Universidade Positivo

Fernando de Mello / André Aguiar

Tradução de artigo: Apache Solr

Curitiba
2017

Índice

Apache Solr Tutorial

Solr é uma engine de busca/armazenamento otimizada, escalável e pronta para implantação para a busca de grandes volumes de dados baseados em texto. Solr está pronta para aplicação em empresas, é rápida e altamente escalável. Neste tutorial, nós iremos aprender o básicos de Solr e como você pode colocar em prática.

Audiência.

Este tutorial será de grande ajuda para todos os desenvolvedores que gostariam de entender as funcionalidades básicas do Apache Solr a fim de desenvolver aplicações sofisticadas e de alta performance.

Pré requisitos

É esperado que o leitor tenha bons conhecimentos em programação Java (apesar de não mandatório) e alguma exposição ao ambientes Lucene e Hadoop.

Overview

Solr é uma plataforma open source utilizada para a criação de aplicações de busca. É construída sobre Lucene (engine de busca de texto). As aplicações desenvolvidas sobre o Solr são sofisticadas e entregam com alta performance.
Foi Yonik Seely que criou o Solr em 2004 a fim de adicionar capacidades de busca ao site da empresa CNET Networks. Em Janeiro de 2006, foi feita open source sob a Apache Software Foundation. Sua versão mais recente, Solr 6.0, foi lançada em 2016 com suporte para a execução de queries SQL em paralelo.
Solr pode ser usado em conjunto com o Hadoop. Enquanto o Hadoop gerencia grandes quantidades de dados, Solr nos ajuda a encontrar a informação solicitada desta grande base de origem. Não apenas para busca, Solr pode ser utilizado para fins de armazenamento; Como outras bases de dados NoSQL, seu armazenamento é não relacional.

Features do Apache Solr

Solr é uma criação baseada sobre a API Java do Lucene. Portanto, usando Solr, é possível tomar vantagem das funcionalidades do Lucene. Vamos dar uma olhada em alguns dos recursos mais proeminentes do Solr-

- **APIs RESTful**: Para se comunicar com o Solr, não é mandatório o conhecimentos em programação Java. Ao invés disso você pode utilizar serviços restful para se comunicar

com ele. É possível usar formatos de arquivos XML, JSON e .CSV e obter resultados com estes tipos de arquivos.
- **Busca full text** - Solr provê todas as capacidades necessárias para uma busca full text, como tokens, frases, checagem de escrita, wildcards e auto-complete.
- **Pronta para empresas**: Dependendo da necessidade da organização, Solr pode ser implantado em quaisquer tipos de sistemas (grandes ou pequenos) como único, distribuído, cloud, etc.
- **Flexível e Extensivo**: Ao extender as classes do Java e configurado de acordo, nós podemos customizar os componentes do Solr facilmente.
- **Base de dados NoSQL**: Solr pode ser usado como base de dados NoSQL em escala de big data, onde nós podemos distribuir as tarefas de busca juntamente com um cluster.
- **Interface de Admin**: Solr proporciona uma interface de usuário fácil de usar, amigável, com diversas funcionalidades, onde podemos administrar tarefas de logs, adicionar, deletar, atualizar e buscar documentos.
- **Altamente escalável**: Nós podemos escalar as capacidades do Solr em conjunto com o Hadoop.
- **Otimizado para texto e ordenado por relevância**: Solr é amplamente utilizado para buscar documentos de texto e os resultados são entregues de acordo com a relevância para o usuário.

Diferentemente do Lucene, não é necessário dominar a linguagem de programação Java para trabalhar com o Apache Solr. Ele fornece um excelente serviço pronto para implementação para construir um search box com auto-complete, que o Lucene não oferece. Usando o Solr, nós podemos escalar, distribuir, e gerenciar o index para aplicações de larga escala (Big Data).

