Apache Solr - Indexing Data

In general, **indexing** is an arrangement of documents or (other entities) systematically. Indexing enables users to locate information in a document.

- Indexing collects, parses, and stores documents.
- Indexing is done to increase the speed and performance of a search query while finding a required document.

Indexing in Apache Solr

In Apache Solr, we can index (add, delete, modify) various document formats such as xml, csv, pdf, etc. We can add data to Solr index in several ways.

In this chapter, we are going to discuss indexing −

- Using the Solr Web Interface.
- Using any of the client APIs like Java, Python, etc.
- Using the **post tool**.

In this chapter, we will discuss how to add data to the index of Apache Solr using various interfaces (command line, web interface, and Java client API)

Adding Documents using Post Command

Solr has a **post** command in its **bin/** directory. Using this command, you can index various formats of files such as JSON, XML, CSV in Apache Solr.

Browse through the **bin** directory of Apache Solr and execute the **–h option** of the post command, as shown in the following code block.

```
[Hadoop@localhost bin]$ cd $SOLR_HOME
[Hadoop@localhost bin]$ post -h
```

On executing the above command, you will get a list of options of the **post command**, as shown below.

```
Usage: post -c <collection> [OPTIONS] <files|directories|urls|-d
["..."]>
or post –help
   collection name defaults to DEFAULT_SOLR_COLLECTION if not
specified
OPTIONS
=======
Solr options:
   -url <base Solr update URL> (overrides collection, host, and
port)
   -host <host> (default: localhost)
   -p or -port <port> (default: 8983)
```

```
   -commit yes|no (default: yes)

Web crawl options:
   -recursive <depth> (default: 1)
   -delay <seconds> (default: 10)

Directory crawl options:
   -delay <seconds> (default: 0)

stdin/args options:
   -type <content/type> (default: application/xml)

Other options:
   -filetypes <type>[,<type>,...] (default:

xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott
,otp,ots,
   rtf,htm,html,txt,log)
   -params "<key> = <value>[&<key> = <value>...]" (values must be
   URL-encoded; these pass through to Solr update request)
   -out yes|no (default: no; yes outputs Solr response to
console)
   -format Solr (sends application/json content as Solr commands
   to /update instead of /update/json/docs)

Examples:
* JSON file:post -c wizbang events.json
* XML files: post -c records article*.xml
* CSV file: post -c signals LATEST-signals.csv
* Directory of files: post -c myfiles ~/Documents
* Web crawl: post -c gettingstarted http://lucene.apache.org/Solr
-recursive 1 -delay 1
* Standard input (stdin): echo '{commit: {}}' | post -c
my_collection -
type application/json -out yes –d
```

Student ID	First Name	Lasst Name	Phone	City
001	Rajiv	Reddy	9848022337	Hyderabad
002	Siddharth	Bhattacharya	9848022338	Kolkata
003	Rajesh	Khanna	9848022339	Delhi
004	Preethi	Agarwal	9848022330	Pune

005	Trupthi	Mohanty	9848022336	Bhubaneshwar
006	Archana	Mishra	9848022335	Chennai

```
* Data as string: post -c signals -type text/csv -out yes -d
$'id,value\n1,0.47'
```

Example

Suppose we have a file named **sample.csv** with the following content (in the **bin** directory).

The above dataset contains personal details like Student id, first name, last name, phone, and city. The CSV file of the dataset is shown below. Here, you must note that you need to mention the schema, documenting its first line.

```
id,    first_name,   last_name,   phone_no,      location
001,   Pruthvi,      Reddy,       9848022337,    Hyderabad
002,   kasyap,       Sastry,      9848022338,    Vishakapatnam
003,   Rajesh,       Khanna,      9848022339,    Delhi
004,   Preethi,      Agarwal,     9848022330,    Pune
005,   Trupthi,      Mohanty,     9848022336,    Bhubaneshwar
006,   Archana,      Mishra,      9848022335,    Chennai
```

You can index this data under the core named **sample_Solr** using the **post** command as follows −

```
[Hadoop@localhost bin]$ post -c Solr_sample sample.csv
```

