Solr Quick Start

This document covers getting Solr up and running, ingesting a variety of data sources into multiple collections, and getting a feel for the Solr administrative and search interfaces.

Requirements

To follow along with this tutorial, you will need...

1. To meet the system requirements
2. An Apache Solr release (download). This tutorial was written using Apache Solr 6.5.1.

Getting Started

Please run the browser showing this tutorial and the Solr server on the same machine so tutorial links will correctly point to your Solr server.

Begin by unzipping the Solr release and changing your working directory to the subdirectory where Solr was installed. Note that the base directory name may vary with the version of Solr downloaded. For example, with a shell in UNIX, Cygwin, or MacOS:

```
/:$ ls solr*
solr-6.5.1.zip
/:$ unzip -q solr-6.5.1.zip
/:$ cd solr-6.5.1/
```

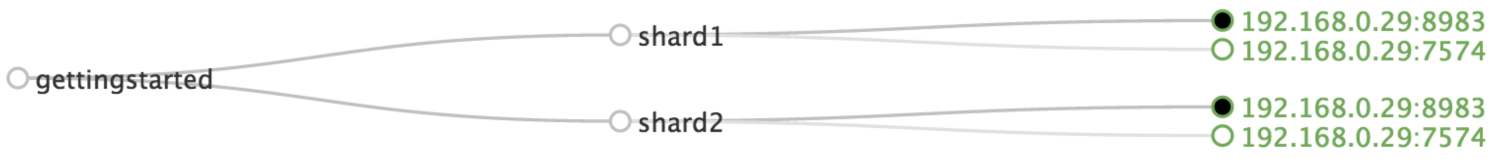
To launch Solr, run: `bin/solr start -e cloud -noprompt`

```
/solr-6.5.1:$ bin/solr start -e cloud -noprompt

Welcome to the SolrCloud example!

Starting up 2 Solr nodes for your example SolrCloud cluster.
...
```

```
Started Solr server on port 8983 (pid=8404). Happy searching!
...

Started Solr server on port 7574 (pid=8549). Happy searching!
...

SolrCloud example running, please visit http://localhost:8983/solr

/solr-6.5.1:$ _
```

You can see that the Solr is running by loading the Solr Admin UI in your web browser: http://localhost:8983/solr/. This is the main starting point for administering Solr.

Solr will now be running two "nodes", one on port 7574 and one on port 8983. There is one collection created automatically, `gettingstarted`, a two shard collection, each with two replicas. The Cloud tab in the Admin UI diagrams the collection nicely:



Indexing Data

Your Solr server is up and running, but it doesn't contain any data. The Solr install includes the `bin/post` tool in order to facilitate getting various types of documents easily into Solr from the start. We'll be using this tool for the indexing examples below.

You'll need a command shell to run these examples, rooted in the Solr install directory; the shell from where you launched Solr works just fine.

- NOTE: Currently the `bin/post` tool does not have a comparable Windows script, but the underlying Java program invoked is available. See the Post Tool, Windows section for details.

Indexing a directory of "rich" files

Let's first index local "rich" files including HTML, PDF, Microsoft Office formats (such as MS Word), plain text and many other formats. `bin/post`features the ability to crawl a directory of files, optionally recursively even, sending the raw content of each file into Solr for extraction and indexing. A Solr install includes a `docs/` subdirectory, so that makes a convenient set of (mostly) HTML files built-in to start with.

```
bin/post -c gettingstarted docs/
```

Here's what it'll look like:

```
/solr-6.5.1:$ bin/post -c gettingstarted docs/
java -classpath /solr-6.5.1/dist/solr-core-6.5.1.jar -Dauto=yes -
Dc=gettingstarted -Ddata=files -Drecursive=yes
org.apache.solr.util.SimplePostTool docs/
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingstarted/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,

Entering recursive mode, max depth=999, delay=0s
Indexing directory docs (3 files, depth=0)
POSTing file index.html (text/html) to [base]/extract
POSTing file quickstart.html (text/html) to [base]/extract
POSTing file SYSTEM_REQUIREMENTS.html (text/html) to [base]/extract
Indexing directory docs/changes (1 files, depth=1)
POSTing file Changes.html (text/html) to [base]/extract
...
4329 files indexed.
COMMITting Solr index changes to
http://localhost:8983/solr/gettingstarted/update...
Time spent: 0:01:16.252
```

The command-line breaks down as follows:

- `-c gettingstarted`: name of the collection to index into
- `docs/`: a relative path of the Solr install `docs/` directory

You have now indexed thousands of documents into the `gettingstarted`collection in Solr and committed these changes. You can search for "solr" by loading the Admin UI Query tab, enter "solr" in the `q` param (replacing `*:*`, which matches all documents), and "Execute Query". See the Searching section below for more information.