Lucene em aplicações de Busca

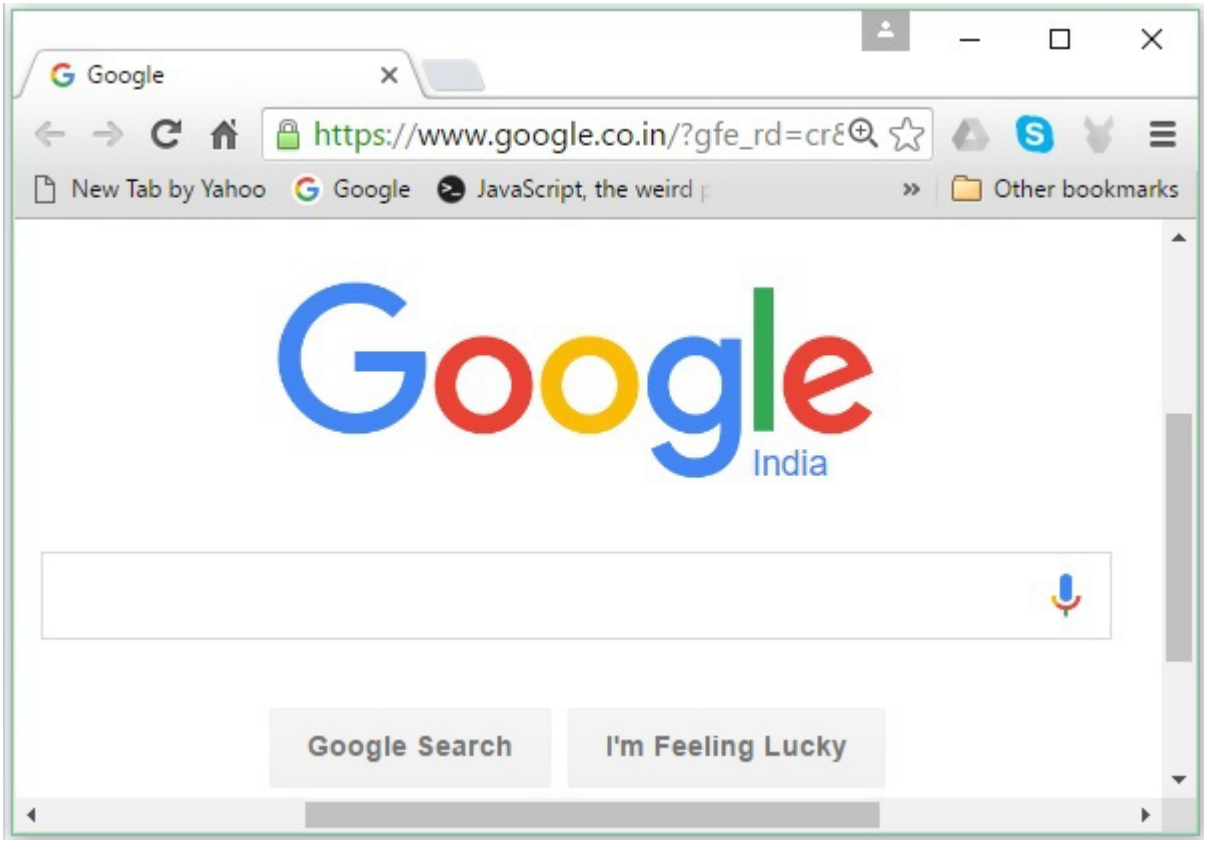
Lucene é uma biblioteca simples, porém muito poderosa, de busca do Java. Pode ser utilizada em qualquer aplicação para adicionar capacidades de busca. Lucene é uma biblioteca escalável e de alta performance usada para indexar e buscar virtualmente qualquer tipo de texto. A biblioteca Lucene oferece o cerne das operações que são requeridas por qualquer aplicação de busca, como index e busca.

Se nós temos um portal web com um grande volume de dados, então nós iremos precisar, muito provavelmente, de um motor de busca em nosso portal para extrair informações relevantes em meio à uma montanha de dados. O Lucene trabalha trabalha como o coração de qualquer aplicação de busca e oferece operações vitais pertinentes ao index e a busca.

Apache Solr - Básico de motores de busca

Um motor de busca (ou search engine, ou também engine de busca) refere-se a uma grande base de dados de Internet com recursos como webpages, grupos de notícias, programas, imagens, etc. Ele ajuda a localizar informações na rede mundial de computadores.

Usuários podem buscar por qualquer informação através de queries em um motor de busca na forma de frases ou palavras chave. O motor de busca então procura em sua base de dados e retorna links relevantes para o usuário.



Componentes de um Motor de Busca

Usualmente, temos três componentes básicos em um motor de busca, listados abaixo:

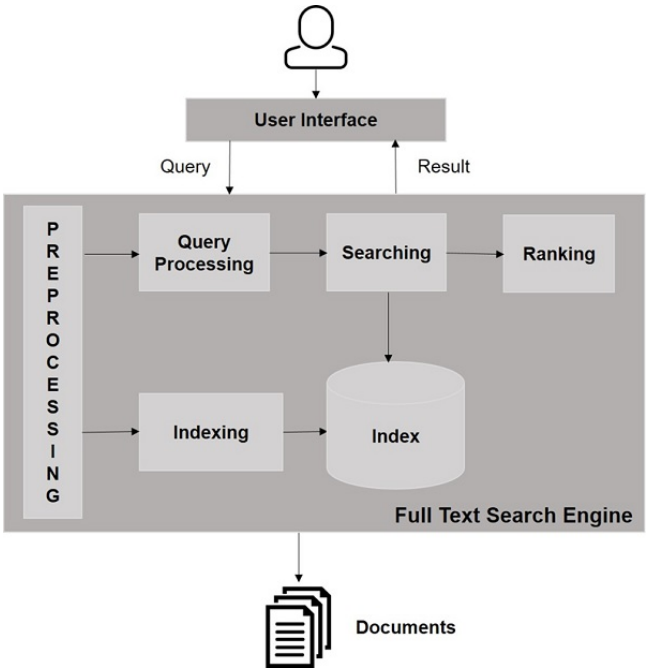
- **Web Crawler**: Web crawlers, também são conhecidos como spiders ou bots. É um componente de software que atravessa a web para coletar informações.
- **Database** (ou base de dados): Toda a informação da internet está armazenada em bases de dados. Elas contém grandes volumes de recursos web.
- **Interfaces de busca**: Este componente é uma interface entre o usuário e a base de dados. Ela ajuda o usuário à buscar dentro da base de dados.

Como funcionam os motores de busca?

Qualquer aplicação de busca requer passar por todas ou pelo menos algumas das seguintes operações.

