Solr Tutorial

18 Apr, 2015 in solr query / solr search tagged solr 5 tutorial / solr facet / solr index / solr query / solr query example / solr search / solr search example / solr tutorial by yonik (updated on April 9, 2016)

Solr Tutorial

This tutorial is for Solr 5.5 and later.
If you have Solr 4, check out the Solr 4 Tutorial.

1. Download Solr

Download and unpack the latest Solr release from the Apache download mirrors. You may want to check out the Solr Prerequisites as well.

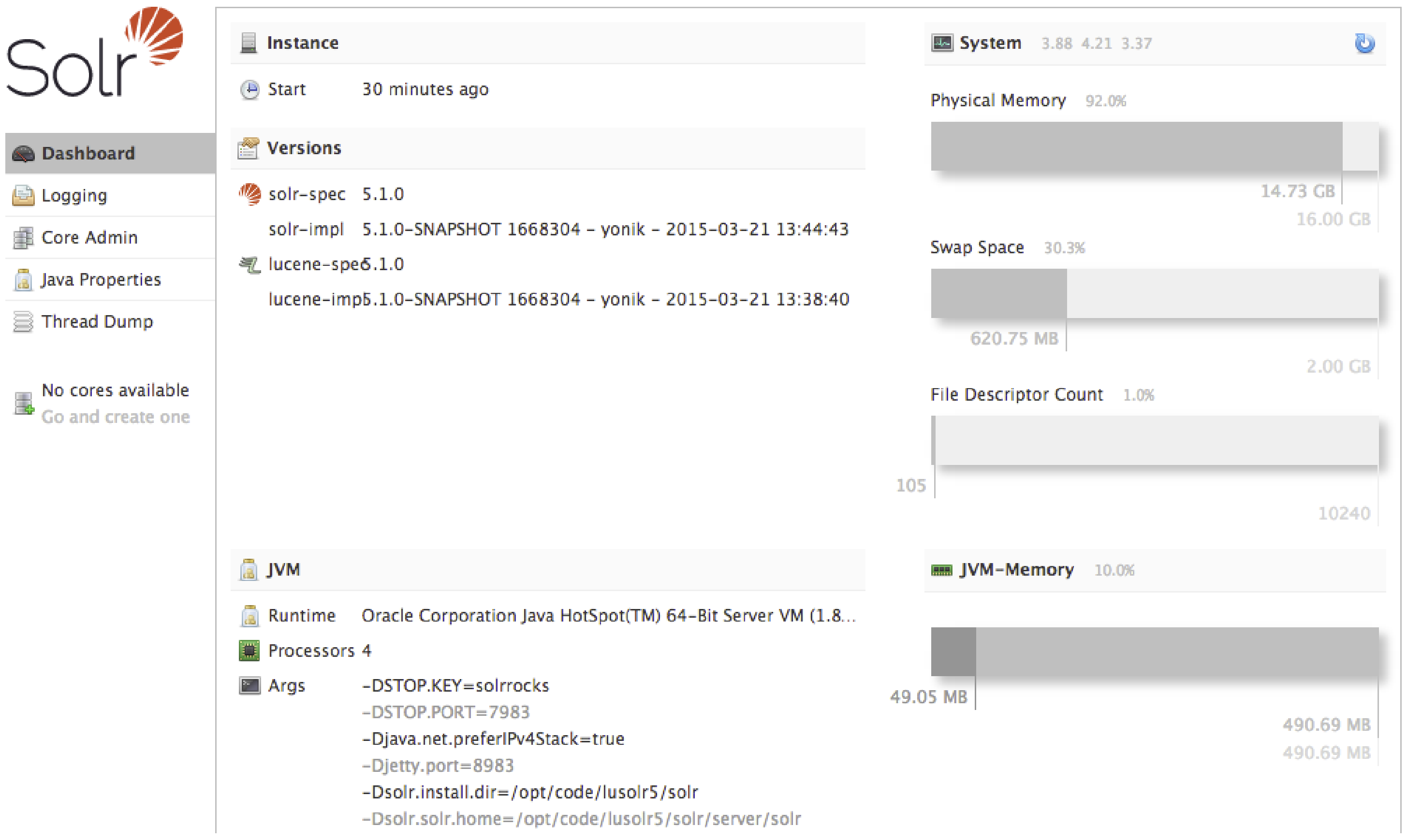
2. Start Solr

```
$ bin/solr start            # this starts solr
$ bin/solr create -c demo   # this creates a document collection called "demo"
```

