**Getting started with SOLR**

**Installing SOLR:**

Prerequisite: You should also have JDK 6 or above installed.

1. Installing On Windows:

Execute following steps to Install SOLR on windows:

* 1. Download the SOLR distribution
  2. Go to <http://www.apache.org/dyn/closer.lua/lucene/solr/5.5.0>
  3. Select a mirror and download the zip file.

e.g. <http://a.mbbsindia.com/lucene/solr/5.5.0/solr-5.5.0.zip>

* 1. Extract the zip file in ‘C’ drive, e.g. ‘C:\solr-5.5.0’

1. Installing On Linux:

Execute following steps to Install SOLR on Linux:

* 1. Download the SOLR distribution, execute below command:

|  |
| --- |
| *wget*[*http://download.nextag.com/apache/lucene/solr/5.3.0/solr-5.3.0.tgz*](http://download.nextag.com/apache/lucene/solr/5.3.0/solr-5.3.0.tgz) |

* 1. After download completion, extract the zip to ‘/local/’. Execute below command:

|  |
| --- |
| *tar -zxvf solr-5.3.0.tgz* |

**Starting SOLR:**

You can use below given commands to start SOLR:

|  |
| --- |
| *solr start*  *or*  *solr start –all*  *or*  *solr start -p <port-number> e.g. solr start -p 8983* |

You can execute following command to get usage details of solr start:

|  |
| --- |
| *solr start -help* |

**Verifying SOLR Status:**

Execute below given command to verify the status of SOLR:

|  |
| --- |
| *solr status* |

It will display following details on command prompt:

|  |
| --- |
| *Found Solr process 32928 running on port 8983*  *{*  *"solr\_home":"C:\\solr-5.5.0\\server\\solr",*  *"version":"5.5.0 2a228b3920a07f930f7afb6a42d0d20e184a943c - mike - 2016-02-16*  *15:22:52",*  *"startTime":"2016-03-18T14:23:46.737Z",*  *"uptime":"0 days, 0 hours, 0 minutes, 4 seconds",*  *"memory":"57.5 MB (%11.7) of 490.7 MB"}* |

**Running Examples:**

For simplicity let’s go with OOTB examples. Solr comes with an example directory which contains some sample files we can use.

The available examples are:

* cloud: SolrCloud example
* dih: Data Import Handler (rdbms, mail, rss, tika)
* schemaless : Schema-less example (schema is inferred from data during indexing)
* techproducts : Kitchen sink example providing comprehensive examples of Solr features

Execute the following steps to run an example on windows:

* 1. Open command prompt and go to ‘C:\solr-5.5.0\bin’ directory
  2. Use the following command to run an example:

|  |
| --- |
| *solr -e [example]* |

Here [example] is one of the above given examples. Let’s run ‘techproducts’.

|  |
| --- |
| *C:\solr-5.5.0\bin>solr -e techproducts* |

You should see solr will execute below given steps in the terminal.

|  |
| --- |
| *Creating Solr home directory C:\solr-5.5.0\example\techproducts\solr*  *Starting up Solr on port 8983 using command:*  *C:\solr-5.5.0\bin\solr.cmd start -p 8983 -s "C:\solr-5.5.0\example\techproducts\*  *solr"*  *Waiting up to 30 to see Solr running on port 8983*  *Started Solr server on port 8983. Happy searching!*  *Copying configuration to new core instance directory:*  *C:\solr-5.5.0\example\techproducts\solr\techproducts*  *Creating new core 'techproducts' using command:*  [*http://localhost:8983/solr/admin/cores?action=CREATE&name=techproducts&instanceDir=techproducts*](http://localhost:8983/solr/admin/cores?action=CREATE&name=techproducts&instanceDir=techproducts)  *{*  *"responseHeader":{*  *"status":0,*  *"QTime":3899},*  *"core":"techproducts"*  *}*  *Indexing tech product example docs from C:\solr-5.5.0\example\exampledocs*  *SimplePostTool version 5.0.0*  *Posting files to [base] url* [*http://localhost:8983/solr/techproducts/update*](http://localhost:8983/solr/techproducts/update) *using*  *content-type application/xml...*  *POSTing file gb18030-example.xml to [base]*  *POSTing file hd.xml to [base]*  *POSTing file ipod\_other.xml to [base]*  *POSTing file ipod\_video.xml to [base]*  *POSTing file manufacturers.xml to [base]*  *POSTing file mem.xml to [base]*  *POSTing file money.xml to [base]*  *POSTing file monitor.xml to [base]*  *POSTing file monitor2.xml to [base]*  *POSTing file mp500.xml to [base]*  *POSTing file sd500.xml to [base]*  *POSTing file solr.xml to [base]*  *POSTing file utf8-example.xml to [base]*  *POSTing file vidcard.xml to [base]*  *14 files indexed.*  *COMMITting Solr index changes to* [*http://localhost:8983/solr/techproducts/update*](http://localhost:8983/solr/techproducts/update)  *Time spent: 0:00:01.410*  *Solr techproducts example launched successfully. Direct your Web browser to*  [*http://localhost:8983/solr*](http://localhost:8983/solr) *to visit the Solr Admin UI* |

**Searchingdocuments in techproducts example:**

Let's try to retrieve the document we just added as part of ‘techproducts’. Solr accepts HTTP requests, you can use your web browser to communicate with Solr:

* [http://localhost:8983/solr/techproducts/select?q= currency&wt=json](http://localhost:8983/solr/techproducts/select?q=%20currency&wt=json)
* <http://localhost:8983/solr/techproducts/select?q=currency&wt=xml>

Here 'q' is query parameter and value is ‘currency’ (some keywordfrom money.xml) and 'wt' response format and value is 'json'. Alternatively we can also pass wt=xml in order to get xml response.

It returns following xml result (wt=xml is passed to get xml response:

|  |
| --- |
| *<response>*  *<lst name="responseHeader">*  *<int name="status">0</int>*  *<int name="QTime">2</int>*  *<lst name="params">*  *<str name="q">currency</str>*  *<str name="wt">xml</str>*  *</lst>*  *</lst><result name="response" numFound="4" start="0">*  *<doc>*  *<str name="id">USD</str>*  *<str name="name">One Dollar</str>*  *<str name="manu">Bank of America</str>*  *<str name="manu\_id\_s">boa</str>*  *<arr name="cat">*  *<str>currency</str>*  *</arr>*  *<arr name="features">*  *<str>Coins and notes</str>*  *</arr>*  *<str name="price\_c">1,USD</str>*  *<bool name="inStock">true</bool>*  *<long name="\_version\_">1528426460326395904</long>*  *</doc>*  *<doc>*  *<str name="id">EUR</str>*  *<str name="name">One Euro</str>*  *<str name="manu">European Union</str>*  *<str name="manu\_id\_s">eu</str>*  *<arr name="cat">*  *<str>currency</str>*  *</arr>*  *<arr name="features">*  *<str>Coins and notes</str>*  *</arr>*  *<str name="price\_c">1,EUR</str>*  *<bool name="inStock">true</bool>*  *<long name="\_version\_">1528426460333735936</long>*  *</doc>*  *<doc>*  *<str name="id">GBP</str>*  *<str name="name">One British Pound</str>*  *<str name="manu">U.K.</str>*  *<str name="manu\_id\_s">uk</str>*  *<arr name="cat">*  *<str>currency</str>*  *</arr>*  *<arr name="features">*  *<str>Coins and notes</str>*  *</arr>*  *<str name="price\_c">1,GBP</str>*  *<bool name="inStock">true</bool>*  *<long name="\_version\_">1528426460334784512</long>*  *</doc>*  *<doc>*  *<str name="id">NOK</str>*  *<str name="name">One Krone</str>*  *<str name="manu">Bank of Norway</str>*  *<str name="manu\_id\_s">nor</str>*  *<arr name="cat">*  *<str>currency</str>*  *</arr>*  *<arr name="features">*  *<str>Coins and notes</str>*  *</arr>*  *<str name="price\_c">1,NOK</str>*  *<bool name="inStock">true</bool>*  *<long name="\_version\_">1528426460335833088</long>*  *</doc>*  *</result></response>* |

Now let’s try to search for all documents having ‘***inStock=true***’ and get the ***name,id*** *and* ***manu*** in result.