Passo	Título	Descrição
1	Adquirir conteúdo bruto	O primeiro passo de qualquer aplicação de busca é coletar os conteúdos chave no qual a busca será conduzida.
2	Construção do documento	O próximo passo é a construção do documento (ou documentos, dependendo do caso) a partir dos dados brutos para que a aplicação possa entender e interpretá-los facilmente
3	Analisar o documento	Antes do index começar, o documento é analisado.
4	Indexando o documento	Assim que o documento é construído e analisado, o próximo passo é indexá-lo para que esse documento possa retornar informações baseados em determinadas chaves, ao invés de todo o conteúdo do documento. Indexar é similar aos índices que temos ao final de um livro onde as palavras mais comuns são mostradas junto com o número da página em que aparecem, para que essas palavras possam ser encontradas rapidamente, ao invés de procurar em todo o livro.
5	Interface de usuário para a Busca	Quando os indexes das base de dados estão prontas, a aplicação então pode performar operações de buscas. Para ajudar o usuário a fazer uma busca, a aplicação precisa oferecer uma interface onde o usuário possa inserir textos e inicar os processos de busca.
6	Construindo a query	Quando o usuário faz a requisição de busca com um texto, a aplicação precisa preparar um objeto de query usando aquele texto, que então pode ser usado para requisitar ao index da base de dados os resultados mais relevantes.
7	Buscando a query	Usando a query, o index da base de dados é requisitado para retornar os detalhes relevantes e os conteúdos dos documentos.
8	Apresentar os resultados	Quando o resultado requisitado é recebido, a aplicação deve decidir como apresentar os resultados ao usuários usando sua interface.

Abaixo segue uma ilustração, apresentando uma visão geral do funcionamento dos motores de busca.



Além dessas operações mais básicas, aplicações de busca podem oferecer interfaces de usuários administrador para ajudar administradores à controlarem o nível da busca baseado nos perfis de usuários. A análise dos resultados de busca é outro importante e avançado aspecto de qualquer aplicação de busca.

Instalando o Solr no Hadoop

Siga os passos a seguir para efetuar o download e instalar o Solr.

Passo 1

Abra a homepage do Apache Solr, através do link https://lucene.apache.org/solr/

Passo 2

Clique no botão Download no topo da página inicial. Ao clicar você será redirecionado para a página onde há vários mirrors do Apache Solr. Selecione um mirror e clique sobre o mesmo, e você será redirecionado para a página onde você pode fazer o download do Solr.

Passo 3

Ao clicar, uma pasta chamada Solr-6.2.0.tqz será baixada em sua pasta de Downloads. Extraia o conteúdo do arquivo baixado.

Passo 4

Crie uma pasta chamada Solr no diretório Home do Hadoop e mova o conteúdo do arquivo extraído para lá, conforme o comando abaixo.

```
$ mkdir Solr
$ cd Downloads
$ mv Solr-6.2.0 /home/Hadoop/
```

Verificação

Navegue pela pasta bin no diretório Home do Solr e verifique a instalação usando a opção version, demonstrado no bloco de código abaixo.

```
$ cd bin/
$ ./Solr version
6.2.0
```

Configurando os diretórios

Abra o arquivo .bashrc usando o comando

```
[Hadoop@localhost ~]$ source ~/.bashrc
```

Agora defina o home e os diretórios do Apache Solr como a seguir:

```
export SOLR_HOME = /home/Hadoop/Solr
export PATH = $PATH:/$SOLR_HOME/bin/
```

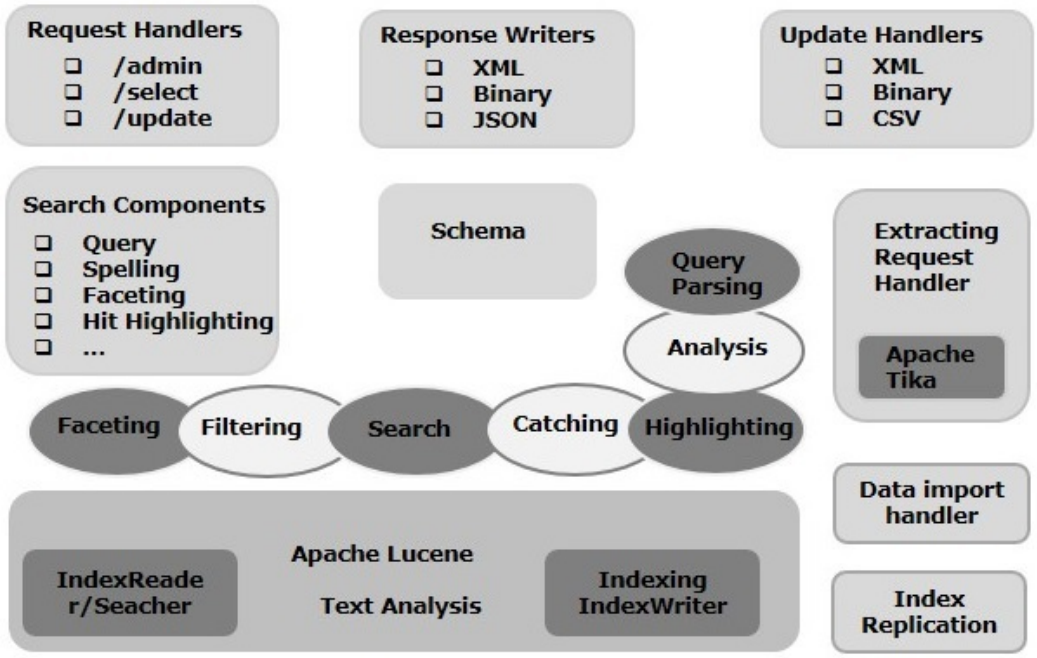
Abra o terminal e execute o seguinte comando

```
[Hadoop@localhost Solr]$ source ~/.bashrc
```

Agora você pode executar os comandos do Solr a partir de qualquer diretório.

Apache Solr Arquitetura

Neste capítulo discutiremos a arquitetura do Apache Solr. A ilustração a seguir demonstra o diagrama da arquitetura do Solr.



