Solr

Solr in DSpace

What is Solr: http://lucene.apache.org/solr/features.html

DSpace uses Solr as a part of Discovery as index to speed up access to content metadata and data about access to DSpace (for statistics). It also provides faceting, search results filtering and in newer versions of DSpace also hit highlighting and "More like this". If Discovery is enabled, the DSpace search field accepts Solr search syntax.
Discovery is an optional part of DSpace since 1.7 (with big improvements and configuration format changes in 1.8). When enabled, Discovery replaces DSpace Search and Browse and provides Solr-based statistics. Since DSpace 3, it is also the default storage for the DSpace OAI-PMH provider (server) responses.

Do I need to read this page?

To gain the benefits of faceting and filtering in XMLUI, all you need to do is enable Discovery. The rest of these page describes some advanced uses of Solr - if you want to query Solr directly for theme customization or read DSpace metadata from outside DSpace.

Please, note, that to get data from Solr, you **don't** technically need to enable the Discovery aspect, but you **do need** to populate the index. The *statistics* core is populated automatically in DSpace 1.6+. To populate the *search* core (DSpace 1.7+), you need to run `[dspace]/bin/dspace index-discovery` (you will probably want to schedule it in cron to run periodically, too). In DSpace versions older than 4.x, the command was called `[dspace]/bin/dspace update-discovery-index`. There should be no reason to access the *oai* core (DSpace 3.0), because it contains the same information as the search core, but if you want to populate it, run `[dspace]/bin/dspace oai import`.

Connecting to Solr

By default, the DSpace Solr server is configured to listen only on localhost, port 8080 (unless you specified another port in Tomcat configuration and the `[dspace]/config/modules/discovery.cfg` config file). That means that you cannot connect from another machine to the dspace server port 8080 and request a Solr URL - you'll get a HTTP 403 error. This configuration was done for security considerations - Solr index contains some data that is not accessible via public DSpace interfaces and some of the data might be sensitive.

Before you try to follow the advice below to bypass the localhost restriction, please note:

- Exposing the Solr interface means, that any restricted metadata such as dc.description.provenance and non-anonymized usage statistics (client IPs, user agent strings) will be accessible.
- Exposing the Solr interface also means that it will be exposed for **write access**. There is no easy way to expose only read access.
- Never expose Solr to the internet. If you're exposing it to an IP within your network, add it as an exception to the LocalHostRestrictionFilter. If you have to expose Solr to a public IP, use a SSH tunnel or a VPN for the connection.

Bypassing localhost restriction temporarily

While you could make Solr publicly accessible by changing this default configuration, this is not recommended, because Solr indexes may contain some data you might consider private. Instead, use one of following simple means to bypass this restriction temporarily. All of them will make Solr accessible only to the machine you're connecting from for as long as the connection is open.

1. **OpenSSH client - port forwarding**
   connect to DSpace server and forward its port 8080 to localhost (machine we're connecting from) port 1234

   ```
   ssh -L 1234:127.0.0.1:8080 mydspace.edu
   ```

   makes mydspace.edu:8080 accessible via localhost:1234 (type http://localhost:1234 in browser address bar); also opens ssh shell
   exit ssh to terminate port forwarding
   Alternatively:

   ```
   ssh -N -f -L 1234:127.0.0.1:8080 mydspace.edu
   ```

   run with -N and -f flags if you want ssh to go to background
   kill the ssh process to terminate port forwarding
2. **PuTTY client - port forwarding**
   Local port forwarding:

   ```
   Connection - SSH - Tunnels
   Source port: 1234
   Destination: localhost:8080
   Local
   Auto
   Add
   ```

   Once you're connected in PuTTY, visit http://localhost:1234/solr/ and you should see Solr's web interface. No browser configuration is necessary.

