Discovery

1 What is DSpace Discovery
    1.1 What is a Sidebar Facet
    1.2 What is a Search Filter
    1.3 What is a tag cloud facet
2 Discovery Changelist
    2.1 DSpace 6.0
    2.2 DSpace 5.0
    2.3 DSpace 4.0
    2.4 DSpace 3.0
    2.5 DSpace 1.8
    2.6 DSpace 1.7
3 Configuration files
4 General Discovery settings (config/modules/discovery.cfg)
5 Modifying the Discovery User Interface (config/spring/api/discovery.xml)
    5.1 Structure Summary
    5.2 Default settings
    5.3 Non indexed metadata fields
    5.4 Search filters & sidebar facets Customization
        5.4.1 Hierarchical (taxonomies based) sidebar facets
    5.5 Sort option customization for search results
    5.6 DiscoveryConfiguration
        5.6.1 Configuring lists of sidebarFacets and searchFilters
        5.6.2 Configuring and customizing search sort fields
        5.6.3 Adding default filter queries (OPTIONAL)
        5.6.4 Access Rights Awareness
            5.6.4.1 Access Rights Awareness - technical details
        5.6.5 Customizing the Recent Submissions display
        5.6.6 Customizing hit highlighting & search snippets
            5.6.6.1 Hit highlighting technical details
        5.6.7 "More like this" configuration
            5.6.7.1 "More like this" technical details
        5.6.8 "Did you mean" spellcheck aid for search configuration
            5.6.8.1 "Did you mean" spellcheck aid for search technical details
        5.6.9 Customizing the "Tag Cloud" facet
        5.6.10 Disabling the "Has file(s)" facet
6 Discovery Solr Index Maintenance
7 Advanced Solr Configuration
8 Internationalization

What is DSpace Discovery

The Discovery Module enables faceted searching & browsing for your repository.

Although these techniques are new in DSpace, they might feel familiar from other platforms like Aquabrowser 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

Since 6.0, Discovery is the only out-of-the-box Search and Browse infrastructure provided in DSpace.

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.



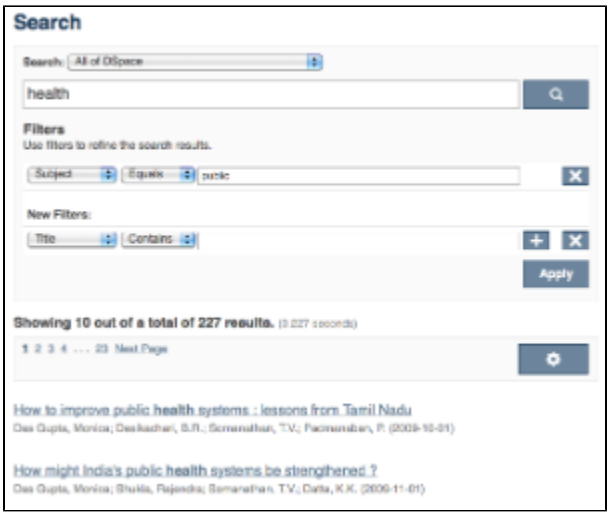
In this example, there are 3 Sidebar Facets, Author, Subject and Date Issued. It's important to know that multiple metadata fields can be included in one facet. For example, the Author facet above includes values from both dc.contributor.author as well as dc.creator.

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.

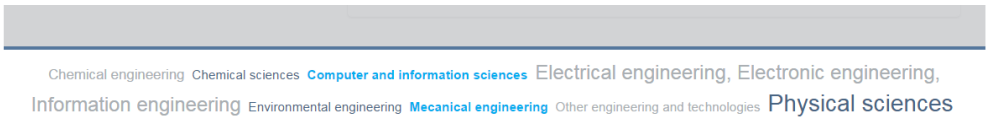
In a faceted search, a user can modify the list of displayed search results by specifying additional "filters" that will be applied on the list of search results. In DSpace, a filter is a contain condition applied to specific facets. In the example below, a user started with the search term "health", which yielded 500 results. After applying the filter "public" on the facet "Subject", only 227 results remain. Each time a user selects a sidebar facet it will be added as a filter. Active filters can be altered or removed in the 'filters' section of the search interface.