Componentes

A seguir estão os principais blocos (componentes) do Apache Solr.

- **Request Handler**: As requisições que são enviadas para o Solr são processadas por esses handlers. As requisições podem ser queries de busca ou atualizações do Index. Baseado em nossa requisição, nós precisamos selecionar o handler de requisições. Para passar uma requisição para o Solr, geralmente mapeia-se o handler para um end-point URI e a requisição especificada será tratada por ele.
- **Componente de Busca**: Um componente de busca é uma feature de busca oferecida pelo Apache Solr. Podem ser verificação gramaticais, busca, quantidade de hits, etc. Estes componentes podem ser registrados como handlers de busca. Múltiplos componentes podem ser registrados em um handler de busca.
- **Análise Sintática de Queries (Query Parser)**: O Query parser analisa as queries que passamos para o Solr e verifica a query atrás de erros de sintaxe. Após este procedimento, ele o traduz para um formato que o Lucene possa compreender.
- **Response Writer**: Um response writer em Apache Solr é o componente que gera o resultado formatado para as queries do usuário. Solr possui suporte para formatos como XML, JSON, CSV, etc. O Solr possui response writers diferentes para cada tipo de resposta.
- **Analyzer/Tokenizer**: O Lucene reconhece dados em formato de tokens. O Apache Solr analisa o conteúdo, divide em tokens e repassa estes tokens para o Lucene. Um Analyzer do Apache Solr examina os campos de textos e gera um fluxo de tokens. O tokenizer quebra este fluxo preparado pelo analyzer em tokens.

- **Processador de Requisições de Update**: Sempre que enviamos uma requisição de update para o Apache Solr, a requisição é executada por um conjunto de plugins (assinatura, log, index), coletivamente conhecidos como o processador de requisições de update (update request processor). Este processador é responsável pelas modificações como adicionar um campo, excluir um campo, etc.

Apache Solr - Terminologia

Neste capítulo iremos tentar entender o real significado de alguns termos que são frequentemente usados quando se trabalha com o Solr.

Terminologia Geral

A lista a seguir é uma compilação dos termos usados entre todos as configurações do Solr:

- **Instance:** Assim como tomcat instance ou uma jetty instance, este termo refere-se ao servidor da aplicação, que executa dentro de uma JVM. O diretório principal do Solr referencia cada uma destas instâncias, onde cada core ou mais podem ser configurados para executar em cada instância.
- **Core:** Enquanto executa-se múltiplos indexes em sua aplicação, você pode possuir múltiplos cores em cada instância, ao invés de múltiplas instâncias cada uma tendo um único core.
- **Home:** O termo $SOLR_HOME refere-se ao diretório principal onde estão todas as informações referentes ao cores e seus indexes, configurações e deéndências.
- **Shard**: Em ambientes distribuídos, os dados são particionados em múltiplas instâncias do Solr, onde cada pedaço dos dados são conhecidos como um Shard. Ele contém um subconjunto de todo o Index.

Terminologia SolrCloud

Em uma das seções passadas abordamos a instalação do Solr em modo standalone. Note que também podemos instalar o Solr em modo distribuído (ambiente cloud) onde o Solr é instalado em um modo master-slave. Em modo distribuído, o index é criado no servidor master e replicado em um ou mais servidores slaves.

Os termos chaves associados ao Solr Cloud são:

- **Node**: No Solr Cloud cada instância é conhecida como Node.
- **Cluster**: Todos os nodes do ambiente combinados dão origem a um cluster.
- **Collection**: Um cluster com index lógico é conhecido como Collection.
- **Shard**: Shard é a porção da coleção que possui uma ou mais réplicas do index.

- **Replica**: No Solr Cloud, uma cópia de um shard que é executada em um node é conhecido como Replica.
- **Leader**: Também pode ser a réplica de um Shard, que distribui as requisições do Solr Cloud para as Replicas seguintes.
- **Zookeeper**: É um projeto Apache que o Solr Cloud usa para centralizar a configuração e coordenação, para gerenciar o cluster e eleger um leader.

Apache Solr - Iniciando o Solr

Arquivos de Configuração

Os principais arquivos de configuração do Apache Solr são:

- **Solr.xml**: é o arquivo no diretório $SOLR_HOME que contém informações relacionadas ao Solr Cloud. Para carregar os cores, o Solr refere-se à este arquivo, que ajuda a os identificar.
- **Solrconfig.xml**: Este arquivo contém as definições e configurações específicas dos cores relacionadas ao gerenciamento de requisições e formatação de resposta, juntamente com index, configuração e gerenciamento de memória.
- **Schema.xml**: Este arquivo conteém todo o esquema juntamente com os campos e os tipos de campos.
- **Core.properties**: Este arquivo contém as configurações específicas do core. Ele é referenciado para a descoberta de cores, pois contém o nome de cada core e o caminho dos dados. pode ser utilizado em qulquer diretório, que será então tratado como o diretórios dos cores.