To index your own data, re-run the directory indexing command pointed to your own directory of documents. For example, on a Mac instead of `docs/` try `~/Documents/` or `~/Desktop/`! You may want to start from a clean, empty system again rather than have your content in addition to the Solr `docs/` directory; see the Cleanup section below for how to get back to a clean starting point.

Indexing Solr XML

Solr supports indexing structured content in a variety of incoming formats. The historically predominant format for getting structured content into Solr has been Solr XML. Many Solr indexers have been coded to process domain content into Solr XML output, generally HTTP POSTed directly to Solr's `/update` endpoint.

Solr's install includes a handful of Solr XML formatted files with example data (mostly mocked tech product data). NOTE: This tech product data has a more domain-specific configuration, including schema and browse UI. The `bin/solr` script includes built-in support for this by running `bin/solr start -e techproducts` which not only starts Solr but also then indexes this data too (be sure to `bin/solr stop -all` before trying it out). However, the example below assumes Solr was started with `bin/solr start -e cloud` to stay consistent with all examples on this page, and thus the collection used is "gettingstarted", not "techproducts".

Using `bin/post`, index the example Solr XML files in `example/exampledocs/`:

```
bin/post -c gettingstarted example/exampledocs/*.xml
```

Here's what you'll see:

```
/solr-6.5.1:$ bin/post -c gettingstarted example/exampledocs/*.xml
java -classpath /solr-6.5.1/dist/solr-core-6.5.1.jar -Dauto=yes -
Dc=gettingstarted -Ddata=files org.apache.solr.util.SimplePostTool
example/exampledocs/gb18030-example.xml ...
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingstarted/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,
POSTing file gb18030-example.xml (application/xml) to [base]
POSTing file hd.xml (application/xml) to [base]
POSTing file ipod_other.xml (application/xml) to [base]
POSTing file ipod_video.xml (application/xml) to [base]
POSTing file manufacturers.xml (application/xml) to [base]
POSTing file mem.xml (application/xml) to [base]
POSTing file money.xml (application/xml) to [base]
POSTing file monitor.xml (application/xml) to [base]
POSTing file monitor2.xml (application/xml) to [base]
POSTing file mp500.xml (application/xml) to [base]
POSTing file sd500.xml (application/xml) to [base]
POSTing file solr.xml (application/xml) to [base]
POSTing file utf8-example.xml (application/xml) to [base]
POSTing file vidcard.xml (application/xml) to [base]
14 files indexed.
COMMITting Solr index changes to
http://localhost:8983/solr/gettingstarted/update...
Time spent: 0:00:02.077
```

...and now you can search for all sorts of things using the default Solr Query Syntax (a superset of the Lucene query syntax)...

NOTE: You can browse the documents indexed at http://localhost:8983/solr/gettingstarted/browse. The `/browse` UI allows getting a feel for how Solr's technical capabilities can be worked with in a familiar, though a bit rough and prototypical, interactive HTML view. (The `/browse` view defaults to assuming the `gettingstarted` schema and data are a catch-all mix of structured XML, JSON, CSV example data, and unstructured rich documents. Your own data may not look ideal at first, though the `/browse` templates are customizable.)

Indexing JSON

Solr supports indexing JSON, either arbitrary structured JSON or "Solr JSON" (which is similar to Solr XML).