<http://localhost:8983/solr/techproducts/select?q=inStock:true&wt=xml&fl=name,id,manu>

It will return following result:

|  |
| --- |
| *<response>*  *<lst name="responseHeader">*  *<int name="status">0</int>*  *<int name="QTime">1</int>*  *<lst name="params">*  *<str name="fl">name,id,manu</str>*  *<str name="q">inStock:true</str>*  *<str name="wt">xml</str>*  *</lst>*  *</lst>*  *<result name="response" numFound="17" start="0">*  *<doc>*  *<str name="id">GB18030TEST</str>*  *<str name="name">Test with some GB18030 encoded characters</str>*  *</doc>*  *<doc>*  *<str name="id">SP2514N</str>*  *<str name="name">*  *Samsung SpinPoint P120 SP2514N - hard drive - 250 GB - ATA-133*  *</str>*  *<str name="manu">Samsung Electronics Co. Ltd.</str>*  *</doc>*  *<doc>*  *<str name="id">6H500F0</str>*  *<str name="name">*  *Maxtor DiamondMax 11 - hard drive - 500 GB - SATA-300*  *</str>*  *<str name="manu">Maxtor Corp.</str>*  *</doc>*  *<doc>*  *<str name="id">MA147LL/A</str>*  *<str name="name">Apple 60 GB iPod with Video Playback Black</str>*  *<str name="manu">Apple Computer Inc.</str>*  *</doc>*  *<doc>*  *<str name="id">TWINX2048-3200PRO</str>*  *<str name="name">*  *CORSAIR XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail*  *</str>*  *<str name="manu">Corsair Microsystems Inc.</str>*  *</doc>*  *<doc>*  *<str name="id">VS1GB400C3</str>*  *<str name="name">*  *CORSAIR ValueSelect 1GB 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) System Memory - Retail*  *</str>*  *<str name="manu">Corsair Microsystems Inc.</str>*  *</doc>*  *<doc>*  *<str name="id">VDBDB1A16</str>*  *<str name="name">*  *A-DATA V-Series 1GB 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) System Memory - OEM*  *</str>*  *<str name="manu">A-DATA Technology Inc.</str>*  *</doc>*  *<doc>*  *<str name="id">3007WFP</str>*  *<str name="name">Dell Widescreen UltraSharp 3007WFP</str>*  *<str name="manu">Dell, Inc.</str>*  *</doc>*  *<doc>*  *<str name="id">VA902B</str>*  *<str name="name">ViewSonic VA902B - flat panel display - TFT - 19"</str>*  *<str name="manu">ViewSonic Corp.</str>*  *</doc>*  *<doc>*  *<str name="id">0579B002</str>*  *<str name="name">Canon PIXMA MP500 All-In-One Photo Printer</str>*  *<str name="manu">Canon Inc.</str>*  *</doc>*  *</result></response>* |

**Shutting down SOLR:**

To shutdown Solr,go to ***‘<solr-installation-directory>/bin’***directory and use below given command. This will shut down Solr. You can specify the specific solr instance port if want to shutdown specific instance, else use ‘-all’ to shut down all instances.

|  |
| --- |
| *solr stop -all*  *or*  *solr stop -p <port-number>e.g. solr stop -p 8983* |

**Setting SOLR HOME:**

You can set solr home (solr.sole.home) directory while starting the solr. To know what is current location of ”***solr\_home***”, start solr using “***solr start***” command and execute “***solr status***”, it will display “***solr\_home***” path on console. By default it will be ***‘<solr-installation-directory>/ server/solr’****.*

To set the new “***solr\_home”*** directory execute following command while starting solr.

|  |
| --- |
| *solr start -s C:\mycores*  *or*  *solr start -s C:\mycores -p 8983 (If running the instance on specific port)* |

This will set the “***solr\_home”*** to ‘***C:\mycores’***. Note that before you set up your solr home directory you should create “***solr.xml***” inside the directory which you want to make ***solr\_home***. Contents of the ***solr.xml*** should contain at least below given configuration.

|  |
| --- |
| *<solrcloud>*  *<str name="host">${host:}</str>*  *<str name="hostContext">${hostContext:solr}</str>*  *<int name="hostPort">${jetty.port:8984}</int>*  *</solrcloud>* |

**Creating Custom Cores:**

**You can create your own ‘cores’ using following 4 approaches:**

1. Create *schema.xml, solrconfig.xml* and *core.properties* under *‘<solr-installation-directory>/ /server/solr’*directory.

Core will be loaded automatically when solr server will start. In this case it is mandatory to create configurations inside ***<solr-installation-directory>/server/solr***directory as given above.

* Go to *‘****<****solr-installation-directory>/server/solr*’directory.It default ***solr\_home***directory where all solr instances and cores will be created by default.You will also notice that there is a ‘***solr.xml***’ file which comes with OOTB solr. This ‘***solr.xml***’ file is for configuring one or more Solr Cores, as well as allowing Cores to be added, removed, and reloaded via HTTP requests. ‘***solr.xml***’ contains a***‘<solrcloud/>***’ configuration. When SOLR finds this tag, it searches for core.properties in all its subfolders and loads them. You can also create your own configuration in ‘***solr.xml***’; we will create our own solr.xml later.For more info see:<http://wiki.apache.org/solr/CoreAdmin>
* Create a directory e.g. books\_search. The name of this directory and name of ‘core’ should be same. In other words create the directory with same name which you want to use to create solr ‘*core*’.
* Go to newly created directory*‘books\_search’*, e.g. *‘****<****solr-installation-directory>/server/solr/books\_search’ or ‘C:\solr-5.5.0\server\solr\books\_search’.*
* Create a directory with name *‘data’*. This directory will be used by solr to maintain indexes for the data created under your core.
* Create a directory with name ***‘conf’***. Here we will have core specific configurations such as***‘schema.xml’***and ***‘solrconfig.xml’****.*
* Go to newly created directory ***‘conf’***andcreate ***‘solrconfig.xml’***. Visit [*http://wiki.apache.org/solr/SolrConfigXml*](http://wiki.apache.org/solr/SolrConfigXml)for details on configurations. For now we will keep below given configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample config -->*  *<!-- For more details about configurations options that may appear in*  *this file, see* [*http://wiki.apache.org/solr/SolrConfigXml*](http://wiki.apache.org/solr/SolrConfigXml)*-->*  *<config>*  *<luceneMatchVersion>5.5.0</luceneMatchVersion>*  *<!-- 'dataDir' parameter is used to specify an alternate directory to hold all index data other than the default './data' under the Solr home. If replication is in use, this should match the replication configuration.If this directory is not absolute, then it is relative to the directory you're in when you start SOLR.*  *If you do not specify dataDir then 'data' folder will be automatically created under your core parallel to 'conf' folder.-->*  *<!--Please note that this data directory should already exist, otherwise solr will throw error-->*  *<!--<dataDir>/var/data/solr/books\_search</dataDir>-->*  *<!--<dataDir>C:\mycores\solr\books\_search\data</dataDir> -->*  *<!-- We need to register query handler in order to query data.*  *Here 'standard' request handler is a query handler and implicitly mapped on path '/select' -->*  *<requestHandler name="standard" class="solr.StandardRequestHandler" />*  *<admin>*  *<!--Default query is set to ‘all ’-->*  *<defaultQuery>\*:\*</defaultQuery>*  *</admin>*  *</config>* |