Iniciando o Solr no Hadoop

Após instalar o Solr, navegue até a pasta bin no diretório home e inicie o Solr com o seguinte comando.

```
[Hadoop@localhost ~]$ cd
[Hadoop@localhost ~]$ cd Solr/
[Hadoop@localhost Solr]$ cd bin/
[Hadoop@localhost bin]$ ./Solr start
```

Este comando inicia o Solr no background, escutando no port 8983 apresentando a seguinte mensagem:

```
Waiting up to 30 seconds to see Solr running on port 8983 [\]
Started Solr server on port 8983 (pid = 6035). Happy searching!
```

Iniciando o Solr em foreground

É possível iniciar o Solr em foreground usando a opção **-f**.

```
[Hadoop@localhost bin]$ ./Solr start –f

5823 INFO  (coreLoadExecutor-6-thread-2) [   ] o.a.s.c.SolrResourceLoader
Adding 'file:/home/Hadoop/Solr/contrib/extraction/lib/xmlbeans-2.6.0.jar' to
classloader
5823 INFO  (coreLoadExecutor-6-thread-2) [   ] o.a.s.c.SolrResourceLoader
Adding 'file:/home/Hadoop/Solr/dist/Solr-cell-6.2.0.jar' to classloader
5823 INFO  (coreLoadExecutor-6-thread-2) [   ] o.a.s.c.SolrResourceLoader
Adding 'file:/home/Hadoop/Solr/contrib/clustering/lib/carrot2-guava-18.0.jar'
to classloader
5823 INFO  (coreLoadExecutor-6-thread-2) [   ] o.a.s.c.SolrResourceLoader
Adding 'file:/home/Hadoop/Solr/contrib/clustering/lib/attributes-binder1.3.1.jar'
to classloader
5823 INFO  (coreLoadExecutor-6-thread-2) [   ] o.a.s.c.SolrResourceLoader
Adding 'file:/home/Hadoop/Solr/contrib/clustering/lib/simple-xml-2.7.1.jar'
to classloader
……
……
……
……
12901 INFO  (coreLoadExecutor-6-thread-1) [   x:Solr_sample] o.a.s.u.UpdateLog
Took 24.0ms to seed version buckets with highest version 1546058939881226240 12902
INFO  (coreLoadExecutor-6-thread-1) [   x:Solr_sample]
o.a.s.c.CoreContainer registering core: Solr_sample
12904 INFO  (coreLoadExecutor-6-thread-2) [   x:my_core] o.a.s.u.UpdateLog Took
16.0ms to seed version buckets with highest version 1546058939894857728
12904 INFO  (coreLoadExecutor-6-thread-2) [   x:my_core] o.a.s.c.CoreContainer
registering core: my_core
```

Iniciando o Solr em outro port

Usando a opção **-p** no comando de início, nós podemos iniciar o Solr em outro port, como mostrado no bloco de código.

```
[Hadoop@localhost bin]$ ./Solr start -p 8984
Waiting up to 30 seconds to see Solr running on port 8984 [-]
Started Solr server on port 8984 (pid = 10137). Happy searching!
```

Parando oSolr

É possível parar o Solr usando o comando **stop**.

```
$ ./Solr stop
```

Este comando para o Solr e apresenta a seguinte mensagem.

```
Sending stop command to Solr running on port 8983 ... waiting 5 seconds to
allow Jetty process 6035 to stop gracefully.
```

Reiniciando o Solr

O comando **restart** pára o Solr por 5 segundos e o reinicia de novo. Você pode reiniciar o Solr com o seguinte comando.

```
./Solr restart
```

Este comando reinicia o Solr e apresenta a seguinte mensagem:

```
Sending stop command to Solr running on port 8983 ... waiting 5 seconds to
allow Jetty process 6671 to stop gracefully.
Waiting up to 30 seconds to see Solr running on port 8983 [|]  [/]
Started Solr server on port 8983 (pid = 6906). Happy searching!
```

Solr - Comando de Ajuda

O comando de ajuda pode ser usado para checar o uso do prompt do Solr e suas opções.

```
[Hadoop@localhost bin]$ ./Solr -help

Usage: Solr COMMAND OPTIONS
   where COMMAND is one of: start, stop, restart, status, healthcheck,
create, create_core, create_collection, delete, version, zk
Standalone server example (start Solr running in the background on port 8984):
   ./Solr start -p 8984
SolrCloud example (start Solr running in SolrCloud mode using localhost:2181
to connect to Zookeeper, with 1g max heap size and remote Java debug options enabled):
   ./Solr start -c -m 1g -z localhost:2181 -a "-Xdebug -
   Xrunjdwp:transport = dt_socket,server = y,suspend = n,address = 1044"
Pass -help after any COMMAND to see command-specific usage information,
such as: ./Solr start -help or ./Solr stop -help
```

Solr – Comando status

Este comando de status pode ser usado para buscar e encontrar quais são as instâncias rodando em seu computador. Ela pode prover informações sobre uma instância do Solr, como sua versão, uso de memória, etc
Você pode checar o status de uma instância do Solr, usando o comando **status**, como a seguir.

```
[Hadoop@localhost bin]$ ./Solr status
```

Na execução, o comando acima mostra os status do Solr como a seguir.