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

What is a tag cloud facet

Tag cloud facet is another way to display facets of your repository in a "tag cloud" form in which the importance of each tag is show with font size or color. This format is useful for quickly perceiving the most prominent terms.



This is a classic "tag cloud" facet in a DSpace repository.

Discovery Changelist

DSpace 6.0

The legacy search engine (based on Apache Lucene) and legacy Browse system (based on database tables) have been removed from DSpace 6.0 or above. Instead, DSpace now only uses Discovery for all Search/Browse capabilities.

In addition, to support the new Configuration options, all of the Discovery configurations in `discovery.cfg` have been prefixed with "`discovery.`" (see configuration below).

DSpace 5.0

The new JSPUI-only tag cloud facet feature is disabled by default. In order to enable it, you will need to set up the corresponding processor that the PluginManager will load to actually perform the tag cloud query on the relevant pages. This is configured in the dspace.cfg configuration file using the following properties:

- plugin.sequence.org .dspace.plugin.CommunityHomeProcessor
- plugin.sequence.org .dspace.plugin.CollectionHomeProcessor
- plugin.sequence.org .dspace.plugin.SiteHomeProcessor

The tag cloud has been declared there for you but it is commented out.

DSpace 4.0

Starting from DSpace 4.0, Discovery is the default search and browse solution for DSpace.

General improvements:

- Browse interfaces now also use Discovery index (rather than the legacy, now retired, Lucene index)
- "Did you means" spell check aid for search

DSpace 3.0

Starting from DSpace 3.0, Discovery is also supported in JSPUI.

General improvements:

- Hierarchical facets sidebar facets
- Improved & more intuitive user interface
- Access Rights Awareness (enabled by default). Access restricted or embargoed content is hidden from anonymous search/browse.
- Authority control & variants awareness (homonyms are shown separately in a facet if they have different authority ID). All variant forms as recognized by the authority framework are indexed. See Authority Framework

XMLUI-only:

- Hit highlighting and search snippets support
- "More like this" (related items)

Bugfixes and other changes

- Auto-complete functionality has been removed in XMLUI from search queries due to performance issues. JSPUI still supports auto-complete functionality without performance issues.

DSpace 1.8

- Configuration moved from dspace.cfg into `config/modules/discovery.cfg` and `config/spring/api/discovery.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

DSpace 1.7

- Sidebar browse facets that can be configured to use contents from any metadata field
    - Dynamically generated timespans for dates

- Customizable "recent submissions" view on the repository homepage, collection and community pages
- Hit highlighting & search snippets

Configuration files

The configuration for discovery is located in 2 separate files.

- General settings: The `discovery.cfg` file located in the `[dspace-install-dir]/config/modules directory`.
- User Interface Configuration: The `discovery.xml` file is located in `[dspace-install-dir]/config/spring/api/` directory.

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

The `discovery.cfg` file is located in the `[dspace]/config/modules` directory and contains following properties. Any of these properties may be overridden in your `local.cfg` (see Configuration Reference):

Property:	**discovery.search.server**
Example Value:	`discovery.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. If you are uncertain whether this property is set correctly, you can use a commandline tool like "wget" to perform a query against the Solr index (and ensure Solr responds). For example, the below query searches the Solr index for "test" and returns the response on standard out: `wget -O - http://localhost:8080/solr/search/select?q=test`
Property:	**discovery.index.authority.ignore[.field]**