3. Go!

You're now ready to start using Solr!
To verify it's up and running, you can point your browser at the admin page:

```
http://localhost:8983/solr/
```



```
Start        30 minutes ago

solr-spec    5.1.0
solr-impl    5.1.0-SNAPSHOT 1668304 - yonik - 2015-03-21 13:44:43
lucene-spec  5.1.0
lucene-impl  5.1.0-SNAPSHOT 1668304 - yonik - 2015-03-21 13:38:40

Runtime      Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8...
Processors   4
Args         -DSTOP.KEY=solrrocks
             -DSTOP.PORT=7983
             -Djava.net.preferIPv4Stack=true
             -Djetty.port=8983
             -Dsolr.install.dir=/opt/code/lusolr5/solr
             -Dsolr.solr.home=/opt/code/lusolr5/solr/server/solr

System  3.88  4.21  3.37
Physical Memory  92.0%     14.73 GB / 16.00 GB
Swap Space  30.3%          620.75 MB / 2.00 GB
File Descriptor Count  1.0%   105 / 10240
JVM-Memory  10.0%          49.05 MB / 490.69 MB / 490.69 MB
```

Indexing and Retrieving a Document

Now that Solr is running, we can add a document (also known as "indexing" a document) with curl:

```
$ curl http://localhost:8983/solr/demo/update -d '
[
 {"id" : "book1",
  "title_t" : "The Way of Kings",
  "author_s" : "Brandon Sanderson"
 }
]'
```

And then we can ask for it back:

```
$ curl http://localhost:8983/solr/demo/get?id=book1

{
  "doc": {
    "id" : "book1",
    "author_s": "Brandon Sanderson",
    "title_t" : "The Way of Kings",
    "_version_": 1410390803582287872
  }
}
```

Of course for queries, you can always just use your browser and click on the link
http://localhost:8983/solr/demo/get?id=book1
or cut'n'paste the URL into your browser and modify the query directly in the address bar to try out different requests.

Dynamic Fields

The "id" field is already pre-defined in every schema. Lucene and Solr need to know the types of fields so that they can be indexed in the correct way. There are a number of options for defining new fields:

- Edit the schema.xml file to define the fields.
- Use the Schema API to add the new fields.
- Use dynamicFields, a form of convention-over-configuration that maps field names to field types based on patterns in the field name. For example, every field ending in "_i" is taken to be an integer.
- Use "schemaless" mode, where field types are auto-detected (guessed) based on the first value seen for that field

For other document fields in this tutorial, we have chosen to use **convention over configuration** via **dynamic fields**. Dynamic fields includes the essential benefits of **schemaless** – namely the ability to **add new fields on the fly** without having to pre-define them.

Our schema has some common dynamicField patterns defined for use:

Field Suffix	Multivalued Suffix	Type	Description
_t	_txt	text_general	Indexed for full-text search so individual words or phrases may be matched.

_s	_ss	string	A string value is indexed as a single unit. This is good for sorting, faceting, and analytics. It’s not good for full-text search.
_i	_is	int	a 32 bit signed integer
_l	_ls	long	a 64 bit signed long
_f	_fs	float	IEEE 32 bit floating point number (single precision)
_d	_ds	double	IEEE 64 bit floating point number (double precision)
_b	_bs	boolean	true or false
_dt	_dts	date	A date in Solr’s date format
_p		location	A lattitude and longitude pair for geo-spatial search

Updating a Document

Let’s update **book1** and add **cat_s**, a category field, a publication year, and an ISBN. Via dynamic fields, a field name ending with **_i** tells Solr to treat the value as an **integer**, while a field name ending with **_s** is treated as a **string**.

```
$ curl http://localhost:8983/solr/demo/update -d '
[
 {"id"         : "book1",
  "cat_s"      : { "add" : "fantasy" },
  "pubyear_i"  : { "add" : 2010 },
  "ISBN_s"     : { "add" : "978-0-7653-2635-5" }
 }
]'
```

Now go ahead and ask for the document back, and you should see the new fields:

```
$ curl http://localhost:8983/solr/demo/get?id=book1
```

See Atomic Updates for more document update options.