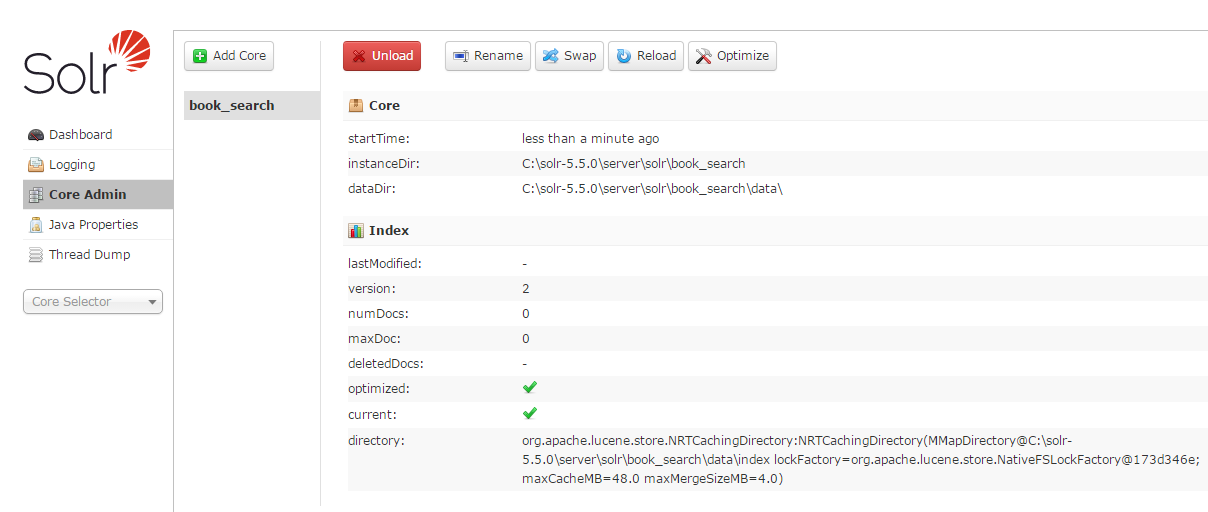
* Go to created directory ***‘conf’*** and create ***‘schema.xml’***.For details on field configuration visit: <http://wiki.apache.org/solr/SchemaXml>,for now we will keep following configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample schema -->*  *<schema name='books\_search' version='1.6'>*  *<!--The <types> section allows you to define a list of <fieldtype> declarations you wish to use in your schema, along with the underlying Solr class that should be used for that type, as well as the default options you want for fields that use that type.Any subclass of FieldType may be used as a field type class, using either its full package name, or the "solr" alias if it is in the default Solr package. For common numeric types (integer, float, etc...) there are multiple implementations provided depending on your needs, please see SolrPlugins for information on how to ensure that your own custom Field Types can be loaded into Solr.-->*  *<!--For details on types configuration visit:* [*http://wiki.apache.org/solr/SchemaXml#Types*](http://wiki.apache.org/solr/SchemaXml#Types) *-->*  *<types><fieldtype name="string" class="solr.StrField" />*  *<fieldtype name="long" class="solr.TrieLongField" />*  *</types>*  *<!--For details on field configuration visit:* [*http://wiki.apache.org/solr/SchemaXml#Fields*](http://wiki.apache.org/solr/SchemaXml#Fields)*-->*  *<fields>*  *<!-- Note: We have defined 2 types (long and string) of fieldTypes above under types elements.so we have to use only these two type of data types in field declaration. (As given below)-->*  *<!-- indexed=true makes a field searchable (and sortable and facetable).*  *For e.g., if you have a field named test1 with indexed=true, then you can search it like q=test1:foo, where foo is the value you are searching for.*  *If indexed=false for field test1 then that query will return no results, even if you have a document in Solr with test1's value being foo. -->*  *<!-- stored=true means you can retrieve the field when you search. If you want to explicitly retrieve the value of a field in your query, you will use the fl param in your query like fl=test1 (Default is fl=\* meaning retrieve all stored fields). Only if stored=true for test1, the value will be returned. Else it will not be returned. -->*  *<field name="id" type="long" indexed="true" stored="true" required="true" />*  *<field name="title" type="string" indexed="true" stored="true" required="true" />*  *<field name="author" type="string" indexed="true" stored="true" required="false"/>*    *<!-- One of the powerful features of Lucene is that you don't have to pre-define every field when you first create your index. Even though Solr provides strong datatyping for fields, it still preserves that flexibility using "Dynamic Fields". Using <dynamicField> declarations, you can create field rules that Solr will use to understand what datatypeshould be used whenever it is given a field name that is not explicitly defined, but matches a prefix or suffix used in a dynamicField.-->*  *<dynamicField name="\*\_str" type="string" indexed="true" stored="true"multiValued="true"/>*    *<!-- Since we are going do fullText search we need field value to be set as multiValued='true', otherwise solr will throw error while loading data-->*  *<field name="fullText" type="string" indexed="true" multiValued="true"/></fields>*  *<!--It is used when content of a 'source' field needs to be added/INDEXED on some other destination field. The idea behind is that we want to be able to search in all the fields of the document at the same time.A common requirement is to copy or merge all input fields into a single solr field. That's why we have used '\*' here. It can be declared within <fields> elements as well as outside the<fields> element -->*    *<copyField source="\*" dest="fullText"/>*  *<!--The <uniqueKey> declaration can be used to inform Solr that there is a field in your index which should be unique for all documents.If a document is added that contains the same value for this field as an existing document, the old document will be deleted.It is not mandatory for a schema to have a uniqueKey field, but an overwhelming majority of them do. It shouldn't matter whether you rename this to something else (and change the <uniqueKey> value), but occasionally it has in the past. We recommend that you just leave this definition alone. -->*  *<uniqueKey>id</uniqueKey>*  *<!-- Use LocalParams to build queries:* [*http://wiki.apache.org/solr/LocalParams*](http://wiki.apache.org/solr/LocalParams)*-->*  *</schema>* |

* Go back to *‘****<****solr-installation-directory>/server/solr/books\_search’ e.g. ‘C:\solr-5.5.0\server\solr\books\_search’* directory.
* Create ***‘core.properties’*** file under *‘books\_search’* directory and add following configurations. Please note that this file should contain at least *‘name’* property. Other properties will be taken default if not provided externally in core.properties. E.g. if config=mysolrconfig.xml (you can provide any name) is not provided SOLR will search for ‘solrconfig.xml’ (a default solr config file name) in <core>/conf directory.

|  |
| --- |
| #For more details on core.properties visit: #<https://cwiki.apache.org/confluence/display/solr/Defining+core.properties>  #Name of the core  *name=book\_search*  #SOLR Config xml name, It can be any name e.g. mysolrconfig.xml.  *config=mysolrconfig.xml*  #SOLR schema xml name, It can be any name e.g. myschema.xml  *schema=myschema.xml*  #Indicates SOLR that, load this core on startup.  *loadOnStartup=true*  *#Data Directory, we can provide data directory here as well.*  *#dataDir=C:\mycores\solr\books\_search\data* |

* Start the SOLR instance using *‘solr start -p 8983’* command. It should automatically load the core configurations and you can see the newly created core in Admin UI. Open <http://localhost:8983/solr/#/~cores/>. You will see below details on admin UI.



**Note:** You should either use this way to create cores or use below given approaches to create cores. Otherwise configurations as part of “solr.xml” will override the configurations given in “solr.xml” in below mentioned approaches.In other words **“solr.xml”** in*‘****<****solr-installation-directory>/server/solr’ e.g. ‘C:\solr-5.5.0\server\solr’*directory will take preference.

1. Create solr.xml, *schema.xml and solrconfig.xml* outside solr installation directory.

* Go to any preferred location on file system e.g. *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) directory.
* Set the ***solr\_home***to *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) and start SOLR. Use **‘*solr start -s C:\mycores\solr’*** command to set ***solr\_home***and start SOLR.
* Create **‘*solr.xml’*** *(as discussed above that we can create our own configuration)* with below given contents. This enables automatic discovery of solr cores within *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) directory.

|  |
| --- |
| *<?xml version="1.0" encoding="UTF-8" ?>*  *<!--*  *This is a sample of a simple "solr.xml" file for configuring one or*  *more Solr Cores, as well as allowing Cores to be added, removed, and*  *reloaded via HTTP requests. More information about options available in this configuration file, and Solr Core administration can be found online:* [*http://wiki.apache.org/solr/CoreAdmin*](http://wiki.apache.org/solr/CoreAdmin)  *-->*  *<!--This is sample config -->*  *<solr>*  *<!--When SOLR finds this tag, it searches for core.propertiesin all its subfolders and loads them-->*  *<!--You can create a core.properties file inside your core directory with appropriate configuration and SOLR will load that core automatically -->*  *<solrcloud/>*  *</solr>* |

* Create a directory e.g. search. The name of this directory and name of ‘core’ should be same. In other words create the directory with same name which you want to use to create solr ‘*core*’.
* Go to newly created directory *‘search’*. e.g. *‘/local/mycores/solr/search’* or *‘C:\mycores\solr\search’* (on windows)*.*
* Create a directory with name ***‘data’***. This directory will be used by solr to maintain indexes for the data created under your core.
* Create a directory with name ***‘conf’***. Here we will have core specific configurations such as *‘****schema.xml’***and *‘****solrconfig.xml****’.*
* Go to newly created directory *‘conf’* and create *‘****solrconfig.xml****’*. Visit [*http://wiki.apache.org/solr/SolrConfigXml*](http://wiki.apache.org/solr/SolrConfigXml)for details on configurations. For now we will keep below given configuration.