Example Value:	`discovery.index.authority.ignore=true`
	
	`discovery.index.authority.ignore.dc.contributor.author=false`

Informational Note:	By default, Discovery will use the authority information in the metadata to disambiguate homonyms. Setting this property to false will make the indexing process the same as the metadata doesn't include authority information. The configuration can be different on a field (<schema>.<element>.<qualifier>) basis, the property without field set the default value.
Property:	**discovery.index.authority.ignore-prefered[.field]**

Example Value:	`discovery.index.authority.ignore-prefered=true`
	
	`discovery.index.authority.ignore-prefered.dc.contributor.author=false`

Informational Note:	By default, Discovery will use the authority information in the metadata to query the authority for the preferred label. Setting this property to false will make the indexing process the same as the metadata doesn't include authority information (i.e. the preferred form is the one recorded in the metadata value). The configuration can be different on a field (<schema>.<element>.<qualifier>) basis, the property without field set the default value. **If the authority is a remote service, disabling this feature can greatly improve performance.**

Property:	**discovery.index.authority.ignore-variants[.field]**

Example Value:	`discovery.index.authority.ignore-variants=true`
	
	`discovery.index.authority.ignore-variants.dc.contributor.author=false`

Informational Note:	By default, Discovery will use the authority information in the metadata to query the authority for variants. Setting this property to false will make the indexing process the same, as the metadata doesn't include authority information. The configuration can be different on a per-field (<schema>.<element>.<qualifier>) basis, the property without field set the default value. **If authority is a remote service, disabling this feature can greatly improve performance.**

Modifying the Discovery User Interface (`config/spring/api/discovery.xml`)

The `discovery.xml` file is located in the `[dspace]/config/spring/api` directory.

Structure Summary

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 summary 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, also controls the metadata fields that should not be indexed in the search core (item provenance for example).
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:	**DiscoverySearchFilterFacet**
Purpose:	Defines which metadata fields should be offered as a contextual sidebar browse options, each of these facets has also got to be a search filter
Default:	dc.contributor.author, dc.creator, dc.subject.* and dc.date.issued
Class:	**HierarchicalSidebarFacetConfiguration**
Purpose:	Defines which metadata fields contain hierarchical data and should be offered as a contextual sidebar option

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)
Class:	**DiscoveryHitHighlightingConfiguration**
Purpose:	Defines which metadata fields can contain hit highlighting & search snippets
Default:	dc.title, dc.contributor.author, dc.subject, dc.description.abstract & full text from text files.
Class:	**TagCloudFacetConfiguration**
Purpose:	Defines the tag cloud appearance configuration bean and the search filter facets to appear in the tag cloud form. You can have different "**TagCloudFacetConfiguration**" per community or collection or the home page

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 already done so, 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
  - **searchFilterAuthor:** groups the metadata fields dc.contributor.author & dc.creator with a facet limit of 10, sorted by occurrence count
  - **searchFilterSubject:** groups all subject metadata fields (dc.subject.*) with a facet limit of 10, sorted by occurrence count
  - **searchFilterIssued:** 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
  - **searchFilterAuthor:** contains the dc.contributor.author & dc.creator metadata fields
  - **searchFilterSubject:** contains the dc.subject.* metadata fields
  - **searchFilterIssued:** contains the dc.date.issued metadata field with the type "date"
- 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.
- Hit highlighting
  - The fields dc.title, dc.contributor.author & dc.subject can contain hit highlighting.
  - The dc.description.abstract & full text field are used to render search snippets.
- Non indexed metadata fields
  - **Community/Collections:** dc.rights (copyright text)
  - **Items:** dc.description.provenance

Many of the properties contain lists that 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 them.