Solr Search Requests

First, lets add a few more documents so we have something to search for. This time we’ll demonstrate indexing documents in CSV (comma separated values) format:

```
$ curl http://localhost:8983/solr/demo/update?commitWithin=5000 -H 'Content-type:text/csv' -d '
id,cat_s,pubyear_i,title_t,author_s,series_s,sequence_i,publisher_s
book1,fantasy,2010,The Way of Kings,Brandon Sanderson,The Stormlight Archive,1,Tor
```

```
book2,fantasy,1996,A Game of Thrones,George R.R. Martin,A Song of Ice and Fire,1,Bantam
book3,fantasy,1999,A Clash of Kings,George R.R. Martin,A Song of Ice and Fire,2,Bantam
book4,sci-fi,1951,Foundation,Isaac Asimov,Foundation Series,1,Bantam
book5,sci-fi,1952,Foundation and Empire,Isaac Asimov,Foundation Series,2,Bantam
book6,sci-fi,1992,Snow Crash,Neal Stephenson,Snow Crash,,Bantam
book7,sci-fi,1984,Neuromancer,William Gibson,Sprawl trilogy,1,Ace
book8,fantasy,1985,The Black Company,Glen Cook,The Black Company,1,Tor
book9,fantasy,1965,The Black Cauldron,Lloyd Alexander,The Chronicles of Prydain,2,Square Fish
book10,fantasy,2001,American Gods,Neil Gaiman,,,Harper'
```

We added the **`commitWithin=5000`** parameter to indicate that we would like our updates to be visible within 5000 milliseconds (5 seconds). The Lucene library that Solr uses for full-text search works off of point-in-time snapshots that must be periodically refreshed in order for queries to see new changes.

Note that although we often use JSON in our examples, Solr is actually **data format agnostic** – you're not artificially tied to any particular transfer-syntax or serialization format such as JSON or XML.

Your First Solr Search Request

Now let's query our book collection! For example, we can find all books with "black" in the title field:

```
http://localhost:8983/solr/demo/query?
  q=title_t:black
  fl=author_s,title_t
```

The **`fl`** parameter stands for "field list" and specifies what stored fields should be returned from documents matching the query. We should see a result like the following:

```
{"response":{"numFound":2,"start":0,"docs":[
    {
      "title_t":"The Black Company",
      "author_s":"Glen Cook"},
    {
      "title_t":"The Black Cauldron",
      "author_s":"Lloyd Alexander"}]
}}
```

See Solr Query for more solr query examples and syntax.

Solr Search Request in JSON

If you prefer using JSON to search the index, you can use the JSON Request API:

```
$ curl http://localhost:8983/solr/demo/query -d '
{
  "query" : "title_t:black",
  "fields" : ["title_t", "author_s"]
}'
```

Sorting and Paging Search Results

By default, Solr will return the top 10 documents ordered by highest score (relevance) first. Let's change things up and return the top 3 search results, limiting them to books published by Bantam, and sorting by publication year descending:

```
$ curl http://localhost:8983/solr/demo/query -d '
```

```
q=*:*&
fq=publisher_s:Bantam&
rows=3&
sort=pubyear_i desc&
fl=title_t,pubyear_i'
```

And we get the response as requested:

```
"response":{"numFound":5,"start":0,"docs":[
    {
      "pubyear_i":1999,
      "title_t":["A Clash of Kings"]},
    {
      "pubyear_i":1996,
      "title_t":["A Game of Thrones"]},
    {
      "pubyear_i":1992,
      "title_t":["Snow Crash"]}]
}
```

Parameter Explanation:

- **`q=*:*`** – The *:* query matches all documents in the index.
- **`fq=publisher_s:Bantam`** – "fq" parameters are filter queries, which don't affect scoring, but filter out documents that don't match the given query. These are cached separately and reused across different requests, greatly accelerating throughput. See Advanced Filter Caching in Solr for more details.
- **`sort=pubyear_i desc`** – This sorts on the "pubyear_i" field descending. Solr has many advanced sorting options such as tie-break sorts and sorting by a function of document fields!
- **`rows=3`** – "rows" specifies the number of results to return, while "start" specifies an offset into the sorted list for paging purposes. Also see Deep Paging for options to efficiently page deeply into result sets.

// WORK IN PROGRESS, MORE TO FOLLOW SOON

JSON Facet API
Facet Functions & Analytics
Sub-Facets / Nested Facets

Next Steps

Welcome to the Apache Solr community! Now that you've discovered just how easy it is to get up and running, you should check out all of the other powerful features that Solr has to offer.

Remember to subscribe to the **solr-user** mailing list where you'll meet a ton of helpful users and developers!