Solr includes a small sample Solr JSON file to illustrate this capability. Again using `bin/post`, index the sample JSON file:

```
bin/post -c gettingstarted example/exampledocs/books.json
```

You'll see:

```
/solr-6.5.1:$ bin/post -c gettingstarted example/exampledocs/books.json
java -classpath /solr-6.5.1/dist/solr-core-6.5.1.jar -Dauto=yes -
Dc=gettingstarted -Ddata=files org.apache.solr.util.SimplePostTool
example/exampledocs/books.json
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingstarted/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,

POSTing file books.json (application/json) to [base]/json/docs
1 files indexed.
COMMITting Solr index changes to
http://localhost:8983/solr/gettingstarted/update...
Time spent: 0:00:00.493
```

For more information on indexing Solr JSON, see the Solr Reference Guide section Solr-Style JSON

To flatten (and/or split) and index arbitrary structured JSON, a topic beyond this quick start guide, check out Transforming and Indexing Custom JSON data.

Indexing CSV (Comma/Column Separated Values)

A great conduit of data into Solr is via CSV, especially when the documents are homogeneous by all having the same set of fields. CSV can be conveniently exported from a spreadsheet such as Excel, or exported from databases such as MySQL. When getting started with Solr, it can often be easiest to get your structured data into CSV format and then index that into Solr rather than a more sophisticated single step operation.

Using `bin/post` index the included example CSV file:

```
bin/post -c gettingstarted example/exampledocs/books.csv
```

In your terminal you'll see:

```
/solr-6.5.1:$ bin/post -c gettingstarted example/exampledocs/books.csv
java -classpath /solr-6.5.1/dist/solr-core-6.5.1.jar -Dauto=yes -
Dc=gettingstarted -Ddata=files org.apache.solr.util.SimplePostTool
example/exampledocs/books.csv
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/gettingstarted/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,

POSTing file books.csv (text/csv) to [base]
1 files indexed.
COMMITting Solr index changes to
http://localhost:8983/solr/gettingstarted/update...
Time spent: 0:00:00.109
```

For more information, see the Solr Reference Guide section CSV Formatted Index Updates

Other indexing techniques

- Import records from a database using the Data Import Handler (DIH).
- Use SolrJ from JVM-based languages or other Solr clients to programmatically create documents to send to Solr.
- Use the Admin UI Documents tab to paste in a document to be indexed, or select `Document Builder` from the `Document Type` dropdown to build a document one field at a time. Click on the `Submit Document` button below the form to index your document.

Updating Data

You may notice that even if you index content in this guide more than once, it does not duplicate the results found. This is because the example `schema.xml` specifies a "`uniqueKey`" field called "`id`". Whenever you POST commands to Solr to add a document with the same value for the `uniqueKey` as an existing document, it automatically replaces it for you. You can see that that has happened by looking at the values for `numDocs` and `maxDoc` in the core-specific Overview section of the Solr Admin UI.

`numDocs` represents the number of searchable documents in the index (and will be larger than the number of XML, JSON, or CSV files since some files contained more than one document). The maxDoc value may be larger as the maxDoc count includes logically deleted documents that have not yet been physically removed from the index. You can re-post the sample files over and over again as much as you want and `numDocs` will never increase, because the new documents will constantly be replacing the old.

Go ahead and edit any of the existing example data files, change some of the data, and re-run the SimplePostTool command. You'll see your changes reflected in subsequent searches.

Deleting Data

You can delete data by POSTing a delete command to the update URL and specifying the value of the document's unique key field, or a query that matches multiple documents (be careful with that one!). Since these commands are smaller, we specify them right on the command line rather than reference a JSON or XML file.