Non indexed metadata fields

The discovery.xml file has configuration to not index certain metadata fields for communities/collections/items. The configuration is handled in the "toIgnoreMetadataFields" property located in the "org.dspace.discovery.configuration.DiscoveryConfigurationService" bean. Below is an example configuration that excludes dc.description.provenance for items & dc.rights for communities/collections:

```
<property name="toIgnoreMetadataFields">
    <map>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.COMMUNITY"/></key>
            <list>
                <!--Introduction text-->
                <!--<value>dc.description</value>-->
                <!--Short description-->
                <!--<value>dc.description.abstract</value>-->
                <!--News-->
                <!--<value>dc.description.tableofcontents</value>-->
                <!--Copyright text-->
                <value>dc.rights</value>
                <!--Community name-->
                <!--<value>dc.title</value>-->
            </list>
        </entry>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.COLLECTION"/></key>
            <list>
                <!--Introduction text-->
                <!--<value>dc.description</value>-->
                <!--Short description-->
                <!--<value>dc.description.abstract</value>-->
                <!--News-->
                <!--<value>dc.description.tableofcontents</value>-->
                <!--Copyright text-->
                <value>dc.rights</value>
                <!--Collection name-->
                <!--<value>dc.title</value>-->
            </list>
        </entry>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.ITEM"/></key>
            <list>
                <value>dc.description.provenance</value>
            </list>
        </entry>
    </map>
</property>
```

By adding additional values to the appropriate lists additional metadata can be excluded from the search core, a reindex is required after altering this file to ensure that the values are removed from the index.

Search filters & sidebar facets Customization

This section explains the properties for search filters & sidebar facets. Each sidebar facet must occur in the reference list of the search filters. Below is an example configuration of a search filter that is not used as a sidebar facet.

```
<bean id="searchFilterTitle" class="org.dspace.discovery.configuration.DiscoverySearchFilter">
    <property name="indexFieldName" value="title"/>
    <property name="metadataFields">
        <list>
            <value>dc.title</value>
        </list>
    </property>
</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 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.

Sidebar facets extend the search filter and add some extra properties to it, below is an example of a search filter that is also used as a sidebar facet.

```
<bean id="searchFilterAuthor" 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>
```

Note that the class has changed from **DiscoverySearchFilter** to **SidebarFacetConfiguration** this is needed to support the extra properties.

- **facetLimit** (optional)**:** The maximum number of values to be shown. This property is optional, if none is specified the default value "10" will be used. If the filter has the type **date**, 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. The default value is COUNT.
    - **COUNT** Facets will be sorted by the amount of times they appear in the repository
    - **VALUE** Facets will be sorted alphabetically
- **type**(optional): the type of the sidebar facet it can either be "date" or "text", "text" is the default value.
    - **text:** The facets will be treated as is
    - **date**: Only the year will be stored in the Solr index. These years are automatically displayed in ranges that get smaller when you select one.

Hierarchical (taxonomies based) sidebar facets

Discovery supports specialized drill down in hierarchically structured metadata fields. For this drill down to work, the metadata in the field for which you enable this must be composed out of terms, divided by a splitter. For example, you could have a dc.subject.taxonomy field in which you keep metadata like "CARTOGRAPHY::PHOTOGRAMMETRY", in which Cartography and Photogrammetry are both terms, divided by the splitter "::". The sidebar will only display the top level facets, when clicking on view more all the facet options will be displayed.

```
<bean id="searchFilterSubject" class="org.dspace.discovery.configuration.HierarchicalSidebarFacetConfiguration">
    <property name="indexFieldName" value="subject"/>
    <property name="metadataFields">
        <list>
            <value>dc.subject</value>
        </list>
    </property>
    <property name="sortOrder" value="COUNT"/>
    <property name="splitter" value="::"/>
    <property name="skipFirstNodeLevel" value="false"/>
</bean>
```

Note that the class has changed from **SidebarFacetConfiguration** to **HierarchicalSidebarFacetConfiguration** this is needed to support the extra properties.

- **splitter** (required)**:** The splitter used to split up the separate nodes
- **skipFirstNodeLevel** (optional)**:** Whether or not to show the root node level. For some hierarchical data there is a single root node. In most cases it doesn't need to be shown since it isn't relevant. **This property is true by default.**

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
- The configuration of the tag cloud facet

Configuring lists of sidebarFacets and searchFilters

After modifying sidebarFacets and searchFilters, don't forget to reindex existing items by running `[dspace]/bin/dspace index-discovery -b`, otherwise the changes will not appear.

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

Each sidebar facet must also occur in the list of the search filters.

