFilterAuthor:** contains the dc.contributor.author & dc.creator metadata fields and has a non tokenized autocomplete configured
  - **searchFilterSubject:** contains the dc.subject.* metadata fields and has a non tokenized autocomplete configured
  - **searchFilterIssued:** contains the dc.date.issued metadata field with the type "date" and has a tokenized autocomplete

- Sort fields
  - **sortTitle**: contains the dc.title metadata field
  - **sortDateIssued:** contains the dc.date.issued metadata field, this sort has the type date configured.

- defaultFilterQueries
  - The default configuration contains no defaultFilterQueries
  - The default filter queries are disabled by default but there is an example in the default configuration in comments which allows discovery to only return items (as opposed to also communities/collections).

- Recent Submissions
  - The recent submissions are sorted by dc.date. accessioned which is a date and a maximum number of 5 recent submissions are displayed.

Many of the properties contain lists which use references to point to the configuration elements. This way a certain configuration type can be used in multiple discovery configurations so there is no need to duplicate these.

SidebarFacet Customization

This section explains the properties of an individual SidebarFacet, like SidebarFacetAuthor, SidebarFacetSubject and SidebarFacetDateIssued from the default configuration. In order to create custom SidebarFacets, you can either modify specific properties of those that already exist or create a totally new one from scratch.

Here's what the SidebarFacetAuthor looks like:

```
<bean id="sidebarFacetAuthor" class="org.dspace.discovery.configuration.SidebarFacetConfiguration">
        <property name="indexFieldName" value="author"/>
        <property name="metadataFields">
            <list>
                <value>dc.contributor.author</value>
                <value>dc.creator</value>
            </list>
        </property>
        <property name="facetLimit" value="10"/>
        <property name="sortOrder" value="COUNT"/>
        <property name="type" value="text"/>
    </bean>
```

The id & class attributes are mandatory for this type of bean. The properties that it contains are discussed below.

- **indexFieldName** (Required)**:** A unique sidebarfacet field name, the metadata will be indexed in SOLR under this field name.
- **metadataFields** (Required): A list of the metadata fields that need to be included in the facet.
- **facetLimit** (optional)**:** The maximum number of values to be shown. This property is optional, if none is specified 10 will be used. When a type of **d ate** is given, this property will not be used since dates are automatically grouped together.
- **sortOrder** (optional)**:** The sort order for the sidebar facets, it can either be COUNT or VALUE. If none is given the COUNT value is used as a default.
  - **COUNT** Facets will be sorted by the amount of times they appear in the repository
  - **VALUE** Facets will be sorted alphanumeric
- **type** (optional): the type of the sidebar facet it can either be date or text, if none is defined text will be used.
  - **text:** The facets will be treated as is
  - **date**: Only the year will be identified from the values and stored in the SOLR index. These years are automatically grouped together and offered as a drill-down browse.

SearchFilter Customization

This section explains the properties of an individual SearchFilter, like searchFilterTitle, searchFilterAuthor, searchFilterSubject and searchFilterIssued from the default configuration. In order to create custom Search Filters, you can either modify specific properties of those that already exist or create a totally new one from scratch.

Here's what the searchFilterAuthor looks like:

```
<bean id="searchFilterAuthor" class="org.dspace.discovery.configuration.DiscoverySearchFilter">
    <property name="indexFieldName" value="author"/>
    <property name="metadataFields">
        <list>
            <value>dc.contributor.author</value>
            <value>dc.creator</value>
        </list>
    </property>
    <property name="fullAutoComplete" value="true"/>
    <property name="type" value="text"/>
</bean>
```

The id & class attributes are mandatory for this type of bean. The properties that it contains are discussed below.

- **indexFieldName** (Required)**:** A unique search filter field name, the metadata will be indexed under this field name
- **metadataFields** (Required): A list containing the metadata fields which can be used in this filter
- **fullAutoComplete** (optional): If set to true the values indexed for autocomplete will not be tokenized, if set to false tokenization will occur. Tokenization is the process of breaking up text strings in individual words. In this case, with tokenization activated, a title like "Medical Guidelines" will respond both to the "M" and to the "G", because both words are indexed individually for auto-completion.
- **type** (optional): the type of the search filter it can either be date or text, if none is defined text will be used.
    - **text:** The metadata will be treated as is
    - **date**: With a type of date the dates will receive the following format: yyyy-MM-dd (2011-07-01)

Sort option customization for search results

This section explains the properties of an individual SortConfiguration, like sortTitle and sortDateIssued from the default configuration. In order to create custom sort options, you can either modify specific properties of those that already exist or create a totally new one from scratch.

Here's what the sortTitle SortConfiguration looks like:

```
<bean id="sortTitle" class="org.dspace.discovery.configuration.DiscoverySortFieldConfiguration">
        <property name="metadataField" value="dc.title"/>
        <property name="type" value="text"/>
 </bean>
```

The id & class attributes are mandatory for this type of bean. The properties that it contains are discussed below.

- **metadataField** (Required): The metadata field indicating the sort values
- **type** (optional): the type of the sort option can either be date or text, if none is defined text will be used.

DiscoveryConfiguration

The DiscoveryConfiguration Groups configurations for sidebar facets, search filters, search sort options and recent submissions. If you want to show the same sidebar facets, use the same search filters, search options and recent submissions everywhere in your repository, you will only need one DiscoveryConfiguration and you might as well just edit the defaultConfiguration.