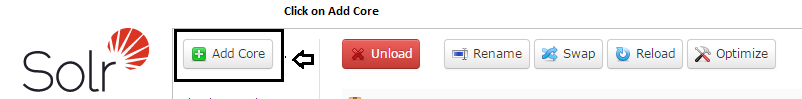
|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample config -->*  *<config>*  *<luceneMatchVersion>5.5.0</luceneMatchVersion>*  *<!—Please note that this data directory should already exist, otherwise solr will throw error-->*  *<!--<dataDir>/var/data/solr/search</dataDir> -->*  *<!-- <dataDir>C:\search\data\solr</dataDir> -->*  *<!-- We need to register query handler in order to query data.*  *Here 'standard' request handler is a query handler and implicitly mapped on path '/select' -->*  *<requestHandler name="standard" class="solr.StandardRequestHandler" />*  *<admin>*  *<defaultQuery>\*:\*</defaultQuery>*  *</admin>*  *</config>* |

* Go to created directory ***‘conf’*** and create ***‘schema.xml’***. For details on field configuration visit: <http://wiki.apache.org/solr/SchemaXml>

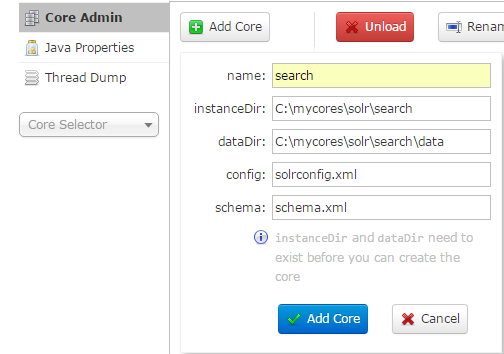
For now we will keep following configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample schema -->*  *<schema name='search' version='1.6'>*  *<types>*  *<fieldtype name="string" class="solr.StrField" />*  *<fieldtype name="long" class="solr.TrieLongField" />*  *</types>*  *<fields>*  *<field name="id" type="long" indexed="true" stored="true" required="true" />*  *<field name="title" type="string" indexed="true" stored="true" required="true" />*  *<field name="author" type="string" indexed="true" stored="true" required="false"/>*  *<dynamicField name="\*\_str" type="string" indexed="true" stored="true"multiValued="true"/>*  *<field name="fullText" type="string" indexed="true"multiValued="true"/>*  *</fields>*  *<copyField source="\*" dest="fullText"/>*  *<uniqueKey>id</uniqueKey></schema>* |

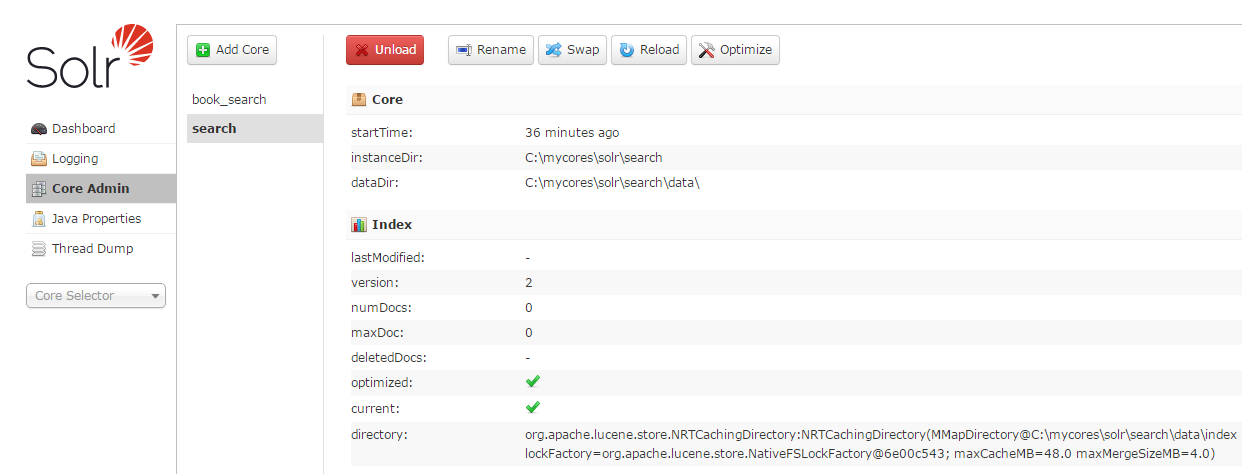
* Go to <http://localhost:8983/solr/#/~cores>
* Click on ‘**Add Core**’



* Provide core **‘name’**, **‘instanceDir’** (instance directory) and **‘dataDir’** (data directory) values and leave other fields as is. Here I am creating core with name ‘search’.



* Click on **“Add Core”**, it will create new core ‘search’. You can view the core details by visiting: <http://localhost:8983/solr/#/~cores/search>



1. Create solr.xml, *schema.xml and solrconfig.xml* outside solr installation directory, and use command line.

* Go to any preferred location on file system e.g. *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) directory.
* Set the ***solr\_home***to *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) and start SOLR. Use **‘*solr start -s C:\mycores\solr’*** command to set ***solr\_home***and start SOLR.
* Create **‘*solr.xml’***with below given contents.

|  |
| --- |
| *<?xml version="1.0" encoding="UTF-8" ?>*  *<!--This is sample config -->*  *<solr>*  *<!--When SOLR finds this tag, it searches for core.properties in all its subfolders and loads them-->*  *<!--You can create a core.properties file inside your core directory with appropriate configuration and SOLR will load that core automatically -->*  *<solrcloud/>*  *</solr>* |

* Create a directory e.g. example. The name of this directory and name of core should be same. In other words create the directory with same name which you want to use to create solr *core*.
* Go to newly created directory ***‘example’***. e.g. *‘/local/mycores/solr/example’* or *‘C:\mycores\solr\example’* (on windows)*.*
* Create a directory with name ***‘data’***. This directory will be used by solr to maintain indexes for the data created under your core.
* Create a directory with name ***‘conf’***. Here we will have core specific configurations such as***‘schema.xml’*** and ***‘solrconfig.xml’****.*
* Go to newly created directory ***‘conf’*** and create *‘****solrconfig.xml’***. Visit [*http://wiki.apache.org/solr/SolrConfigXml*](http://wiki.apache.org/solr/SolrConfigXml) for details on configurations.

For now we will keep below given configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample config -->*  *<config>*  *<luceneMatchVersion>5.5.0</luceneMatchVersion>*  *<dataDir>C:\mycores\solr\example\data</dataDir>*  *<!-- We need to register query handler in order to query data.*  *Here 'standard' request handler is a query handler and implicitly mapped on path '/select' -->*  *<requestHandler name="standard" class="solr.StandardRequestHandler" />*  *<admin>*  *<defaultQuery>\*:\*</defaultQuery>*  *</admin>*  *</config>* |

* Go to created directory ***‘conf’*** and create ***‘schema.xml’***. For details on field configuration visit: <http://wiki.apache.org/solr/SchemaXml>, for now we will keep following configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample schema -->*  *<schema name='example' version='1.6'>*  *<types>*  *<fieldtype name="string" class="solr.StrField" />*  *<fieldtype name="long" class="solr.TrieLongField" />*  *</types>*  *<fields>*  *<field name="id" type="long" indexed="true" stored="true" required="true" />*  *<field name="title" type="string" indexed="true" stored="true" required="true" />*  *<field name="author" type="string" indexed="true" stored="true" required="false"/>*  *<dynamicField name="\*\_str" type="string" indexed="true" stored="true"multiValued="true"/>*  *<field name="fullText" type="string" indexed="true"multiValued="true"/>*  *</fields>*  *<copyField source="\*" dest="fullText"/>*  *<uniqueKey>id</uniqueKey>*  *</schema>* |

* Open command prompt, go to ‘*<solr-installation-directory>/bin’*and type below given command.

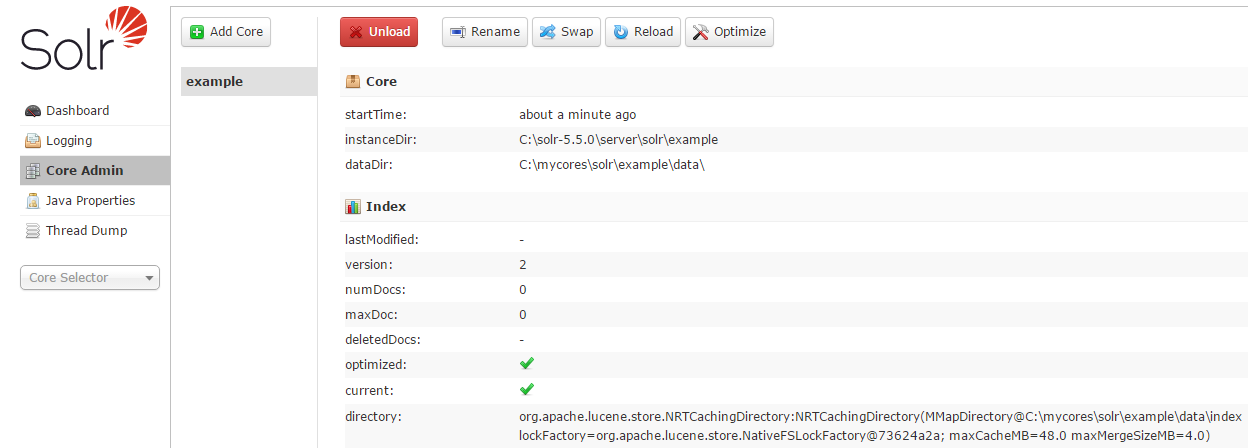
|  |
| --- |
| *solr create\_core [-c name] [-d confdir] [-p port]*  ***e.g.****solr create\_core-c example -d C:\mycores\solr\example -p 8983*  *or(for Linux)*  *solr create\_core -c example -d /local/mycores/solr/example -p 8983* |

The Solr create command has following options:

* -c <name> - Name of the core or collection to create (required).
* -d <confdir> - The configuration directory, useful in the SolrCloud mode.
* -n <configName> – The configuration name. This defaults to the same name as the core or collection.
* -p <port> – Port of a local Solr instance to send the create command to; by default the script tries to detect the port by looking for running Solr instances.
* -s <shards> – Number of shards to split a collection into, default is 1.
* -rf <replicas> – Number of copies of each document in the collection. The default is 1.
* You can see below response after executing below command:

|  |
| --- |
| *Copying configuration to new core instance directory:*  *C:\solr-5.5.0\server\solr\example*  *Creating new core 'example' using command:*  [*http://localhost:8983/solr/admin/cores?action=CREATE&name=example&instanceDir=example*](http://localhost:8983/solr/admin/cores?action=CREATE&name=example&instanceDir=example)  *{*  *"responseHeader":{*  *"status":0,*  *"QTime":218},*  *"core":"example"*  *}* |

* It will create new core ***‘example’***. You can view the core details by visiting: <http://localhost:8983/solr/#/~cores/example>



1. Create solr.xml, *schema.xml and solrconfig.xml* outside solr installation directory, and use http rest call.

* Set the ***solr\_home***to *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) and start SOLR. Use **‘*solr start -s C:\mycores\solr’*** command to set ***solr\_home***and start SOLR.
* Go to any preferred location on file system e.g. *‘/local/mycores/solr’* or *‘C:\mycores\solr’* (on windows) directory.
* Create **‘*solr.xml’***with below given contents.

|  |
| --- |
| *<?xml version="1.0" encoding="UTF-8" ?>*  *<!--This is sample config -->*  *<solr>*  *<!--When SOLR finds this tag, it searches for core.properties in all its subfolders and loads them-->*  *<!--You can create a core.properties file inside your core directory with appropriate configuration and SOLR will load that core automatically -->*  *<solrcloud/>*  *</solr>* |

* Create a directory e.g. example2. The name of this directory and name of core should be same. In other words create the directory with same name which you want to use to create solr *core*.
* Go to newly created directory ***‘example2’***. e.g. *‘/local/mycores/solr/example2’* or *‘C:\mycores\solr\example2’* (on windows)*.*
* Create a directory with name ***‘data’***. This directory will be used by solr to maintain indexes for the data created under your core.
* Create a directory with name ***‘conf’***. Here we will have core specific configurations such as***‘schema.xml’*** and ***‘solrconfig.xml’****.*
* Go to newly created directory ***‘conf’*** and create *‘****solrconfig.xml’***. Visit [*http://wiki.apache.org/solr/SolrConfigXml*](http://wiki.apache.org/solr/SolrConfigXml) for details on configurations. For now we will keep below given configuration.

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample config -->*  *<config>*  *<luceneMatchVersion>5.5.0</luceneMatchVersion>*  *<dataDir>C:\mycores\solr\example2\data</dataDir>*  *<!-- We need to register query handler in order to query data.*  *Here 'standard' request handler is a query handler and implicitly mapped on path '/select' -->*  *<requestHandler name="standard" class="solr.StandardRequestHandler" />*  *<admin>*  *<defaultQuery>\*:\*</defaultQuery>*  *</admin>*  *</config>* |

* Go to created directory ***‘conf’*** and create ***‘schema.xml’***. For now we will keep following configuration.For details on field configuration visit: <http://wiki.apache.org/solr/SchemaXml>

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample schema -->*  *<schema name='examples' version='1.6'>*  *<types>*  *<fieldtype name="string" class="solr.StrField" />*  *<fieldtype name="long" class="solr.TrieLongField" />*  *</types>*  *<fields>*  *<field name="id" type="long" indexed="true" stored="true" required="true" />*  *<field name="title" type="string" indexed="true" stored="true" required="true" />*  *<field name="author" type="string" indexed="true" stored="true" required="false"/>*  *<dynamicField name="\*\_str" type="string" indexed="true" stored="true"multiValued="true"/>*  *<field name="fullText" type="string" indexed="true"multiValued="true"/>*  *</fields>*  *<copyField source="\*" dest="fullText"/>*  *<uniqueKey>id</uniqueKey>*  *</schema>* |

* Open web browser and use below REST URL to create core.

|  |
| --- |
| <http://localhost:8983/solr/admin/cores?action=CREATE&name=example2&instanceDir=C:\mycores\solr\example2&dataDir=C:\mycores\solr\example2\data> |

* It will create new core with name ‘example2’ and will return below response.

|  |
| --- |
| *<response>*  *<lst name="responseHeader">*  *<int name="status">0</int>*  *<int name="QTime">162</int>*  *</lst>*  *<str name="core">example2</str>*  *</response>* |

* You can view the core details by visiting: <http://localhost:8983/solr/#/~cores/example2>

**Creating documents and indexes:**

In Solr, a Document is the unit of search and index.Solr can achieve fast search results because instead on searching on actual text it searches on Indexes.

This is similar to finding pages in a book related to a keyword by scanning the index/glossary at the back of a book, instead of searching every word of every page of the book.

This type of index is called an inverted index, because it inverts a page-centric data structure (***page->words***) to a keyword-centric data structure (***word->pages***).

Solr stores this index in a directory called **“index”** in the data directory.

**Schema:**

Before adding documents to Solr, you need to specify the schema, represented in a file called schema.xml. Please note that, we should not change the schema after documents are added to index. It is not advisable.

**The schema declares:**

* What type of fields there are
* Which field should be used as the unique/primary key
* Which fields are required
* How to index and search each field

**Field Types:**

In Solr, every field has a type. Examples of basic field types available in Solr include, float, long, double, date, text etc.

**Defining a field:**

A field in SOLR schema will look something like this:

***<field name="id" type="long" indexed="true" stored="true" multiValued="true"/>***

* **name**: Name of the field
* **type**: Field type
* **indexed**: Should this field be added to the inverted index?
* **stored**: Should the original value of this field be stored?
* **multiValued**: Can this field have multiple values?

The indexed and stored attributes are important because ***"indexed=true"*** makes a field searchable (and sortable and facetable).

For e.g., if i have a field named ***"title"*** with indexed=true, then you can search it like ***"q=title:Java",***where ***"Java"*** is the value you are searching for. If ***indexed=false*** for field ***"title"*** then that query will return no results, even if you have a document in Solr with title's value being foo.

Whereas***"stored=true"*** means you can retrieve the field when you search. If you want to explicitly retrieve the value of a field in your query, you will use the ***"fl"*** param in your query like ***"fl=title"*** (Default is ***fl=\**** meaning retrieve all stored fields). Only if ***"stored=true"*** for ***title***, the value will be returned. Else it will not be returned.

It is also advised that you should not store all the fields all the time, because storing fields increases the size of the index, and the larger the index, the slower the search. Larger index requires more disk IO to get to the same amount of data.

**What happens when you load documents to SOLR for indexing?**

When you add documents to SOLR, it goes through various transformations such as include lower-casing, removing word stems etc. before being added to the index.

This is known as analysis phase. After the analysis phase, a sequence of tokens will be generated which will be added to the index. Tokens are not the original text; these are keywords which you may use for search query. Indexed fields (fields which are configured to *indexed="true"* in schema.xml) are fields which will undergo an analysis phase, and are added to the index at the end.

Also note that, if a field is not indexed, it cannot be searched on.

Let’s say that I have created a new core called ‘***books***’, If you will open solr admin UI and open core configurations, you will notice that “numDocs”and “maxDoc” count is 0. It means there are no documents loaded. Now let’s try to load documents for creating indexes. We can load XML, JSON & CSV into SOLR, and solr will create indexes based on schema defined in “schema.xml” (Refer schema definitions explained above).

SOLR accepts XML, JSON and CSV documents in a specified format. It has certain schema/format which we need to follow to create XML, JSON and CSV for loading. So let’s see how XMLs, JSONs and CSVs will look like.

**Format of XML:**

We need to create input XML for SOLR in below format. Start element will be <add> always and inside <add> element you can have multiple <doc> elements. Also note that input XML should be aligned with the “***schema.xml***” used while creating core.