```
<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"
- ...

Access Rights Awareness

By default, when searching and browsing using Discovery, you will only see items that you have access to. So, your search/browse results may differ if you are logged into DSpace. This Access Rights Awareness feature ensures that anonymous users (and search engines) are not able to access information (both files and metadata) about embargoed or private items. It also provides you with more direct control over who can see individual items within your DSpace.

How does Access Rights Awareness work?

Access Rights Awareness checks the "READ" access on the Item.

If the "Anonymous" group has "READ" access on the Item, then anonymous/public users will be able to view that Item's metadata and locate that Item via DSpace's search/browse system. In addition, search engines will also be able to index that Item's metadata. However, even with Anonymous READ set at the Item-level, you may still choose to access-restrict the downloading/viewing of *files* within the Item. To do so, you would restrict "READ" access on individual Bitstream(s) attached to the Item.

If the "Anonymous" group does NOT have "READ" access on the Item, then anonymous users will never see that Item appear within their search/browse results (essentially the Item is "invisible" to them). In addition, that Item will be invisible to search engines, so it will never be indexed by them. However, any users who have been given READ access will be able to find/locate the item after logging into DSpace. For example, if a "Staff" group was provided "READ" access on the Item, then members of that "Staff" group would be able to locate the item via search/browse after logging into DSpace.

How can I disable Access Rights Awareness?

If you prefer to allow all access-restricted or embargoed Items to be findable within your DSpace, you can choose to turn off Access Rights Awareness. However, please be aware that this means that restricting "READ" access on an Item will not really do anything – the Item metadata will be available to the public no matter what group(s) were given READ access on that Item.

This feature can be switched off by going to the `[dspace.dir]/config/spring/api/discovery.xml` file & commenting out the bean & the alias shown below.

```
<bean class="org.dspace.discovery.SolrServiceResourceRestrictionPlugin" id="solrServiceResourceIndexPlugin"/>

<alias name="solrServiceResourceIndexPlugin" alias="org.dspace.discovery.SolrServiceResourceRestrictionPlugin"/>
```

The Browse Engine only supports the "Access Rights Awareness" if the Solr/Discovery backend is enabled (see Defining the Storage of the Browse Data). However, it is enabled by default for DSpace 3.x and above.

Access Rights Awareness - technical details

The *DSpaceObject* class has an *updateLastModified()* method which will be triggered each time an authorization policy changes. This method is only implemented in the item class where the last_modified timestamp will be updated and a modify event will be fired. By doing this we ensure that the discovery consumer is called and the item is reindexed. Since this feature can be switched off a separate plugin has been created: the *SolrServiceResourceRestrictionPlugin.* Whenever we reindex a DSpace object all the read rights will be stored in the read field. We make a distinction between groups and users by adding a *'g'* prefix for groups and the *'e'* prefix for epersons.

When searching in discovery all the groups the user belongs to will be added as a filter query as well as the users identifier. If the user is an admin all items will be returned since an admin has read rights on everything.

Customizing the Recent Submissions display

This paragraph only applies to XMLUI. JSPUI relies on the Browse Engine to show "recent submissions". This requires that the Solr/Discovery backend is enabled (see Defining the Storage of the Browse Data).

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.

Customizing hit highlighting & search snippets

The hit highlighting configuration element contains all settings necessary to display search snippets & enable hit highlighting.

This section only applies to XMLUI. JSPUI does not currently support "highlighting & search snippets".
Disabling hit highlighting / search snippets

You can disable hit highlighting / search snippets by commenting out the entire `<property name="hitHighlightingConfiguration">` Configuration in the `[dspace]/config/spring/api/discovery.xml` configuration file.