   Dynamic port forwarding/SOCKS proxy*:

   ```
   Connection - SSH - Tunnels
   Source port: 1234
   Dynamic
   Auto
   Add
   ```

   Once you're connected in PuTTY, you'll need to configure your browse to use localhost:1234 as a SOCKS proxy (and remove "localhost" and "127.0.0.1" from addresses to bypass this proxy - like in the next step)
3. **OpenSSH client - SOCKS proxy**
   connect to DSpace server and run a SOCKS proxy server on localhost port 1234; configure browser to use localhost:1234 as SOCKS proxy and remove "localhost" and "127.0.0.1" from addresses that bypass this proxy
   all browser requests now originate from dspace server (source IP is dspace server's IP) - dspace is the proxy server
   type http://localhost:8080in browser address bar - localhost here is the dspace server

   ```
   ssh -D 1234 mydspace.edu
   ```

*Note about Putty as SOCKS proxy - while it can be configured, it raises a security exception when Solr is accessed. If you figure this out, please add this method here.

Bypassing localhost restriction permanently

Privacy warning

Before you read this chapter, make sure you read Connecting to Solr and understand the consequences of any changes.

Instructions specific to Tomcat 7 and newer

Here's how you can:

1. turn off the localhost filter in Tomcat
2. replace it with a RemoteAddrValve and allow an enumerated set of IP addresses or subnets (in the following example the 127.0.0.1, 123.123.123.123 IPs and the 111.222.333.* subnet would be allowed):

Change your server.xml or alternatively your context fragment (i.e. conf/Catalina/localhost/solr.xml) like this:

```
<Context path="/solr" reloadable="true">
        <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127\.0\.0\.1|123\.123\.123\.123|111\.222\.233\.d+"/>
        <Parameter name="LocalHostRestrictionFilter.localhost" value="false" override="false" />
</Context>
```

Do not forget to include localhost (i.e. 127.0.0.1) in the allowed list, otherwise Discovery, OAI 2.0 and other things depending on Solr) won't work.

See also:

-
- Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Instructions specific to Tomcat 6 and older

Please, note that the syntax of the "allow" attribute changed in Tomcat 7 to a single regular expression. In Tomcat 6 and older, it was a comma-separated list of regular expressions, therefore this worked in Tomcat 6, but does not work in Tomcat 7+:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="111.222.233.*, 123.123.123.123, 127.0.0.1" />
```

See also: Tomcat 6 documentation: Remote Address Filter

Accessing Solr

Solr cores

DSpace contains a so-called multicore installation of Solr. That means that there are multiple Solr indexes and configurations sharing one Solr codebase. If you're familiar with Apache HTTPD, it is analogous to multiple virtual hosts running on one Apache server (separate configuration and webpages), except that individual Solr cores are accessible via different URL (as opposed to virtualhost IP:port).

The two Solr instances in DSpace Discovery are called "`search`" and "`statistics`". `search` contains data about communities, collections, items and bitstreams. `statistics` contains data about searches, accessing users, IPs etc. The two instances are accessible at following URLs (relative to the dspace server):

```
http://localhost:8080/solr/search/
http://localhost:8080/solr/statistics/
```

Solr admin interface

Both Solr cores have separate administration interfaces which let you view their respective schemas, configurations, set up logging and submit queries. The schema browser here is very useful to list fields (and their types) included in each index and even see an overview of most common values of individual fields with their frequency.

```
http://localhost:8080/solr/search/admin/
http://localhost:8080/solr/statistics/admin/
```

Solr queries

The base URL of the default Solr search handler is as follows:

```
http://localhost:8080/solr/search/search
http://localhost:8080/solr/statistics/search
```

Using the knowledge of particular fields from Solr Admin and Solr syntax (SolrQuerySyntax, CommonQueryParameters) you can make your own search requests. You can also read a brief tutorial to learn the query syntax quickly.
You can also look at the solr log file (in older dspace versions, this was logged to `catalina.out`) to see queries generated by XMLUI in real time:

```
tail -f /dspace/log/solr.log
```

(depending on your OS, Tomcat installation method and logging settings, the path may be different)

Solr responses

By default, Solr responses are returned in XML format. However, Solr can provide several other output formats including JSON and CSV. Discovery uses the javabin format. The Solr request parameter is wt (e.g. &wt=json). For more information, see Response Writers, QueryResponseWriters.
An interesting option is to specify an XSLT stylesheet that can transform the XML response (server-side) to any format you choose, typically HTML. Append &wt=xslt&tr=example.xsl to the Solr request URL. The .xsl files must be provided in the `[dspace]/solr/search/conf/xslt/` directory. For more information, see XsltResponseWriter.

PHP example

```
$solr_baseurl_dspace = "http://localhost:8080/solr/search/query?";
$solr_query = "test";
$solr_URL_dspace = $solr_baseurl_dspace."wt=phps&q=".urlencode($solr_query." AND withdrawn:false"); // use
withdrawn:false with DSpace newer than 1.8
$response_dspace = file_get_contents($solr_URL_dspace, false, stream_context_create(array('http' => array
('timeout' => 10))));
$result_dspace = unserialize($response_dspace);
$num_dspace = $result_dspace['response']['numFound'];
echo $num_dspace;
```

Keep in mind that although using the phps writer may be faster, it's not recommended for untrusted user data (see PHP unserialize() notes).

Examples

Date of last deposited item

To get all items (search.resourcetype:2) sorted by date accessioned (dc.date.accessioned_dt) in order from newest to oldest (desc; %20 is just an url-encoded space character):

```
http://localhost:8080/solr/search/select?q=search.resourcetype:2&sort=dc.date.accessioned_dt%20desc
```

Note:

search.resourcetype:2	items
search.resourcetype:3	communities
search.resourcetype:4	collections

To get only the first (newest) item (rows=1) with all but the date accessioned field filtered out (fl=dc.date.accessioned) and without the Solr response header (omitHeader=true):

```
http://localhost:8080/solr/search/select?q=search.resourcetype:2&sort=dc.date.accessioned_dt%
20desc&rows=1&fl=dc.date.accessioned&omitHeader=true
```

Top downloaded items by a specific user

```
http://localhost:8080/solr/statistics/select?indent=on&start=0&rows=10&fl=*%
2Cscore&qt=standard&wt=standard&explainOther=&hl.fl=&facet=true&facet.field=epersonid&q=type:0
```

Note:

facet.field=epersonid	You want to group by epersonid, which is the user id
type:0	Interested in bitstreams only

Number of items in a specific community

Community here is specified by its "`community_id`" - the identifier from the "`community`" table in database. The result is the "`numFound`" attribute of the "`result`" element. This example returns number of items (`search.resourcetype:2`) in community with community_id=85 (`location.comm:85`):

```
http://localhost:8080/solr/search/select/?q=location.comm:85+AND+search.resourcetype:2&start=0&rows=0&indent=on
```

Breakdown of submitted items per month

Show breakdown of items (`search.resourcetype:2`) submitted (`facet.date=dc.date.accessioned_dt`) per month (facet.date.gap=+MONTH) in the year 2016 (`facet.date.start=2016-01-01T00:00:00Z&facet.date.end=2017-01-01T00:00:00Z`):

```
http://localhost:8080/solr/search/select?indent=on&rows=0&facet=true&facet.date=dc.date.accessioned_dt&facet.
date.start=2016-01-01T00:00:00Z&facet.date.end=2017-01-01T00:00:00Z&facet.date.gap=%2B1MONTH&q=search.
resourcetype:2
```

Statistics breakdown per event type

Starting from DSpace 3, there is a `statistics_type` field in the `statistics` core that contains the "usage event type". Currently, the available types are `search`, `view`, `search_result` and `workflow`. Here's how to get event breakdown by type, excluding robots (`isBot:false`):

```
http://localhost:8080/solr/statistics/select?indent=on&rows=0&facet=true&facet.field=statistics_type&q=isBot:
false
```

Statistics: breakdown of downloads per month

Show breakdown of bitstream (`type:0`) downloads per month in the year 2016, excluding robots (`isBot:false`):