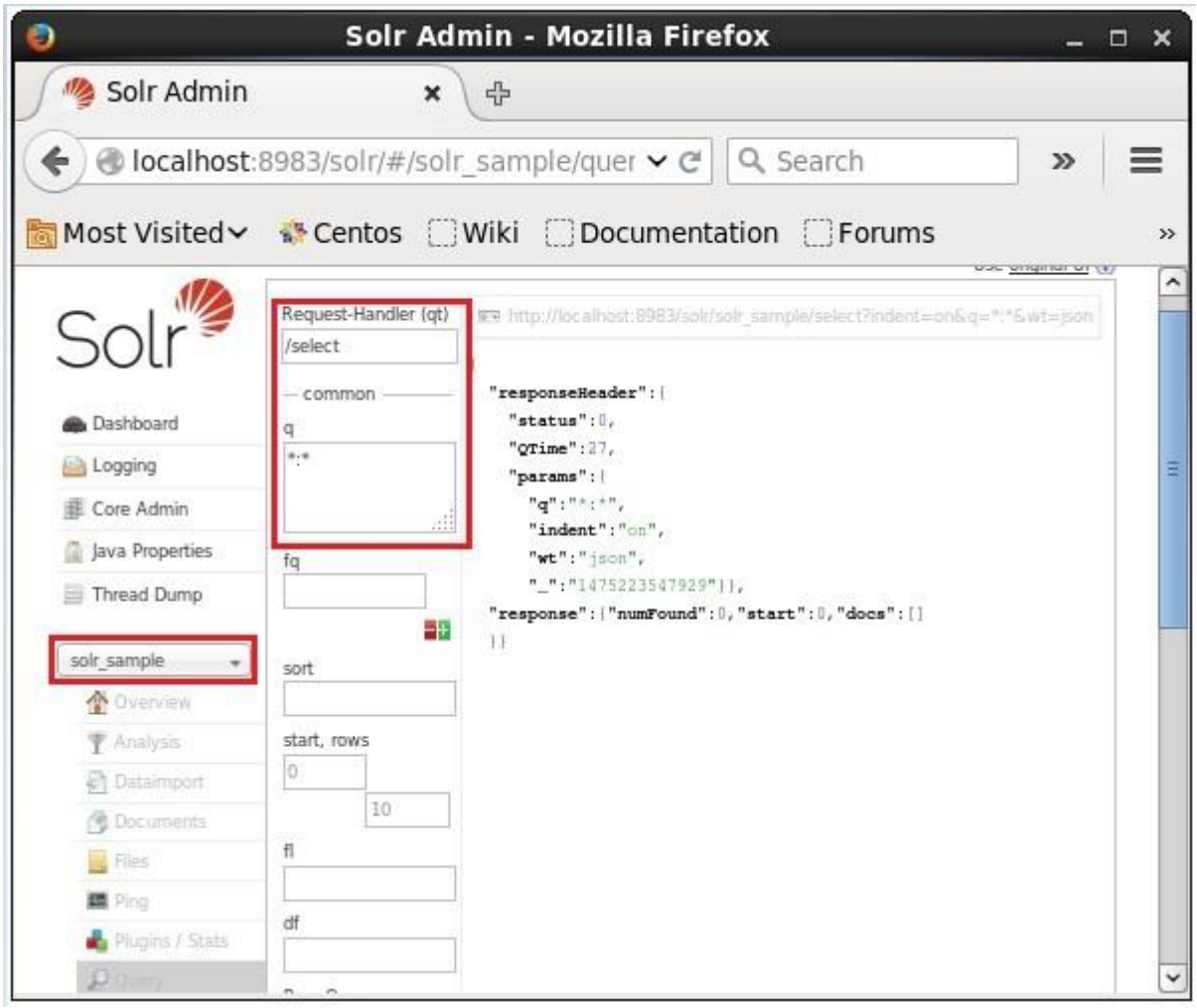
On executing the above command, the given document is indexed under the specified core, generating the following output.

```
/home/Hadoop/java/bin/java -classpath
/home/Hadoop/Solr/dist/Solr-core
6.2.0.jar -Dauto = yes -Dc = Solr_sample -Ddata = files
org.apache.Solr.util.SimplePostTool sample.csv
SimplePostTool version 5.0.0
Posting files to [base] url
http://localhost:8983/Solr/Solr_sample/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,html,txt,log
POSTing file sample.csv (text/csv) to [base]
1 files indexed.
COMMITting Solr index changes to
http://localhost:8983/Solr/Solr_sample/update...
Time spent: 0:00:00.228
```