PLEASE BE AWARE there are two sections where this <property> definition exists. You should comment out both. One is under the `<bean id="defaultConfiguration">` and one is under the `<bean id="homepageConfiguration">`

Alternatively, you may also choose to tweak which fields are shown in hit highlighting, or modify the number of matching words shown (snippets) and/or number of characters shown around the matching word (maxSize).

For this change to take effect in the User Interface, you will need to restart Tomcat.
Changes made to the configuration will not automatically be displayed in the user interface. By default, only the following fields are displayed: dc.title, dc.contributor.author, dc.creator, dc.contributor, dc.date.issued, dc.publisher, dc.description.abstract and fulltext.

If additional fields are required, look for the "itemSummaryList" template.

Below is an example configuration of hit highlighting.

```
<property name="hitHighlightingConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightingConfiguration">
        <property name="metadataFields">
            <list>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.title"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.contributor.author"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.subject"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.description.abstract"/>
                    <!-- Max number of characters to display around the matching word (Warning setting to 0
returns entire field) -->
                    <property name="maxSize" value="250"/>
                    <!-- Max number of snippets (matching words) to show -->
                    <property name="snippets" value="2"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                                    <!-- Displays snippets from indexed full text of document (for
supported formats) -->
                    <property name="field" value="fulltext"/>
                    <!-- Max number of characters to display around the matching word (Warning setting to 0
returns entire field) -->
                    <property name="maxSize" value="250"/>
                                    <!-- Max number of snippets (matching words) to show -->
                    <property name="snippets" value="2"/>
                </bean>
            </list>
        </property>
    </bean>
</property>
```

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

- **field** (mandatory)**:** The metadata field to be highlighted (can also be * if all the metadata fields should be highlighted).
- **maxSize** (optional): Limit the number of characters displayed to only the relevant part (use metadata field as search snippet).
- **snippets** (optional)**:** The maximum number of snippets that can be found in one metadata field.

Hit highlighting technical details

The *org.dspace.discovery.DiscoveryQuery* object has a setter & getter for the hit highlighting configuration set in Discovery configuration. If this configuration is given the *resolveToSolrQuery* method located in the *org.dspace.discovery.SolrServiceImpl* class will use the standard Solr highlighting feature (http://wiki.apache.org/solr/HighlightingParameters). The *org.dspace.discovery.DiscoverResult* class has a method to set the highlighted fields for each object & field.

The rendering of search results is no longer handled by the METS format but uses a special type of list named "TYPE_DSO_LIST". Each metadata field (& fulltext if configured) is added in the DRI and IF the field contains hit higlighting the Java code will split up the string & add *DRI highlights* to the list. The XSL for the themes also contains special rendering XSL for the DRI; for Mirage, the changes are located in the *discovery.xsl* file. For themes using the old themes based on structural.xsl, look for the template matching "*dri:list[@type='dsolist']*".

"More like this" configuration

This paragraph only apply to XMLUI. The JSPUI does not currently support the "More like this" feature.

The "more like this"-configuration element contains all the settings for displaying related items on an item display page.
Below is an example of the "more like this" configuration.

```
<property name="moreLikeThisConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryMoreLikeThisConfiguration">
        <property name="similarityMetadataFields">
            <list>
                <value>dc.title</value>
                <value>dc.contributor.author</value>
                <value>dc.creator</value>
                <value>dc.subject</value>
            </list>
        </property>
        <!--The minimum number of matching terms across the metadata fields above before an item is found as
related -->
        <property name="minTermFrequency" value="5"/>
        <!--The maximum number of related items displayed-->
        <property name="max" value="3"/>
        <!--The minimum word length below which words will be ignored-->
        <property name="minWordLength" value="5"/>
    </bean>
</property>
```

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

- similarityMetadataFields: the metadata fields checked for similarity
- minTermFrequency: The minimum number of matching terms accross the metadata fields above before an item is found as related
- max: The maximum number of related items displayed
- minWordLength: The minimum word length below which words will be ignored