```
http://localhost:8080/solr/statistics/select?indent=on&rows=0&facet=true&facet.date=time&facet.date.start=2016-
01-01T00:00:00Z&facet.date.end=2017-01-01T00:00:00Z&facet.date.gap=%2B1MONTH&q=type:0+AND+isBot:false
```

Statistics: number of downloads (item views) for a specific item per month

Show bitstream (`type:0`) downloads per month in the year 2016, excluding robots (`isBot:false`), for a specific item (2163 in the example):

```
http://localhost:8080/solr/statistics/select?indent=on&rows=0&facet=true&facet.date=time&facet.date.start=2016-
01-01T00:00:00Z&facet.date.end=2017-01-01T00:00:00Z&facet.date.gap=%2B1MONTH&q=type:0+owningItem:2163&fq=-isBot:
true&fq=-(bundleName:[*+TO+*]-bundleName:ORIGINAL)&fq=-(statistics_type:[*+TO+*]+-statistics_type:view)
```

Statistics: number of total downloads in a given time span

Show the total repository-wide bitstream (`type:0`) downloads, excluding robots (`isBot:false`), for a specific duration (September 1 2017 through September 1 2018). No need for faceting to get a total count:

```
http://localhost:8080/solr/statistics/select?indent=on&rows=0&q=time:[2017-09-01T00:00:00Z+TO+2018-09-01T00:00:
00Z]+AND+type:0+AND+isBot:false
```

Querying Solr from XMLUI

Since Solr returns its responses in XML, it's possible and easy to call custom Solr queries from XMLUI, process the XML response with XSLT and display the results in human-readable form on the HTML page.
There are two ways how to do that - *synchronously* in Cocoon or *asynchronously* using AJAX (JavaScript) after the page is loaded. Solr queries are usually very fast, so only synchronous calls will be shown here.

You can include another XML document to be processed by XSLT using the document() function. The parameter to this function is a string with the path to the XML document to process. This can be either a static .xml file stored on the server filesystem or a URL, which will be fetched at time of processing. For Solr, the later is what we need. Furthermore, we need to distinguish templates for processing this external XML document as opposed to the input XML document. We'll do this using the `mode` attribute and define a different processing mode for each query.

```
<xsl:apply-templates select="document('http://localhost:8080/solr/search/select?q=search.resourcetype:2&amp;
sort=dc.date.accessioned_dt%20desc&amp;rows=1&amp;fl=dc.date.accessioned_dt&amp;omitHeader=true'))"
mode="solr-response"/>
```

Now we need to define a template with the same mode that matches elements contained in the Solr response XML:

```
<xsl:template match="/response/result/doc/date" mode="solr-response">
    Last item was imported: <xsl:value-of select="text()"/>
</xsl:template>
```

Furthermore, we don't want to hardcode the `http://localhost:8080` Solr URL, because this can be changed in config file and that would break the template. So we'll call a Java function from XSLT to retrieve the configured Solr URL. See the complete example in the next section.

Examples

Date of last deposited item

For description of the query parameters, see above.

1. Add the confman namespace and "confman" to exclude-result-prefixes. (For explanation, see how to Call Java methods from XSLT (Manakin))

   ```
   <xsl:stylesheet
   ...
       xmlns:confman="org.dspace.core.ConfigurationManager"
       exclude-result-prefixes="... confman">
   ```

2. Add this simple template to process the Solr query result. More complex date formatting can be done easily in XSLT 2.0 (see XSLT 2.0 spec), however Cocoon still uses XSLT 1.0 (see DS-995). It is currently also possible to call Java functions to do date formatting.

   ```
   <xsl:template match="/response/result/doc/date" mode="lastItem">
       Last item was imported: <xsl:value-of select="substring(text(), 1, 10)"/>
   </xsl:template>
   ```

3. Add the following code to the place where you want the resulting text to appear:

   ```
   <xsl:variable name="solr-search-url" select="confman:getProperty('discovery', 'search.server')"/>
   <xsl:apply-templates select="document(concat($solr-search-url, '/select?q=search.resourcetype:2&amp;
   sort=dc.date.accessioned_dt%20desc&amp;rows=1&amp;fl=dc.date.accessioned_dt&amp;omitHeader=true'))"
   mode="lastItem"/>
   ```

   For example, to add it after the list of Recent items in Mirage, override its template like this:

```
<xsl:template match="dri:referenceSet[@type = 'summaryList' and @n='site-last-submitted']" priority="2">
    <xsl:apply-templates select="dri:head"/>
    <!-- Here we decide whether we have a hierarchical list or a flat one -->
    <xsl:choose>
        <xsl:when test="descendant-or-self::dri:referenceSet/@rend='hierarchy' or ancestor::dri:
referenceSet/@rend='hierarchy'">
            <ul>
                <xsl:apply-templates select="*[not(name()='head')]" mode="summaryList"/>
            </ul>
        </xsl:when>
        <xsl:otherwise>
            <ul class="ds-artifact-list">
                <xsl:apply-templates select="*[not(name()='head')]" mode="summaryList"/>
            </ul>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:variable name="solr-search-url" select="confman:getProperty('discovery', 'search.server')"/>
    <xsl:apply-templates select="document(concat($solr-search-url, '/select?q=search.resourcetype:2&amp;
sort=dc.date.accessioned_dt%20desc&amp;rows=1&amp;fl=dc.date.accessioned_dt&amp;omitHeader=true'))"
mode="lastItem"/>
</xsl:template>
```

Multicore join queries

Solr supports join queries across multiple cores since Solr 4.0. Thus it's also supported in DSpace 4.0 (which includes Solr 4.4).

example query (not tested)

```
http://localhost:8080/solr/search/select/?q=*:*&fq={!join from=owningItem to=search.resourceid
fromIndex=statistics}title:"Testing title"
```

"AND" search as default

Up to and including DSpace 5 (see DS-2809), Discovery uses the "OR" operator as default if you don't specify an operator between your query keywords. So searching for "John Doe" will also return entries like "Jane Doe" and "John Connor". If you want to change that, you have to edit the `schema.xml` file of the Solr `search` core:

In `[dspace]/solr/search/conf/schema.xml`, find this line:

```
<solrQueryParser defaultOperator="OR"/>
```

and change it to

```
<solrQueryParser defaultOperator="AND"/>
```

Then restart your servlet container (Tomcat).

Warning

It's not officially recommended to change the `defaultOperator` setting. Some unrelated Discovery features might stop working if you do this. I haven't noticed anything wrong, but you might. If something breaks, make sure to notify us and we'll try to fix it or remove this tip.

Deleting Solr index data

If for whatever reason you need to delete the data in your index (which would normally be followed by running `[dspace]/bin/dspace index-discovery` (in DSpace versions older than 4.x, it was called `[dspace]/bin/dspace update-discovery-index`), but you can use the -b parameter instead to reindex everything), here's how you can do it:

Solr delete query

If Solr is running, you can access the following URL from the server where Solr is installed (remember the default localhost restriction):

```
$ curl "http://localhost:8080/solr/search/update?stream.body=<update><delete><query>*:*</query></delete><commit
/></update>"
```

This will delete all documents in the `search` (Discovery) core.

You can verify the number of documents in the core by running the following query and checking the value of the `numFound` attribute in the output:

```
$ curl "http://localhost:8080/solr/search/select/?q=*:*&rows=0"
<?xml version="1.0" encoding="UTF-8"?>
<response>
<lst name="responseHeader"><int name="status">0</int><int name="QTime">5</int><lst name="params"><str name="
rows">0</str><str name="q">*:*</str></lst></lst><result name="response" numFound="0" start="0"/>
</response>
```

The URL listed in the examples is the default Solr URL in DSpace. If you changed it, you can find it in `search.server` in `[dspace]/config/modules/discovery.cfg` (DSpace 1.8+) or in `solr.log.server` in [dspace]/config/dspace.cfg (DSpace 1.7).

Source: Solr Wiki FAQ: How can I delete all documents from my index?

Manually delete Solr index files

If your Solr is broken and you can't issue queries, you can still delete the index files manually:

```
$ rm -rf [dspace]/solr/search/data/
```

Then restart the servlet container or reload the `solr` webapp.

See also:

- Solr: How can I delete all documents from my index?
- DSpace: deleted wrong directory

Set up Solritas (VelocityResponseWriter)

Solritas is a generic search interface on top of a Solr index. It can be useful if you want to explore the contents of a Solr index (core) using facets.

To set it up in DSpace 3.0 (which uses Solr 3.5.0):

- download apache-solr-3.5.0.tgz from http://archive.apache.org/dist/lucene/solr/3.5.0/
- `tar xvzf apache-solr-3.5.0.tgz`
- `mkdir [dspace]/solr/lib`
- `cp ./apache-solr-3.5.0/dist/apache-solr-velocity-3.5.0.jar [dspace]/solr/lib`
- `cp ./apache-solr-3.5.0/contrib/velocity/lib/{commons-beanutils-1.7.0.jar,commons-collections-3.2.1.jar, velocity-1.6.4.jar,velocity-tools-2.0.jar} [dspace]/solr/lib`
- edit `[dspace]/solr/solr.xml` and add the `sharedLib` attribute:

  ```
  <solr persistent="false" sharedLib="lib">
  ```

- edit the solrconfig.xml file of each core where you want to use Solritas. Example for the "search" core: add the velocity ResponseWriter and requestHandler in `[dspace]/solr/search/conf/solrconfig.xml`:

```
<queryResponseWriter name="velocity" class="solr.VelocityResponseWriter"/>

<requestHandler name="/browse" class="solr.SearchHandler">
    <lst name="defaults">
      <str name="v.template">browse</str>
      <str name="v.contentType">text/html;charset=UTF-8</str>
      <str name="title">Solritas</str>
      <str name="wt">velocity</str>
      <str name="defType">dismax</str>
      <str name="q.alt">*:*</str>
      <str name="rows">10</str>
      <str name="fl">*,score</str>
      <str name="facet">on</str>
      <str name="facet.field">title</str>
      <str name="facet.mincount">1</str>
      <str name="qf">
         text^0.5 title^1.5
      </str>
    </lst>
    <!--<lst name="invariants">-->
      <!--<str name="v.base_dir">/solr/contrib/velocity/src/main/templates</str>-->
    <!--</lst>-->
 </requestHandler>
```

- `cp -r ./apache-solr-3.5.0/example/solr/conf/velocity [dspace]/solr/search/conf/`
- restart Tomcat
- Solritas should be available at http://localhost:8080/solr/search/browse/

It should also be possible to use it in other versions of DSpace (starting from 1.6), but these use different versions of Solr, so modify the procedure accordingly (and expect other caveats):

DSpace 6	Solr 4.10.2
DSpace 5	Solr 4.10.2
DSpace 4	Solr 4.4.0
DSpace 3	Solr 3.5.0
DSpace 1.8	Solr 3.3.0
DSpace 1.7	Solr 1.4.1
DSpace 1.6	Solr 1.3.0

Note: In older versions, you may need to specify the queryResponseWriter class as `org.apache.solr.request.VelocityResponseWriter` (I haven't tested it, though)

Resources:

- http://wiki.apache.org/solr/VelocityResponseWriter
- http://lucene.472066.n3.nabble.com/ClassNotFoundException-org-apache-solr-response-VelocityResponseWriter-tp787256p787349.html

Guidepost

Other pages on this wiki describing Solr and Discovery.

- Discovery Official DSpace 3.x documentation
- DSpace Discovery Discovery proposal & purpose, intro video, Discovery 1.8 changes & configuration
- DSpace Discovery HowTo Discovery screenshots (before Discovery was included in DSpace), most content obsolete (pre-1.7.0)

See also:

- Solr Tutorial
- ajax-solr, a JavaScript library for creating user interfaces to Solr.
- /var/log/tomcat6/catalina.out