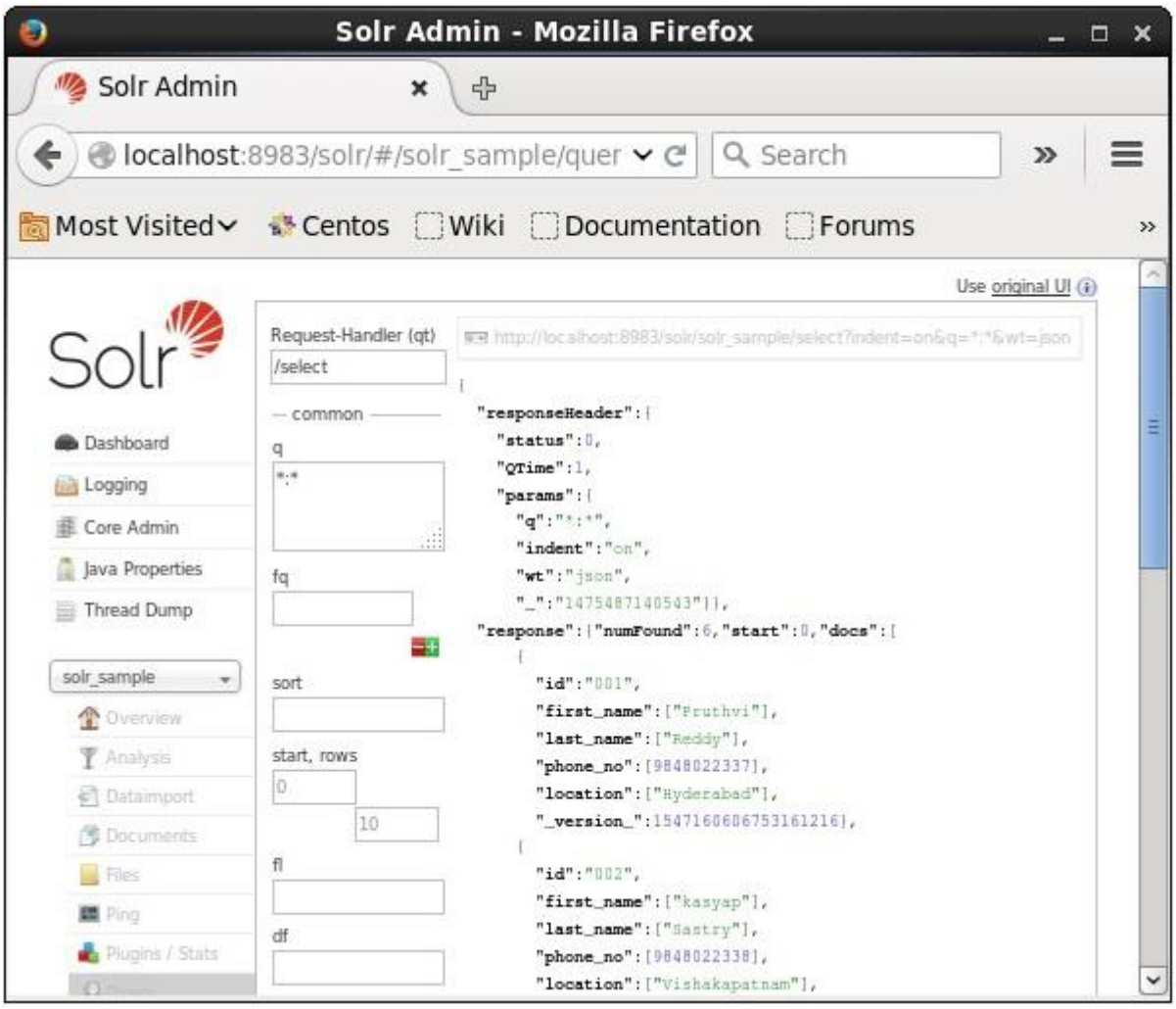
Visit the homepage of Solr Web UI using the following URL −

http://localhost:8983/

Select the core **Solr_sample**. By default, the request handler is **/select** and the query is ":". Without doing any modifications, click the **ExecuteQuery** button at the bottom of the page.



On executing the query, you can observe the contents of the indexed CSV document in JSON format (default), as shown in the following screenshot.



Note − In the same way, you can index other file formats such as JSON, XML, CSV, etc.