"More like this" technical details

The *org.dspace.discovery.SearchService* object has received a *getRelatedItems()* method. This method requires an item & the more-like-this configuration bean from above. This method is implemented in the *org.dspace.discovery.SolrServiceImpl* which uses the item as a query & uses the default Solr parameters for more-like-this to pass the bean configuration to solr (https://cwiki.apache.org/confluence/display/solr/MoreLikeThis). The result will be a list of items or if none found an empty list. The rendering of this list is handled in the *org.dspace.app.xmlui.aspect.discovery.RelatedItems* class.

"Did you mean" spellcheck aid for search configuration

DSpace 4 introduces the use of SOLR's SpellCheckComponent as an aid for search. When a user's search does not return any hits, the user is presented with a suggestion for an alternative search query.



The feature currently only one line of configuration to discovery.xml. Changing the value from true to false will disable the feature.

```
<property name="spellCheckEnabled" value="true" />
```

"Did you mean" spellcheck aid for search technical details

Similar to the More like this configuration, SOLR's spell check component is used with default configuration values. Any of these values can be overridden in the solrconfig.xml file located in dspace/solr/search/conf/. Following links provide more information about the SOLR SpellCheckComponent:

http://wiki.apache.org/solr/SpellCheckComponent

https://cwiki.apache.org/confluence/display/solr/Spell+Checking

Customizing the "Tag Cloud" facet

This paragraph only applies to JSPUI

```
<!-- Set TagCloud configuration per discovery configuration -->
<property name="tagCloudFacetConfiguration" ref="defaultTagCloudFacetConfiguration"/>
```

*Declare the bean (of class: **TagCloudFacetConfiguration**) that holds the configuration for the tag cloud facet.*

```
<!--TagCloud configuration bean for homepage discovery configuration-->
<bean id="homepageTagCloudFacetConfiguration" class="org.dspace.discovery.configuration.
TagCloudFacetConfiguration">
        <!-- Actual configuration of the tagcloud (colors, sorting, etc.) -->
        <property name="tagCloudConfiguration" ref="tagCloudConfiguration"/>
        <!-- List of tagclouds to appear, one for every search filter, one after the other -->
        <property name="tagCloudFacets">
            <list>
                <ref bean="searchFilterSubject" />
            </list>
        </property>
</bean>
```

This bean has two properties:

- **tagCloudConfiguration**: is the bean which describes the actual appearance parameters
- **tagCloudFacets**: the search filter facets which will be used for the tag cloud. If you leave the list empty, no tag cloud will appear. If you declare more than one, such number of tag clouds will appear for each search filter, one after the other.

The appearance configuration can have the following properties, as shown in the following bean:

```
<bean id="tagCloudConfiguration" class="org.dspace.discovery.configuration.TagCloudConfiguration">
              <!-- Should display the score of each tag next to it? Default: false -->
              <property name="displayScore" value="true"/>
              <!-- Should display the tag as center aligned in the page or left aligned? Possible values: true
| false. Default: true  -->
              <property name="shouldCenter" value="true"/>
              <!-- How many tags will be shown. Value -1 means all of them. Default: -1 -->
              <property name="totalTags" value="-1"/>
              <!-- The letter case of the tags.
                      Possible values: Case.LOWER | Case.UPPER | Case.CAPITALIZATION | Case.PRESERVE_CASE |
Case.CASE_SENSITIVE
                      Default: Case.PRESERVE_CASE -->
              <property name="cloudCase" value="Case.PRESERVE_CASE"/>
              <!-- If the 3 CSS classes of the tag cloud should be independent of score (random=yes) or based
on the score. Possible values: true | false . Default: true-->
              <property name="randomColors" value="true"/>
              <!-- The font size (in em) for the tag with the lowest score. Possible values: any decimal.
Default: 1.1 -->
              <property name="fontFrom" value="1.1"/>
              <!-- The font size (in em) for the tag with the lowest score. Possible values: any decimal.
Default: 3.2 -->
              <property name="fontTo" value="3.2"/>
              <!-- The score that tags with lower than that will not appear in the rag cloud. Possible values:
any integer from 1 to infinity. Default: 0 -->
              <property name="cuttingLevel" value="0"/>
              <!-- The distance (in px) between the tags. Default: 5 -->
              <property name="marginRight" value="5"/>
              <!-- The ordering of the tags (based either on the name or the score of the tag)
                      Possible values: Tag.NameComparatorAsc | Tag.NameComparatorDesc | Tag.ScoreComparatorAsc
| Tag.ScoreComparatorDesc
                      Default: Tag.NameComparatorAsc  -->
              <property name="ordering" value="Tag.NameComparatorAsc"/>
    </bean>
```