If you see the sample below and compare schema.xml used above while creating cores, you will notice that field names are same. Exactly same field names we have to use inside <doc> element.

|  |
| --- |
| *<add>*  *<doc>*  *<field name="id">101</field>*  *<field name="title">Java</field>*  *<field name="author">Balaguruswami</field>*  *<!--Example of dynamicField as per schema, anything prefixed with “\_str”-->*  *<field name="language\_str">English</field>*  *<field name="language\_str">French</field>*  *<field name="bookstore\_str">Hyderabad</field>*  *<field name="bookstore\_str">Bangalore</field>*  *<!--Example of copyField as per schema, anything given in fullText will be indexed-->*  *<field name="fullText">This is a java book.</field>*  *....*  *</doc>*  *<doc>*  *<field name="id">102</field>*  *<field name="title">C</field>*  *<field name="author">Yashwant Kanetkar</field>*  *<field name="language\_str">English</field>*  *<field name="language\_str">French</field>*  *<field name="bookstore\_str">Hyderabad</field>*  *<field name="bookstore\_str">Delhi</field>*  *<field name="fullText">This is a C book.</field>*  *...*  *</doc>*  *...more <doc> elements*  *</add>* |

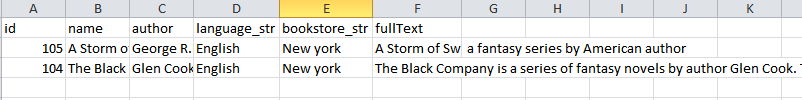
**Format of JSON:**

We need to create JSON in below given format.

|  |
| --- |
| *[*  *{*  *"id" : "103",*  *"title" : "The Lightning Thief",*  *"author" : "Rick Riordan",*  *"language\_str" :"English",*  *"language\_str" :"French",*  *“bookstore\_str" : "London",*  *"bookstore\_str" : "Paris"*  *}*  *,*  *{*  *"id" : "104",*  *"title" : "The Sea of Monsters",*  *"author" : "Rick Riordan",*  *"language\_str" :"English",*  *"language\_str" :"French",*  *"bookstore\_str" : "London",*  *"bookstore\_str" : "Paris"*  *}*  *…..more json data*  *]* |

**Format of CSV:**

Create a CSV file with column name as per the fields defined in schema.xml. See sample below.



**There are 3 basic ways to load documents into SOLR:**

1. **Using command line:**

We can load documents into SOLR using command line.

* Go to *<solr-installation-directory>/example/exampledocs* i.e. *C:\solr-5.5.0\example\exampledocs* (on windows).
* Find the “post.jar” file and copy to your data input directory e.g. “*C:\solr\_input\_data*”.

You can use following command to get usage details of “post.jar”. It is a simple java program which takes input data, parses it and uploads to SOLR using Http REST services.

|  |
| --- |
| *java -jar post.jar --help* |

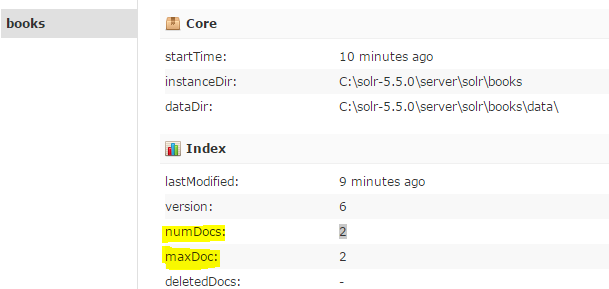
* Run below given command for XML:

|  |
| --- |
| *java -Dc=books -jar -Dtype=application/xml post.jar books.xml*  *or*  *java -Dc=books -jar -Dtype=application/xml post.jar \*.xml (To upload multiple xml files)* |

* You will see following response once data is loaded.

|  |
| --- |
| *SimplePostTool version 5.0.0*  *Posting files to [base] url* [*http://localhost:8983/solr/books/update*](http://localhost:8983/solr/books/update)*using content-type application/xml...*  *POSTing file books.xml to [base]*  *1 files indexed.*  *COMMITting Solr index changes to* [*http://localhost:8983/solr/books/update*](http://localhost:8983/solr/books/update)*...*  *Time spent: 0:00:00.669* |

* Now let’s see the status of core ‘books’ on solr admin UI.



* In the above screen shot you can see that ‘numDocs’ and ‘maxDoc’ count has been increased to 2, since we added two <doc> elements (Refer sample given above). Similarly you can upload all type for files such as JSON, CSV etc.
* Run below given command for JSON:

|  |
| --- |
| *java -Dc=books -jar -Dtype=application/jsonpost.jar books.json*  *or*  *java -Dc=books -jar -Dtype=application/jsonpost.jar \*.json (To upload multiple json docs)* |

* Run below given command for CSV:

|  |
| --- |
| *java -Dc=books -jar -Dtype=text/csv post.jar books.csv*  *or*  *java -Dc=books -jar -Dtype=text/csv post.jar \*.csv (To upload multiple csv docs)* |

* Now if you look at the core details on admin UI (as shown above), you will notice that the no. of documents is increased to 6. It is because we added 2 documents using xml, 2 using JSON and 2 using CSV.

1. **Using Http REST Client or CURL:**

We can load documents into SOLR using curl command or any RESTClient such as SOAPUI, POSTMAN etc.

* Open command prompt and run below curl commands to load documents to SOLR.

**Syntax:**

|  |
| --- |
| *curl -X POST "http://<host>:<port>/solr/<core-name>/update?commit=true" -H 'Content-Type:<MimeType-Of-Document> -d @<document-name>* |

**For XML:**

|  |
| --- |
| *curl -X POST "http://localhost:8983/solr/books/update?commit=true" -H 'Content-Type:application/xml' -d @books.xml* |

**For JSON:**

|  |
| --- |
| *curl -X POST "http://localhost:8983/solr/books/update?commit=true" -H 'Content-Type:application/json' -d @books.json* |

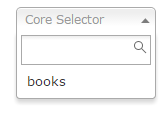
**For CSV:**

|  |
| --- |
| *curl -X POST "http://localhost:8983/solr/books/update?commit=true" -H 'Content-Type:text/csv' -d @books.csv* |

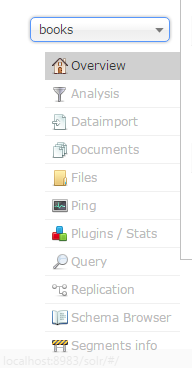
1. **Using solr admin UI:**

We can load documents into SOLR using admin UI.

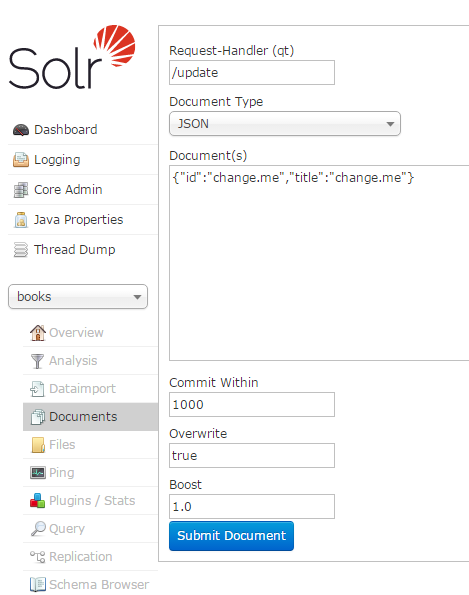
* Open admin UI (<http://localhost:8983/solr/#/>)
* Go to ‘Core Selector’ dropdown menu displayed as given below



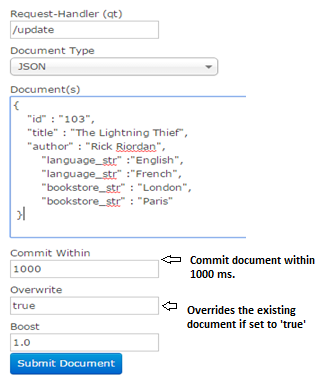
* Select the core where you want to upload documents. E.g. books
* Once you select core you will see following menu:



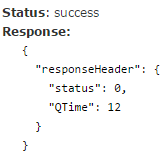
* Go to Documents menu. you will see following screen:



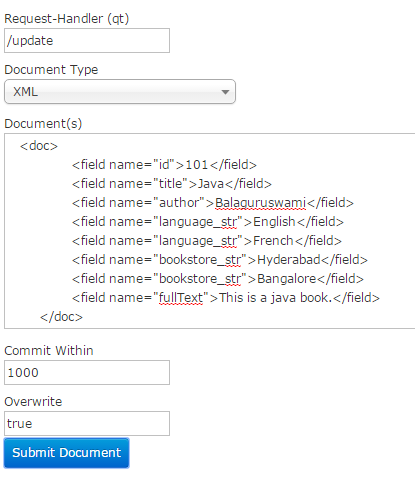
* You can see ‘/update’ request handler on screen, It will post the documents to SOLR in selected formats such as JSON, XML, CSV etc. (see Document Type dropdown). It is same as calling ‘*http://localhost:8983/solr/books/update*’ from http rest client. Let’s upload adocument in JSON format.
* Add a JSON document in “Documents” text box, selected “Document Type” as “JSON”, leave other field with default value and click on “Submit Document“.