Adding Documents using the Solr Web Interface

You can also index documents using the web interface provided by Solr. Let us see how to index the following JSON document.

```
[
   {
      "id" : "001",
      "name" : "Ram",
      "age" : 53,
      "Designation" : "Manager",
      "Location" : "Hyderabad",
   },
   {
      "id" : "002",
      "name" : "Robert",
      "age" : 43,
      "Designation" : "SR.Programmer",
      "Location" : "Chennai",
```

```
    },
    {
        "id" : "003",
        "name" : "Rahim",
        "age" : 25,
        "Designation" : "JR.Programmer",
        "Location" : "Delhi",
    }
]
```

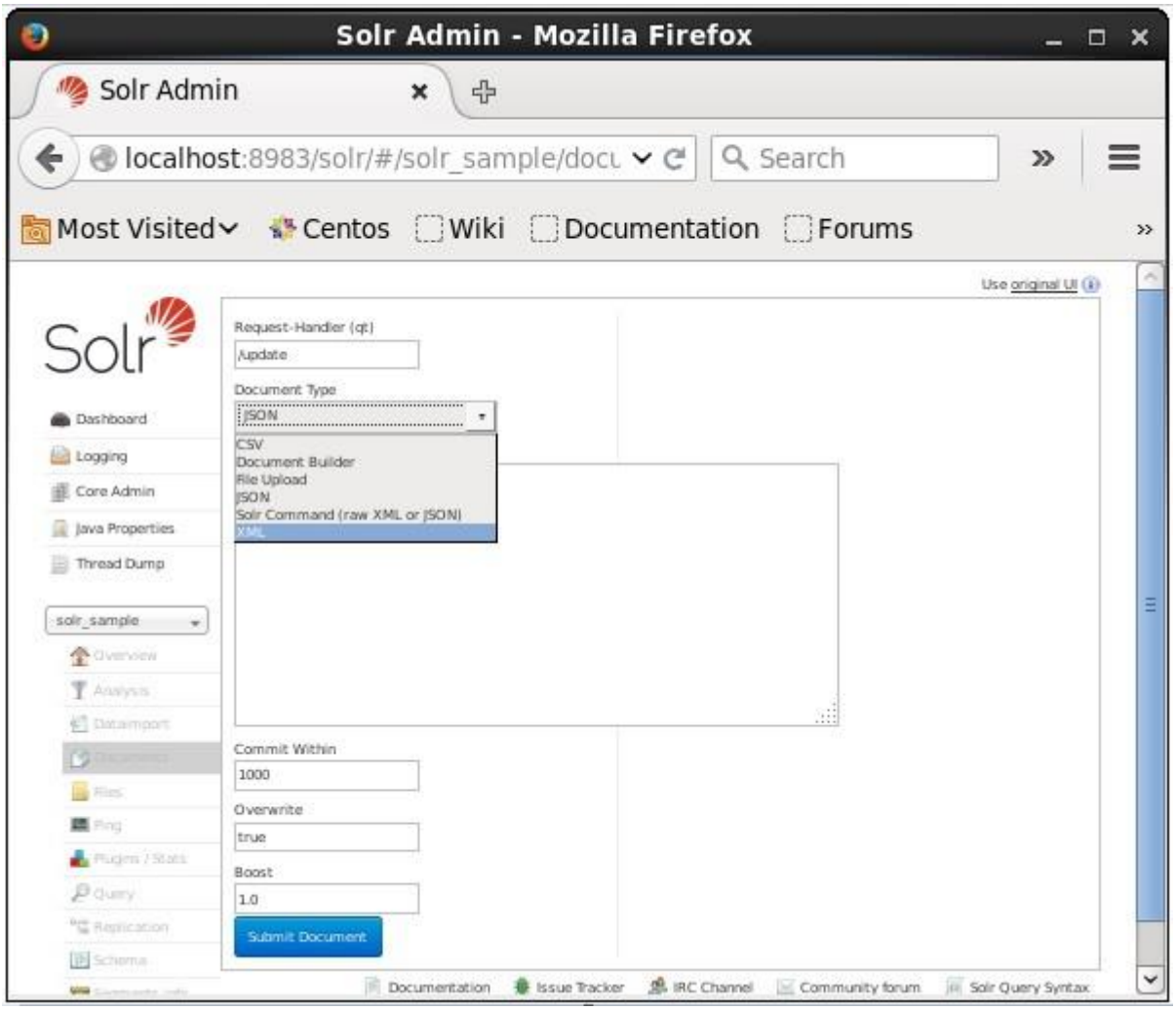
Step 1

Open Solr web interface using the following URL −

http://localhost:8983/

Step 2

Select the core **Solr_sample**. By default, the values of the fields Request Handler, Common Within, Overwrite, and Boost are /update, 1000, true, and 1.0 respectively, as shown in the following screenshot.



Now, choose the document format you want from JSON, CSV, XML, etc. Type the document to be indexed in the text area and click the **Submit Document** button, as shown in the following screenshot.