The DiscoveryConfiguration makes it very easy to use custom sidebar facets, search filters, ... on specific communities or collection homepage. This is particularly useful if your collections are heterogeneous. For example, in a collection with conference papers, you might want to offer a sidebar facet for conference date, which might be more relevant than the actual issued date of the proceedings. In a collection with papers, you might want to offer a facet for funding bodies or publisher, while these fields are irrelevant for items like learning objects.

A DiscoveryConfiguration consists out of five parts

- The list of applicable sidebarFacets
- The list of applicable searchFilters
- The list of applicable searchSortFields
- Any default filter queries (optional)
- The configuration for the Recent submissions display

Configuring lists of sidebarFacets and searchFilters

Below is an example of how one of these lists can be configured. It's important that each of the bean references corresponds with the exact name of the earlier defined Facets, filters or sort options.

```
<property name="sidebarFacets">
    <list>
        <ref bean="sidebarFacetAuthor" />
        <ref bean="sidebarFacetSubject" />
        <ref bean="sidebarFacetDateIssued" />
    </list>
</property>
```

Configuring and customizing search sort fields

The search sort field configuration block contains the available sort fields and the possibility to configure a default sort field and sort order.
Below is an example of the sort configuration.

```
<property name="searchSortConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoverySortConfiguration">
        <!--<property name="defaultSort" ref="sortDateIssued"/>-->
        <!--DefaultSortOrder can either be desc or asc (desc is default)-->
        <property name="defaultSortOrder" value="desc"/>
        <property name="sortFields">
            <list>
                <ref bean="sortTitle" />
                <ref bean="sortDateIssued" />
            </list>
        </property>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are discusses below.

- **defaultSort** (optional): The default field on which the search results will be sorted, this must be a reference to an existing search sort field bean. If none is given relevance will be the default. Sorting according to the internal relevance algorithm is always available, even though it's not explicitly mentioned in the sortFields section.
- **defaultSortOrder** (optional): The default sort order can either be asc or desc.
- **sortFields** (mandatory): The list of available sort options, each element in this list must link to an existing sort field configuration bean.

Adding default filter queries (OPTIONAL)

Default filter queries are applied on all search operations & sidebarfacet clicks. One useful application of default filter queries is ensuring that all returned results are items. As a result, subcommunities and collections that are returned as results of the search operation, are filtered out.
Similar to the lists above, the default filter queries are defined as a list. They are optional.

```
<property name="defaultFilterQueries">
    <list>
        <value>query1</value>
        <value>query2</value>
    </list>
</property>
```

This property contains a simple list which in turn contains the queries. Some examples of possible queries:

- search.resourcetype:2
- dc.subject:test
- dc.contributor.author: "Van de Velde, Kevin"
- ...

Customizing the Recent Submissions display

The recent submissions configuration element contains all the configuration settings to display the list of recently submitted items on the home page or community/collection page. Because the recent submission configuration is in the discovery configuration block, it is possible to show 10 recently submitted items on the home page but 5 on the community/collection pages.

Below is an example configuration of the recent submissions.

```
<property name="recentSubmissionConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryRecentSubmissionsConfiguration">
        <property name="metadataSortField" value="dc.date.accessioned"/>
        <property name="type" value="date"/>
        <property name="max" value="5"/>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are discusses below.

- **metadataSortField** (mandatory): The metadata field to sort on to retrieve the recent submissions
- **max** (mandatory): The maximum number of results to be displayed as recent submissions
- **type** (optional): the type of the search filter it can either be date or text, if none is defined text will be used.

Discovery SOLR Index Maintenance

Command used:	`[dspace]/bin/dspace update-discovery-index [-cbhf[r <item handle>]]`
Java class:	org.dspace.discovery.IndexClient
Arguments (short and long forms):	Description
	called without any options, will update/clean an existing index
`-b`	(re)build index, wiping out current one if it exists
`-c`	clean existing index removing any documents that no longer exist in the db
`-f`	if updating existing index, force each handle to be reindexed even if uptodate
`-h`	print this help message
`-o`	optimize search core
`-r <item handle>`	remove an Item, Collection or Community from index based on its handle

Routine Discovery SOLR Index Maintenance

It is strongly recommended to run maintenance on the Discovery SOLR index daily (from crontab or your system's scheduler), to prevent your servlet container from running out of memory:

```
[dspace]/bin/dspace update-discovery-index -o
```

Advanced SOLR Configuration

Discovery is built as an application layer on top of the Open Source Enterprise Search Server SOLR. Therefore, SOLR configuration can be applied to the SOLR cores that are shipped with DSpace.
The DSpace SOLR instance itself now runs two cores. One for collection DSpace Solr based "statistics", the other for Discovery Solr based "search".

```
solr
 search
    conf
       admin-extra.html
       elevate.xml
       protwords.txt
       schema.xml
       scripts.conf
       solrconfig.xml
       spellings.txt
       stopwords.txt
       synonyms.txt
       xslt
           DRI.xsl
           example.xsl
           example_atom.xsl
           example_rss.xsl
           luke.xsl
    conf2
 solr.xml
 statistics
     conf
         admin-extra.html
         elevate.xml
         protwords.txt
         schema.xml
         scripts.conf
         solrconfig.xml
         spellings.txt
         stopwords.txt
         synonyms.txt
         xslt
             example.xsl
             example_atom.xsl
             example_rss.xsl
             luke.xsl
```