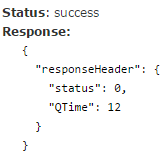
You will get following success response on the screen:



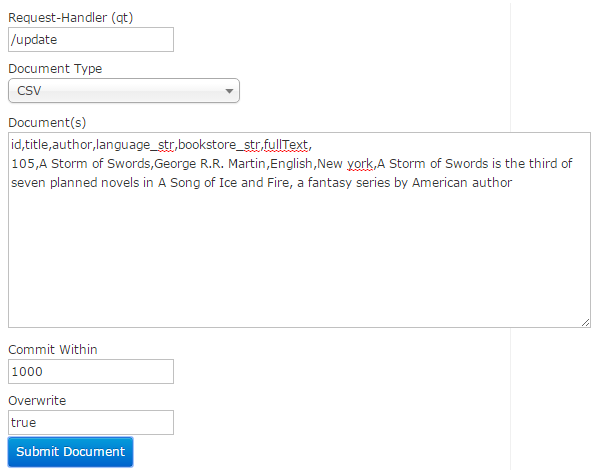
* Add XML documents in “Documents” text box, selected “Document Type” as “XML”, leave other field with default value and click on “Submit Document“.



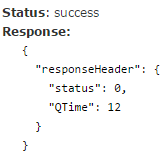
You will get following success response on the screen:



* Add CSV(comma separated text) document in “Documents” text box, selected “Document Type” as “CSV”, leave other field with default value and click on “Submit Document“.



You will get following success response on the screen:



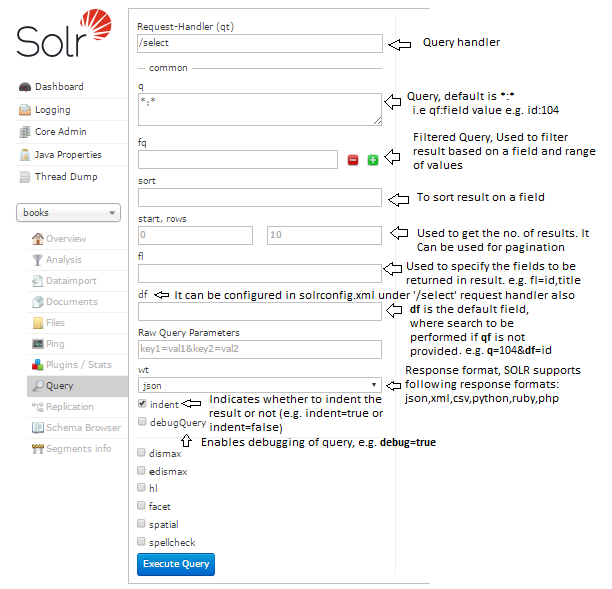
**There are many other ways to load documents in SOLR such as:**

1. Index binary documents such as Word and PDF with [**Solr Cell**](http://wiki.apache.org/solr/ExtractingRequestHandler) (ExtractingRequestHandler).Solr doesn't "end up store the PDF file" itself.However it *can* store the text contents of the PDF extracted from the PDF using a text-extractor such as Tika (if indeed the field is marked as stored in the schema). If you wish to store the PDF file in its entirety you will need to convert the PDF into (for example) Base64 representation and persist the base64 string as a "Stored" field. So when you access the doc you convert back from Base64 to PDF.
2. Use [**SolrJ**](http://wiki.apache.org/solr/Solrj) for Java or other Solr clients to programmatically create documents to send to Solr.

**Search:**

Let’s try to search the recently uploaded documents.

* Open SOLR admin UI; <http://localhost:8983/solr>
* Select the core (e.g. books) and go to query menu. You can directly navigate to query console using following URL: <http://localhost:8983/solr/#/books/query>



* Default search query (q=\*:\*) i.e. [http://localhost:8983/solr/books/select?q=\*:\*&wt=json](http://localhost:8983/solr/books/select?q=*:*&wt=json) will return all the documents available in solr index.
* Get all books which are available in “English” language and get only ‘id, author and title’ in result. Search query will look like:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:English&fl=id,title,author&wt=json&indent=true |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":1},  "response":{"numFound":7,"start":0,"docs":[  {  "id":104,  "title":"The Sea of Monsters",  "author":["Rick Riordan"]},  {  "id":103,  "title":"The Lightning Thief",  "author":["Rick Riordan"]},  {  "id":105,  "title":"A Storm of Swords",  "author":["George R.R. Martin"]},  {  "id":101,  "title":"Java",  "author":["Balaguruswami"]},  {  "id":102,  "title":"C",  "author":["Yashwant"]},  {  "id":107,  "title":"C++",  "author":["Balaguruswami"]},  {  "id":108,  "title":"Phthon",  "author":["XYZ"]}]  }} |

**Phrase Query:**

A phrase query matches multiple terms (words) in sequence.

* Get all books whose title has "The Lightning Thief" phrase. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=title:"The Lightning Thief"&wt=json&indent=true |

**Boolean Operators:**

Lucene supports AND, OR, NOT, "+" (must occur),and "-"(must not occur) as Boolean operators (Note: Boolean operators must be ALL CAPS).

The **AND** operator matches documents where both terms exist anywhere in the text of a single document. This is equivalent to an intersection using sets. The symbol && can be used in place of the word AND.

* Get all books which are available in “English” **and** “French” language and get only ‘id, author and title’ in result. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:English AND language\_str:French&fl=id,title,author&wt=json&indent=true |

The **OR** operator is the default conjunction operator. This means that if there is no Boolean operator between two terms, the OR operator is used. The OR operator links two terms and finds a matching document if either of the terms exist in a document. This is equivalent to a union using set. The symbol || can be used in place of the word OR.

* Get all books whose book stores are in “Hyderabad” **and** are available in “English” **or** “French” language and get only ‘id, author and title’ in result. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=bookstore\_str :Hyderabad AND (language\_str:EnglishOR language\_str:French) &fl=id,title,author&wt=json&indent=true |

The **NOT** operator excludes documents that contain the term after NOT. This is equivalent to a difference using sets. The symbol “!” can be used in place of the word NOT.

* Get all books whose book stores are in “Hyderabad” and **not** in “Delhi”. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=bookstore\_str :Hyderabad NOT bookstore\_str:Delhi&wt=json&indent=true |

**Note:** The NOT operator cannot be used with just one term.

The **"+"** or occur must requires that the term after the "+" symbol exist somewhere in the field of a single document.

* Get all books whose author name must contain“Yashwant” and may contain “Balaguruswami”. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=+YashwantBalaguruswami&df=author&wt=json&indent=true |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":2},  "response":{"numFound":2,"start":0,"docs":[  {  "id":101,  "title":"Java",  "author":"Balaguruswami"},  {  "id":102,  "title":"C",  "author":"Yashwant"}]  }} |

The **"-"** or must not occur excludes documents that contain the term after the "-" symbol.

* Get all books whose author name must not contain “Yashwant” and may contain “Balaguruswami”. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=Yashwant -Balaguruswami&df=author&fl=id,title,author&wt=json&indent=true |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":2},  "**response**":{"numFound":1,"start":0,"docs":[  {  "id":102,  "title":"C",  "author":"Yashwant"}]  }} |

**Wildcards:**

* Get all books whose title starts with “The”, Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=title:The\*&wt=json&indent=true |

* Get all books whose title starts with “The” and ends with “Thief”, Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=title:The\*Thief&wt=json&indent=true |

**Note:** Lucene doesn't support using a **“\*”** symbol as the first character of a search. e.g. “\*Thief”

**Range search:**

* Get all books whose ids are between 101 to 104, Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=id:[101 TO 104]&wt=json&indent=true |

**Sorting:**

* Get all books which are available in “French” language, get only ‘id, author and title’ in result, sort the result on ‘id’ in ascending order and get only 5 results in response. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:French &start=0&rows=5&sort=id asc&fl=id,title,author&wt=json&indent=true |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":2},  "response":{"numFound":6,"start":0,"docs":[  {  "id":101,  "title":"Java",  "author":["Balaguruswami"]},  {  "id":102,  "title":"C",  "author":["Yashwant"]},  {  "id":103,  "title":"The Lightning Thief",  "author":["Rick Riordan"]},  {  "id":104,  "title":"The Sea of Monsters",  "author":["Rick Riordan"]},  {  "id":107,  "title":"C++",  "author":["Balaguruswami"]}]  }} |

* Get all books which are available in “French” language, get only ‘id, author and title’ in result, sort the result on ‘id’ in descending order and get only 10 results in response. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:French &start=0&rows=10&sort=id desc&fl=id,title,author&wt=json&indent=true |

**Default Field (df):**

* Get all books which are available in “French” language using default query (df), get only ‘id, author and title’ in result and sort the result on ‘id’ in ascending order. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=French&sort=id asc&fl=id,title,author&df=language\_str&wt=json&indent=true |

Notice in the query, we are now passing *“df=language\_str”* and *“q=French”.* If “df” is not providedthen an exception will be raised by SOLR.