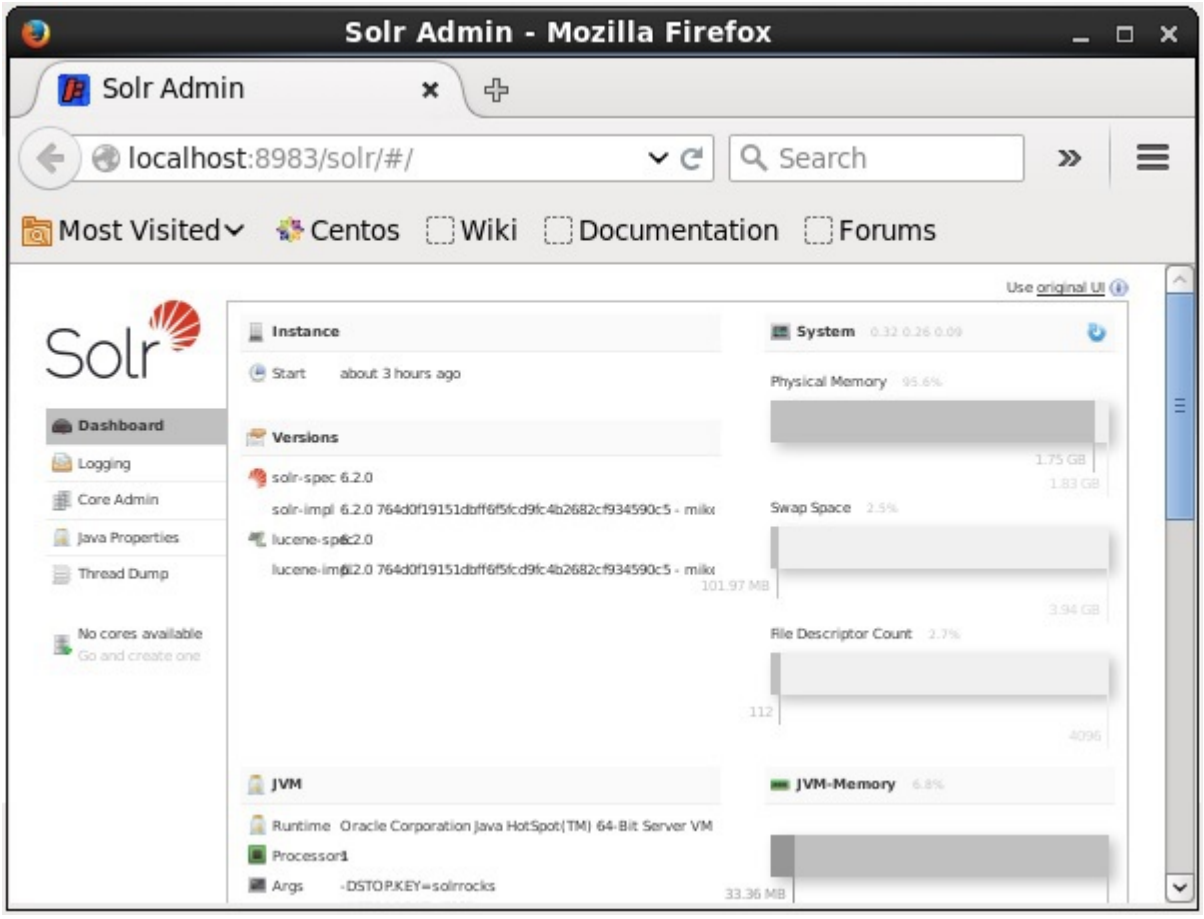
```
Found 1 Solr nodes:

Solr process 6906 running on port 8983 {
  "Solr_home":"/home/Hadoop/Solr/server/Solr",
  "version":"6.2.0 764d0f19151dbff6f5fcd9fc4b2682cf934590c5 -
  mike - 2016-08-20 05:41:37",
  "startTime":"2016-09-20T06:00:02.877Z",
  "uptime":"0 days, 0 hours, 5 minutes, 14 seconds",
  "memory":"30.6 MB (%6.2) of 490.7 MB"
}
```

Solr Admin

Após iniciar o Apache Solr, você pode visitar o homepage da interface web do Solr usando a seguinte URL.

```
Localhost:8983/Solr/
```

A interface do Solr aparecerá como a seguir:

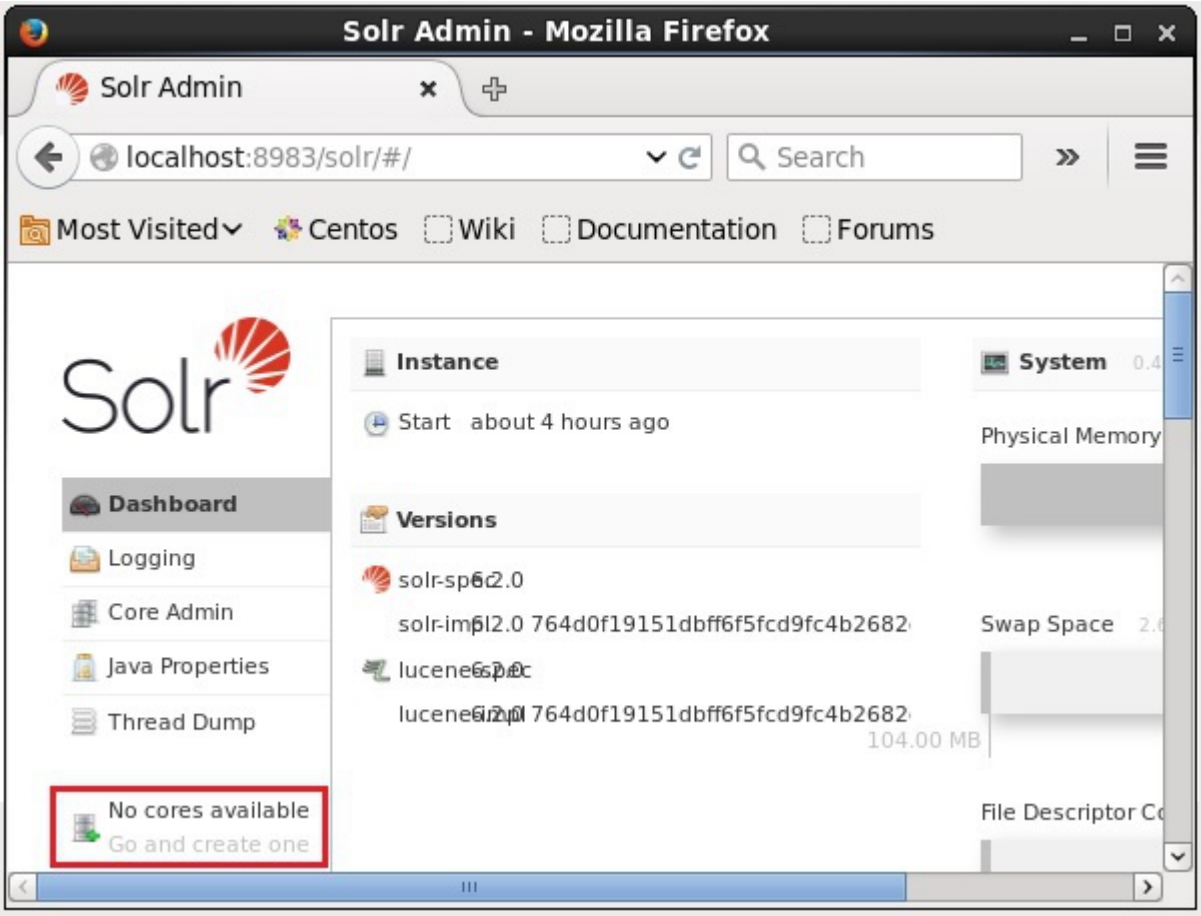


Apache Solr - Core

O Core do Solr é uma instância em execução de um index do Lucene que contém todos os arquivos de configuração requeridas para o uso. Nós precisamos criar um Solr Core para performar operações como index e análises.

Criando um Core

Após instalar e iniciar Solr, você pode conectar ao cliente (interface web) do solr.



Como indicado na captura de tela acima, inicialmente não há nenhum core no Apache Solr. Agora nós veremos como criar um core no Solr.

Usando o comando Create

Uma forma de criar um core é um criar um core sem esquema, usando o comando create, como mostrado abaixo.

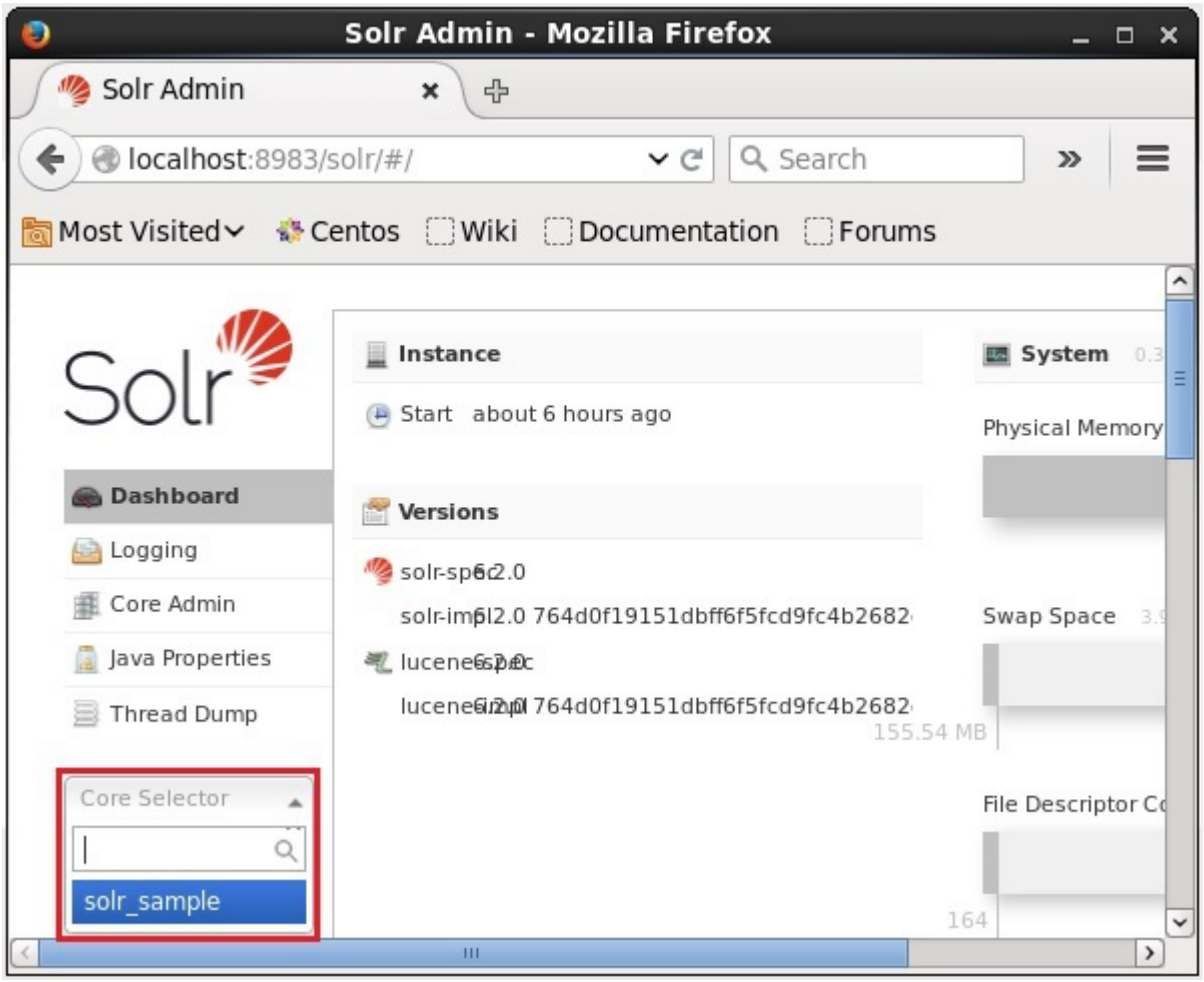
```
[Hadoop@localhost bin]$ ./Solr create -c Solr_sample
```