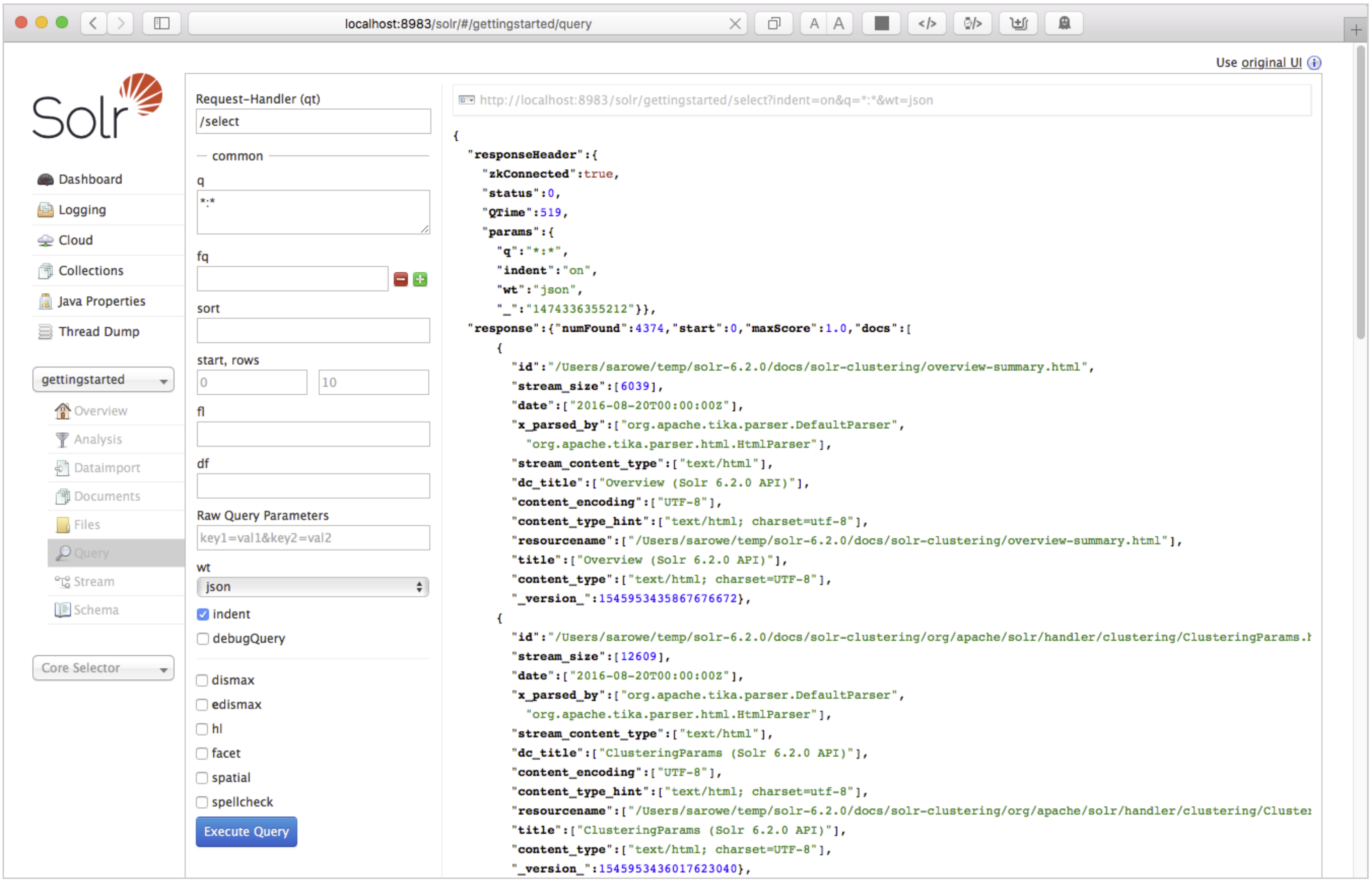
Execute the following command to delete a specific document:

```
bin/post -c gettingstarted -d "<delete><id>SP2514N</id></delete>"
```

Searching

Solr can be queried via REST clients, cURL, wget, Chrome POSTMAN, etc., as well as via the native clients available for many programming languages.

The Solr Admin UI includes a query builder interface - see the `gettingstarted` query tab athttp://localhost:8983/solr/#/gettingstarted/query. If you click the `Execute Query` button without changing anything in the form, you'll get 10 documents in JSON format (`*:*` in the `q` param matches all documents):



Request-Handler (qt): /select
common
q: *:*
fq
sort
start, rows: 0 10
fl
df
Raw Query Parameters: key1=val1&key2=val2
wt: json
☑ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck

http://localhost:8983/solr/gettingstarted/select?indent=on&q=*:*&wt=json

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":519,
    "params":{
      "q":"*:*",
      "indent":"on",
      "wt":"json",
      "_":"1474336355212"}},
  "response":{"numFound":4374,"start":0,"maxScore":1.0,"docs":[
      {
        "id":"/Users/sarowe/temp/solr-6.2.0/docs/solr-clustering/overview-summary.html",
        "stream_size":[6039],
        "date":["2016-08-20T00:00:00Z"],
        "x_parsed_by":["org.apache.tika.parser.DefaultParser",
          "org.apache.tika.parser.html.HtmlParser"],
        "stream_content_type":["text/html"],
        "dc_title":["Overview (Solr 6.2.0 API)"],
        "content_encoding":["UTF-8"],
        "content_type_hint":["text/html; charset=utf-8"],
        "resourcename":["/Users/sarowe/temp/solr-6.2.0/docs/solr-clustering/overview-summary.html"],
        "title":["Overview (Solr 6.2.0 API)"],
        "content_type":["text/html; charset=UTF-8"],
        "_version_":1545953435867676672},
      {
        "id":"/Users/sarowe/temp/solr-6.2.0/docs/solr-clustering/org/apache/solr/handler/clustering/ClusteringParams.h
        "stream_size":[12609],
        "date":["2016-08-20T00:00:00Z"],
        "x_parsed_by":["org.apache.tika.parser.DefaultParser",
          "org.apache.tika.parser.html.HtmlParser"],
        "stream_content_type":["text/html"],
        "dc_title":["ClusteringParams (Solr 6.2.0 API)"],
        "content_encoding":["UTF-8"],
        "content_type_hint":["text/html; charset=utf-8"],
        "resourcename":["/Users/sarowe/temp/solr-6.2.0/docs/solr-clustering/org/apache/solr/handler/clustering/Cluster
        "title":["ClusteringParams (Solr 6.2.0 API)"],
        "content_type":["text/html; charset=UTF-8"],
        "_version_":1545953436017623040},
```

The URL sent by the Admin UI to Solr is shown in light grey near the top right of the above screenshot - if you click on it, your browser will show you the raw response. To use cURL, give the same URL in quotes on the `curl`command line:

```
curl "http://localhost:8983/solr/gettingstarted/select?indent=on&q=*:*&wt=json"
```

Basics

Search for a single term

To search for a term, give it as the `q` param value in the core-specific Solr Admin UI Query section, replace `*:*` with the term you want to find. To search for "foundation":

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=foundation"
```

You'll see:

```
/solr-6.5.1$ curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=foundation"
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":527,
    "params":{
      "q":"foundation",
      "indent":"true",
      "wt":"json"}},
  "response":{"numFound":4156,"start":0,"maxScore":0.10203234,"docs":[
      {
        "id":"0553293354",
        "cat":["book"],
        "name":["Foundation"],
...
```

The response indicates that there are 4,156 hits (`"numFound":4156`), of which the first 10 were returned, since by default `start=0` and `rows=10`. You can specify these params to page through results, where `start` is the (zero-based) position of the first result to return, and `rows` is the page size.

To restrict fields returned in the response, use the `fl` param, which takes a comma-separated list of field names. E.g. to only return the `id` field:

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=foundation&fl=id"
```

`q=foundation` matches nearly all of the docs we've indexed, since most of the files under `docs/` contain "The Apache Software Foundation". To restrict search to a particular field, use the syntax "`q=field:value`", e.g. to search for `Foundation` only in the `name` field:

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=name:Foundation"
```

The above request returns only one document (`"numFound":1`) - from the response:

```
...
  "response":{"numFound":1,"start":0,"maxScore":2.5902672,"docs":[
      {
        "id":"0553293354",
        "cat":["book"],
        "name":["Foundation"],
...
```

Phrase search

To search for a multi-term phrase, enclose it in double quotes: `q="multiple terms here"`. E.g. to search for "CAS latency" - note that the space between terms must be converted to "+" in a URL (the Admin UI will handle URL encoding for you automatically):

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=\"CAS+latency\""
```

You'll get back:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":391,
    "params":{
      "q":"\"CAS latency\"",
      "indent":"true",
      "wt":"json"}},
  "response":{"numFound":3,"start":0,"maxScore":22.027056,"docs":[
      {
        "id":"TWINX2048-3200PRO",
        "name":["CORSAIR  XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400
(PC 3200) Dual Channel Kit System Memory - Retail"],
        "manu":["Corsair Microsystems Inc."],
        "manu_id_s":"corsair",
        "cat":["electronics", "memory"],
        "features":["CAS latency 2,  2-3-3-6 timing, 2.75v, unbuffered, heat-
spreader"],
...
```

Combining searches

By default, when you search for multiple terms and/or phrases in a single query, Solr will only require that one of them is present in order for a document to match. Documents containing more terms will be sorted higher in the results list.