**Facets:**

* Get all books which are available in “English” language, get only ‘id, author and title’ in result, sort the result on ‘id’ in ascending order and get only 10 results in response with facets on id and author fields. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:French&sort=id asc&start=0&rows=10&fl=id,title,author&wt=json&indent=true&facet=true&facet.field=id&facet.field=author |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":14},  "response":{"numFound":6,"start":0,"docs":[  {  "id":101,  "title":"Java",  "author":["Balaguruswami"]},  {  "id":102,  "title":"C",  "author":["Yashwant"]},  {  "id":103,  "title":"The Lightning Thief",  "author":["Rick Riordan"]},  {  "id":104,  "title":"The Sea of Monsters",  "author":["Rick Riordan"]},  {  "id":107,  "title":"C++",  "author":["Balaguruswami"]},  {  "id":108,  "title":"Phthon",  "author":["XYZ"]}]  },  "facet\_counts":{  "facet\_queries":{},  "facet\_fields":{  "id":[  "101",1,  "102",1,  "103",1,  "104",1,  "107",1,  "108",1,  "105",0],  "author":[  "Balaguruswami",2,  "Rick Riordan",2,  "XYZ",1,  "Yashwant Kanetkar",1,  "George R.R. Martin",0]},  "facet\_dates":{},  "facet\_ranges":{},  "facet\_intervals":{},  "facet\_heatmaps":{}}} |

**Highlighting (hl):**

Take a look at the parameters used in the Solr query of the html.

* hl - When set to true, enables highlighted snippets to be generated in the query response.
* hl.q - Specifies an overriding query term for highlighting.
* hl.fl - Specifies a list of fields to highlight. If you want to highlight all fields than place a \*.
* hl.simple.pre - Specifies the text that should appear before a highlighted term. Default is "<em>"
* hl.simple.post - Specifies the text that should appear after a highlighted term. Default is "</em>"
* hl.snippets - Tells solr the number of highlighted snipets to be shown in your result xml, if you specify a number 4 that solr will highlight starting 4 entries of search keyword in your search result.
* Get all books which are available in “French” language using, get only ‘id, author and title’ in result and sort the result on ‘id’ in ascending order and highlight the results on all matching. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=language\_str:French&sort=id asc&fl=id,title,author&wt=json&indent=true&hl=true&hl.fl=\*&hl.simple.pre=<b>&hl.simple.post=</b>&hl.snippets=5 |

Result:

|  |
| --- |
| {  "responseHeader":{  "status":0,  "QTime":54},  "response":{"numFound":4,"start":0,"docs":[  {  "id":102,  "title":"C",  "author":["Yashwant Kanetkar"]},  {  "id":103,  "title":"The Lightning Thief",  "author":["Rick Riordan"]},  {  "id":104,  "title":"The Sea of Monsters",  "author":["Rick Riordan"]},  {  "id":108,  "title":"Phthon",  "author":["XYZ"]}]  },  "highlighting":{  "102":{  "fullText":["<b>French</b>"],  "language\_str":["<b>French</b>"]},  "103":{  "fullText":["<b>French</b>"],  "language\_str":["<b>French</b>"]},  "104":{  "fullText":["<b>French</b>"],  "language\_str":["<b>French</b>"]},  "108":{  "fullText":["<b>French</b>"],  "language\_str":["<b>French</b>"]}}} |

**Filtered Query (fq):**

* Get all books which are available in “English” language using default query (df), get only ‘id, author and title’ in result and sort the result on ‘id’ in ascending order with filtered query (fq) on “id”. Following will be the search query:

|  |
| --- |
| http://localhost:8983/solr/books/select?q=English&fq=id:[101 TO 104]&sort=id asc&fl=id,title,author&df=language\_str&wt=json&indent=true |

Here, "fq" stands for Filter Query. This parameter can be used to specify a query that can be used to restrict the super set of documents that can be returned, without influencing score. It can be very useful for speeding up complex queries since the queries specified with fq are cached independently from the main query. Caching means the same filter is used again for a later query (i.e. there's a cache hit). See [SolrCaching](https://wiki.apache.org/solr/SolrCaching) to learn about the caches Solr uses.

**Refer to following for search handler configuration:**

1. <http://wiki.apache.org/solr/SearchHandler>

**Refer to following for more details on queries:**

1. <http://lucene.apache.org/core/3_5_0/queryparsersyntax.html>
2. <https://wiki.apache.org/solr/CommonQueryParameters>
3. <https://cwiki.apache.org/confluence/display/solr/Common+Query+Parameters>
4. <https://wiki.apache.org/solr/SolrQuerySyntax>

**Refer to following for more details on facets:**

1. <http://wiki.apache.org/solr/SolrFacetingOverview>
2. <http://wiki.apache.org/solr/SimpleFacetParameters>

**Refer to following for more details on highlighting:**

1. <http://wiki.apache.org/solr/HighlightingParameters>
2. <https://cwiki.apache.org/confluence/display/solr/Highlighting>
3. <https://examples.javacodegeeks.com/enterprise-java/apache-solr/solr-highlighting-example/>

**Refer to following for more details on configuring tokanizers/analysers/synonyms/stop words:**

1. <https://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters>
2. <http://nolanlawson.com/2012/10/31/better-synonym-handling-in-solr/>

**Deleting documents:**

You can delete data by sending a POST request with<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 will specify them right on the command line rather than reference an XML file.

**Use following command to delete a specific document with unique key field ‘101’ but allow search for this document to work until update is not committed externally:**

|  |
| --- |
| java -Dc=books -Ddata=args -Dcommit=false -jar post.jar "<delete><id>101</id></delete>" |

**Use following command to delete a specific document with query‘title:Java’but allow search for this document to work until update is not committed externally:**

|  |
| --- |
| java -Dc=books -Ddata=args -Dcommit=false -jar post.jar "<delete><query>title:java</query></delete>" |

It will print following output on console:

|  |
| --- |
| *SimplePostTool version 5.0.0*  *POSTing args to http://localhost:8983/solr/books/update...*  *Time spent: 0:00:00.039* |

Because we have specified "***commit=false***", a search for ***id:101***or ***title:Java*** we still find the document we have deleted.

**Use following command to permanently delete a specific document with unique key field ‘101’:**

|  |
| --- |
| java -Dc=books -Ddata=args -Dcommit=true -jar post.jar "<delete><id>101</id></delete>" |

**Use following command to permanently delete a specific document with query‘title:Java’:**

|  |
| --- |
| java -Dc=books -Ddata=args -Dcommit=true -jar post.jar "<delete><query>title:java</query></delete>" |

It will print following output on console:

|  |
| --- |
| *SimplePostTool version 5.0.0*  *POSTing args to http://localhost:8983/solr/books/update...*  *COMMITting Solr index changes to http://localhost:8983/solr/books/update...*  *Time spent: 0:00:00.208* |

Because we have specified "***commit=true***", a search for ***id:101***or ***title:Java*** we not find the document we have deleted. It is deleted permanently.

**You can delete a document using CURL command as well:**

|  |
| --- |
| curl -X POST "http://localhost:8983/solr/books/update?commit=true" -H 'Content-Type:application/xml' -d "<delete><query>author:Balaguruswami</query></delete>"  **or**  curl -X POST "http://localhost:8983/solr/books/update?commit=true" -H 'Content-Type:application/xml' -d "<delete><id>101</id></delete>" |

**Know about some common search terminologies:**

1. <https://en.wikipedia.org/wiki/Stop_words>
2. <http://www.gobloggingtips.com/stop-words/>
3. <https://www.searchenginejournal.com/how-is-your-search-engine-handling-synonyms/60129/>

**Configuring Analysers, Tokanizers, Synonyms and Stop words in SOLR:**

Refer to:<https://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters>

**Sample solrconfig.xml with analyzer, tokenizer, synonyms and stop words configuration:**

|  |
| --- |
| *<?xml version='1.0' encoding='UTF-8' ?>*  *<!--This is sample schema -->*  *<schema name='books' version='1.6'>*  *<types>*  *<fieldtype name="string" class="solr.StrField" />*  *<fieldtype name="long" class="solr.TrieLongField" />*  *<fieldType name="text\_general" class="solr.TextField" positionIncrementGap="100">*  *<analyzer type="index">*  *<tokenizer class="solr.StandardTokenizerFactory"/>*  *<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />*  *<filter class="solr.LowerCaseFilterFactory"/>*  *</analyzer>*  *<analyzer type="query">*  *<tokenizer class="solr.StandardTokenizerFactory"/>*  *<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />*  *<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>*  *<filter class="solr.LowerCaseFilterFactory"/>*  *</analyzer>*  *</fieldType>*  *</types>*  *<fields>*  *<field name="id" type="long" indexed="true" stored="true" required="true" />*  *<field name="title" type="string" indexed="true" stored="true" required="true" />*  *<field name="author" type="string" indexed="true" stored="true" required="false"/>*  *<dynamicField name="\*\_str" type="string" indexed="true" stored="true" multiValued="true"/>*  *<field name="fullText" type="text\_general" indexed="true" multiValued="true"/>*  *<field name="text" type="text\_general" indexed="true" multiValued="true"/>*  *</fields>*  *<copyField source="\*" dest="fullText"/>*  *<uniqueKey>id</uniqueKey>*  *</schema>* |