When tagCloud is rendered there are some CSS classes that you can change in order to change the appearance of the tag cloud.

Class	Note
tagcloud	General class for the whole tagcloud
tagcloud_1	Specific tag class for tag of type 1 (based on score)
tagcloud_2	Specific tag class for tag of type 2 (based on score)
tagcloud_3	Specific tag class for tag of type 3 (based on score)

Disabling the "Has file(s)" facet

Since DSpace 6, a new "Has file(s)" facet has been enabled by default. This facet shows whether items have or do not have any bitstreams in the "ORIGINAL" bundle.
Should you want to turn this off, you can edit `[dspace]/config/spring/api/discovery.xml` to remove the following line from the `defaultConfiguration` and `homepageConfiguration` beans (in the `sidebarFacets` property):

```
<ref bean="searchFilterContentInOriginalBundle"/>
```

Then restart your servlet container.

Discovery Solr Index Maintenance

Command used:	`[dspace]/bin/dspace index-discovery [-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
`-i <object handle>`	Reindex an individual object (and any child objects). When run on an Item, it just reindexes that single Item. When run on a Collection, it reindexes the Collection itself and all Items in that Collection. When run on a Community, it reindexes the Community itself and all sub-Communities, contained Collections and contained Items.
`-o`	optimize search core
`-r <item handle>`	remove an Item, Collection or Community from index based on its handle
-s	Rebuild the spellchecker, can be combined with -b and -f.

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

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

(Since Solr 4, the underlying optimize operation has been discouraged as mostly unnecessary and renamed. See https://issues.apache.org/jira/browse/SOLR-3141).

Advanced Solr Configuration

Discovery is built as an application layer on top of the Solr open source enterprise search server. Therefore, Solr configuration can be applied to the Solr cores that are shipped with DSpace.
The DSpace Solr instance currently runs several cores (which means indexes in Solr parlance). The "statistics" core is for collection of DSpace usage events for statistical purposes (if you have been collecting statistics for multiple years, you may have chosen to use sharding and you will see one core per each year collected). The "search" core is used by Discovery for for search and faceting, for displaying the collection/community hierarchy and item counts. The "authority" core is used by SolrAuthority to store information about authors, including their data imported from the ORCID registry.

```
solr
 solr.xml
 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
 ...
 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
```

Internationalization

Discovery has its own messages.xml file, located at dspace-xmlui/src/main/resources/aspects/Discovery/i18n/messages.xml. To add your own labels for new fields and facets in a Maven overlay, copy this file to dspace/modules/xmlui/src/main/resources/aspects/Discovery/i18n/messages.xml and modify this file. Alternatively, you may add them to the main messages.xml file. Same goes for translations - it's encouraged to submit a single messages_XX.xml file including messages from all the separate messages.xml files in DSpace.

Advanced search related keys (change "author" to desired field)

Filter name	xmlui.ArtifactBrowser.SimpleSearch.filter.author
Facet heading	xmlui.ArtifactBrowser.AdvancedSearch.type_author
"Filter by" page heading	xmlui.Discovery.AbstractSearch.type_author