You can require that a term or phrase is present by prefixing it with a "+"; conversely, to disallow the presence of a term or phrase, prefix it with a "-".

To find documents that contain both terms "`one`" and "`three`", enter `+one +three` in the `q` param in the Admin UI Query tab. Because the "+" character has a reserved purpose in URLs (encoding the space character), you must URL encode it for `curl` as "`%2B`":

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=%2Bone+%2Bthree"
```

To search for documents that contain the term "`two`" but **don't** contain the term "`one`", enter `+two -one` in the `q` param in the Admin UI. Again, URL encode "+" as "`%2B`":

```
curl "http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=%2Btwo+-one"
```

In depth

For more Solr search options, see the Solr Reference Guide's Searching section.

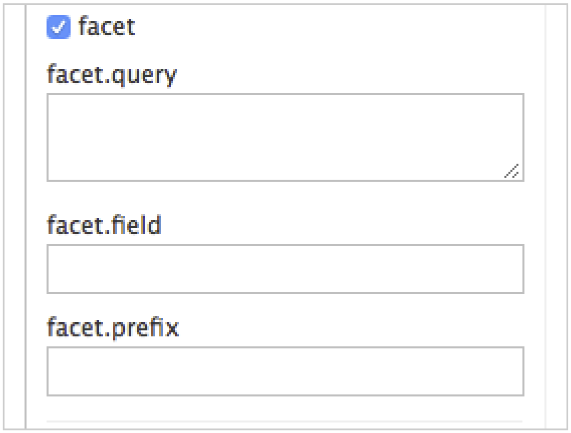
Faceting

One of Solr's most popular features is faceting. Faceting allows the search results to be arranged into subsets (or buckets or categories), providing a count for each subset. There are several types of faceting: field values, numeric and date ranges, pivots (decision tree), and arbitrary query faceting.

Field facets

In addition to providing search results, a Solr query can return the number of documents that contain each unique value in the whole result set.

From the core-specific Admin UI Query tab, if you check the "`facet`" checkbox, you'll see a few facet-related options appear:



To see facet counts from all documents (`q=*:*`): turn on faceting (`facet=true`), and specify the field to facet on via the `facet.field` param. If you only want facets, and no document contents, specify `rows=0`. The `curl` command below will return facet counts for the `manu_id_s` field:

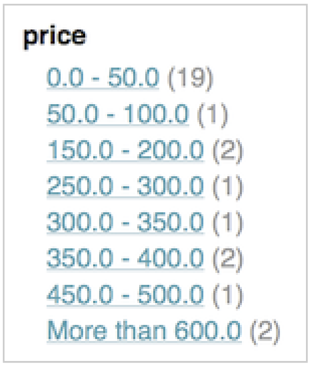
```
curl 'http://localhost:8983/solr/gettingstarted/select?
wt=json&indent=true&q=*:*&rows=0'\
'&facet=true&facet.field=manu_id_s'
```

In your terminal, you'll see:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":201,
    "params":{
      "q":"*:*",
      "facet.field":"manu_id_s",
      "indent":"true",
      "rows":"0",
      "wt":"json",
      "facet":"true"}},
  "response":{"numFound":4374,"start":0,"maxScore":1.0,"docs":[]
  },
  "facet_counts":{
    "facet_queries":{},
    "facet_fields":{
      "manu_id_s":[
        "corsair",3,
        "belkin",2,
        "canon",2,
        "apple",1,
        "asus",1,
        "ati",1,
        "boa",1,
        "dell",1,
        "eu",1,
        "maxtor",1,
        "nor",1,
        "uk",1,
        "viewsonic",1,
        "samsung",0]},
    "facet_ranges":{},
    "facet_intervals":{},
    "facet_heatmaps":{}}}
```

Range facets

For numerics or dates, it's often desirable to partition the facet counts into ranges rather than discrete values. A prime example of numeric range faceting, using the example product data, is `price`. In the `/browse` UI, it looks like this:



price
- 0.0 - 50.0 (19)
- 50.0 - 100.0 (1)
- 150.0 - 200.0 (2)
- 250.0 - 300.0 (1)
- 300.0 - 350.0 (1)
- 350.0 - 400.0 (2)
- 450.0 - 500.0 (1)
- More than 600.0 (2)

The data for these price range facets can be seen in JSON format with this command:

```
curl 'http://localhost:8983/solr/gettingstarted/select?
q=*:*&wt=json&indent=on&rows=0'\
'&facet=true'\
'&facet.range=price'\
'&f.price.facet.range.start=0'\
'&f.price.facet.range.end=600'\
'&f.price.facet.range.gap=50'\
'&facet.range.other=after'
```

In your terminal you will see:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":248,
    "params":{
      "facet.range":"price",
      "q":"*:*",
      "f.price.facet.range.start":"0",
      "facet.range.other":"after",
      "indent":"on",
      "f.price.facet.range.gap":"50",
      "rows":"0",
      "wt":"json",
      "facet":"true",
      "f.price.facet.range.end":"600"}},
  "response":{"numFound":4374,"start":0,"maxScore":1.0,"docs":[]
  },
  "facet_counts":{
    "facet_queries":{},
    "facet_fields":{},
    "facet_ranges":{
      "price":{
        "counts":[
          "0.0",19,
          "50.0",1,
          "100.0",0,
          "150.0",2,
          "200.0",0,
          "250.0",1,
          "300.0",1,
          "350.0",2,
          "400.0",0,
          "450.0",1,
          "500.0",0,
          "550.0",0],
        "gap":50.0,
        "after":2,
        "start":0.0,
        "end":600.0}},
    "facet_intervals":{},
    "facet_heatmaps":{}}}
```

Pivot facets

Another faceting type is pivot facets, also known as "decision trees", allowing two or more fields to be nested for all the various possible combinations. Using the example technical product data, pivot facets can be used to see how many of the products in the "book" category (the `cat`field) are in stock or not in stock. Here's how to get at the raw data for this scenario:

```
curl 'http://localhost:8983/solr/gettingstarted/select?
q=*:*&rows=0&wt=json&indent=on'\
'&facet=on&facet.pivot=cat,inStock'
```

This results in the following response (trimmed to just the book category output), which says out of 14 items in the "book" category, 12 are in stock and 2 are not in stock:

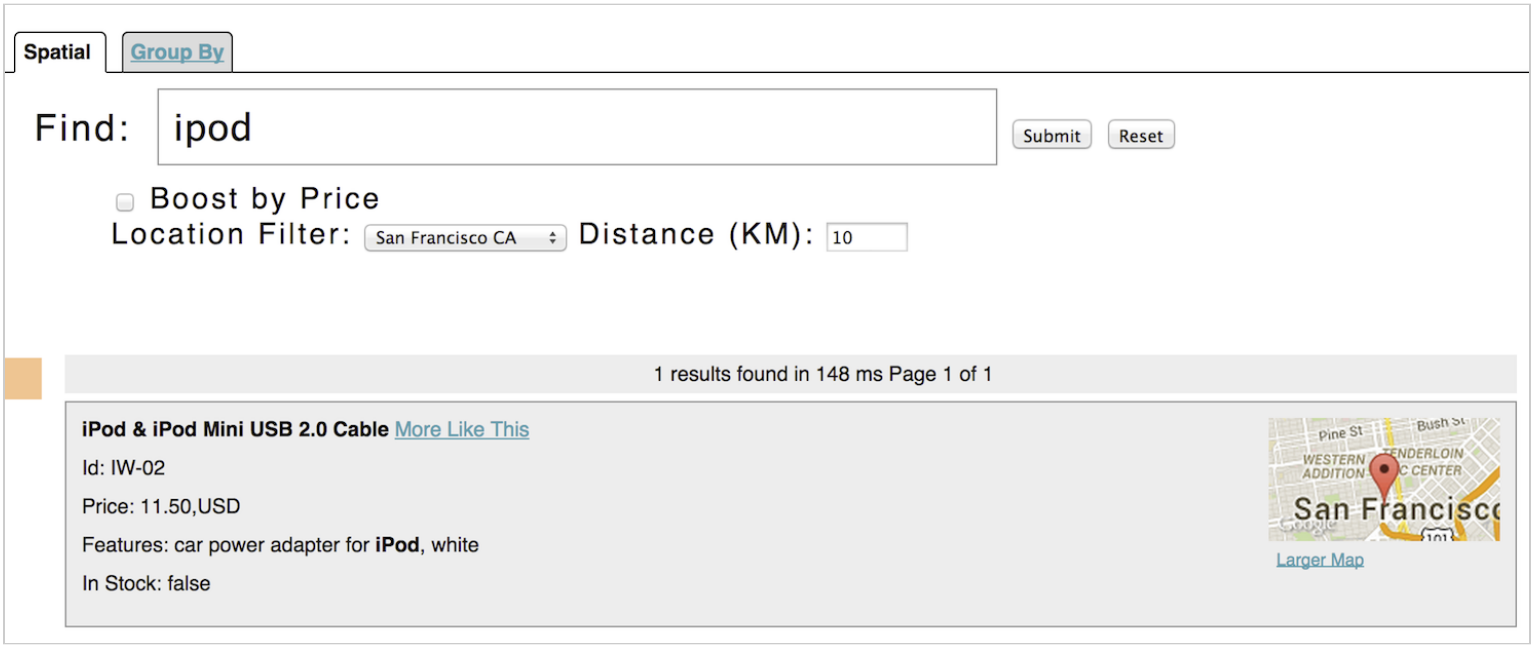
```
...
"facet_pivot":{
  "cat,inStock":[{
      "field":"cat",
      "value":"book",
      "count":14,
      "pivot":[{
          "field":"inStock",
          "value":true,
          "count":12},
        {
          "field":"inStock",
          "value":false,
          "count":2}]},
...
```

More faceting options

For the full scoop on Solr faceting, visit the Solr Reference Guide's Faceting section.

Spatial

Solr has sophisticated geospatial support, including searching within a specified distance range of a given location (or within a bounding box), sorting by distance, or even boosting results by the distance. Some of the example tech products documents in `example/exampledocs/*.xml` have locations associated with them to illustrate the spatial capabilities. To run the tech products example, see the techproducts example section. Spatial queries can be combined with any other types of queries, such as in this example of querying for "ipod" within 10 kilometers from San Francisco:



1 results found in 148 ms Page 1 of 1

iPod & iPod Mini USB 2.0 Cable

Id: IW-02

Price: 11.50,USD

Features: car power adapter for **iPod**, white

In Stock: false

The URL to this example is http://localhost:8983/solr/techproducts/browse?q=ipod&pt=37.7752%2C-122.4232&d=10&sfield=store&fq=%7B%21bbox%7D&queryOpts=spatial&queryOpts=spatial, leveraging the `/browse` UI to show a map for each item and allow easy selection of the location to search near.

To learn more about Solr's spatial capabilities, see the Solr Reference Guide's Spatial Search section.

Wrapping up

If you've run the full set of commands in this quick start guide you have done the following:

- Launched Solr into SolrCloud mode, two nodes, two collections including shards and replicas
- Indexed a directory of rich text files
- Indexed Solr XML files
- Indexed Solr JSON files
- Indexed CSV content
- Opened the admin console, used its query interface to get JSON formatted results
- Opened the /browse interface to explore Solr's features in a more friendly and familiar interface

Nice work! The script (see below) to run all of these items took under two minutes! (Your run time may vary, depending on your computer's power and resources available.)

Here's a Unix script for convenient copying and pasting in order to run the key commands for this quick start guide:

```
date
bin/solr start -e cloud -noprompt
  open http://localhost:8983/solr
  bin/post -c gettingstarted docs/
  open http://localhost:8983/solr/gettingstarted/browse
  bin/post -c gettingstarted example/exampledocs/*.xml
  bin/post -c gettingstarted example/exampledocs/books.json
  bin/post -c gettingstarted example/exampledocs/books.csv
  bin/post -c gettingstarted -d "<delete><id>SP2514N</id></delete>"
  bin/solr healthcheck -c gettingstarted
date
```

Cleanup

As you work through this guide, you may want to stop Solr and reset the environment back to the starting point. The following command line will stop Solr and remove the directories for each of the two nodes that the start script created:

```
bin/solr stop -all ; rm -Rf example/cloud/
```

Where to next?

For more information on Solr, check out the following resources:

- Solr Reference Guide (ensure you match the version of the reference guide with your version of Solr)
- See also additional Resources