Aqui, nós tentamos criar um core chamado Solr_sample no Apache Este comando cria um core mostrando a seguinte mensagem:

```
Copying configuration to new core instance directory:
/home/Hadoop/Solr/server/Solr/Solr_sample

Creating new core 'Solr_sample' using command:
http://localhost:8983/Solr/admin/cores?action=CREATE&name=Solr_sample&instanceD
ir = Solr_sample {
  "responseHeader":{
    "status":0,
```

```
    "QTime":11550
  },
  "core":"Solr_sample"
}
```

Você pode criar múltiplos cores no Solr. Do lado esquerdo do Solr Admin, você pode ver um seletor de core, onde você pode selecionar o recentemente criado core, como mostrado na imagem abaixo.



Usando o comando create_core

Alternativamente, você pode criar um core usando o comando **create_core**. Este comando possui as seguintes opções

-c core_name	Nomeie o core que você pretende criar
-p port_name	Port onde você quer criar o core

-d conf_dir	Diretório de configuração do port

Vamos ver como você pode usar o comando **create_core**. Aqui tentaremos criar um core chamado my_core.

[Hadoop@localhost bin]$ ./Solr create_core -c my_core

Ao executar, o comando acima cria um core mostrando a seguinte mensagem:

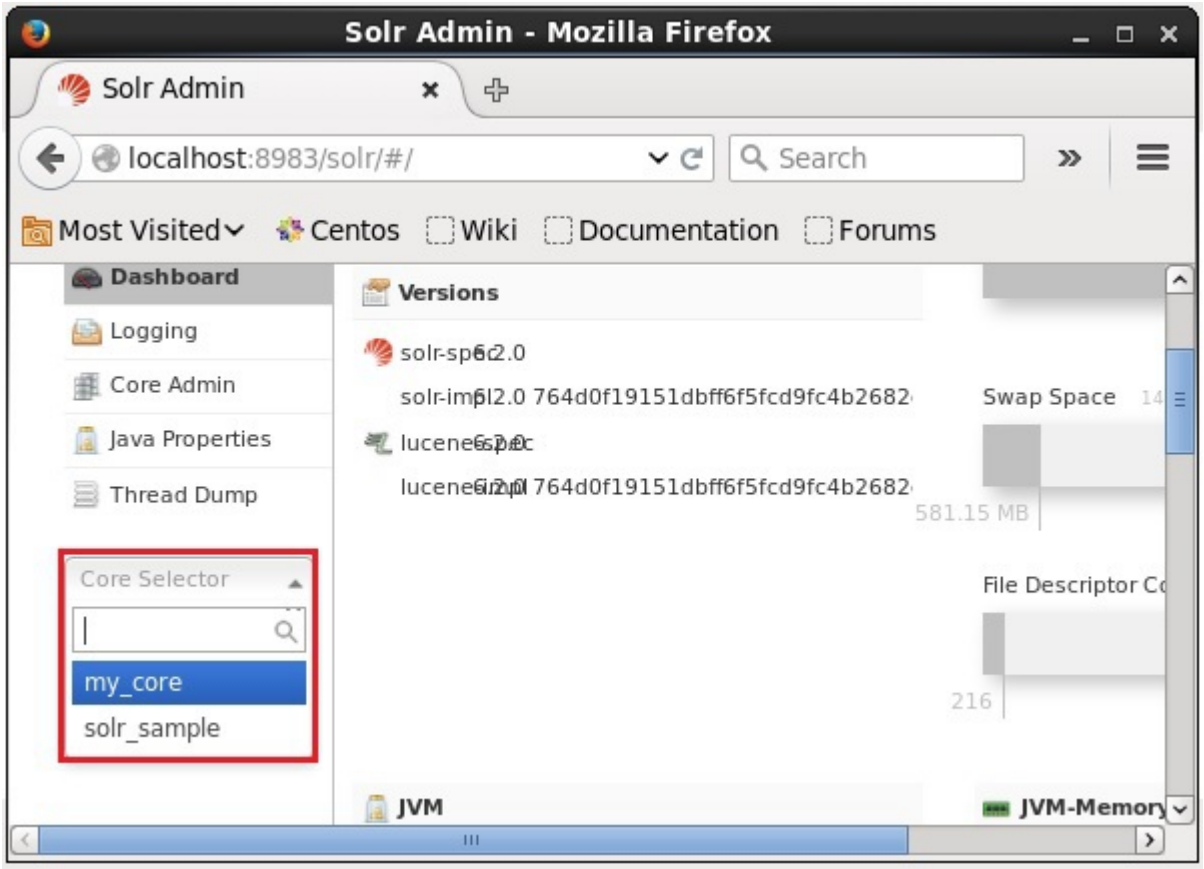
Copying configuration to new core instance directory:

/home/Hadoop/Solr/server/Solr/my_core

Creating new core 'my_core' using command:
http://localhost:8983/Solr/admin/cores?action=CREATE&name=my_core&instanceD
ir = my_core {
  "responseHeader":{
    "status":0,
    "QTime":1346
  },
  "core":"my_core"
}

Deletando um Core

Você pode deletar um core usando o comando **delete** do apache Solr. Supondo que tenhamos criado um core chamado my_core no Solr, como mostrado na captura de tela abaixo:



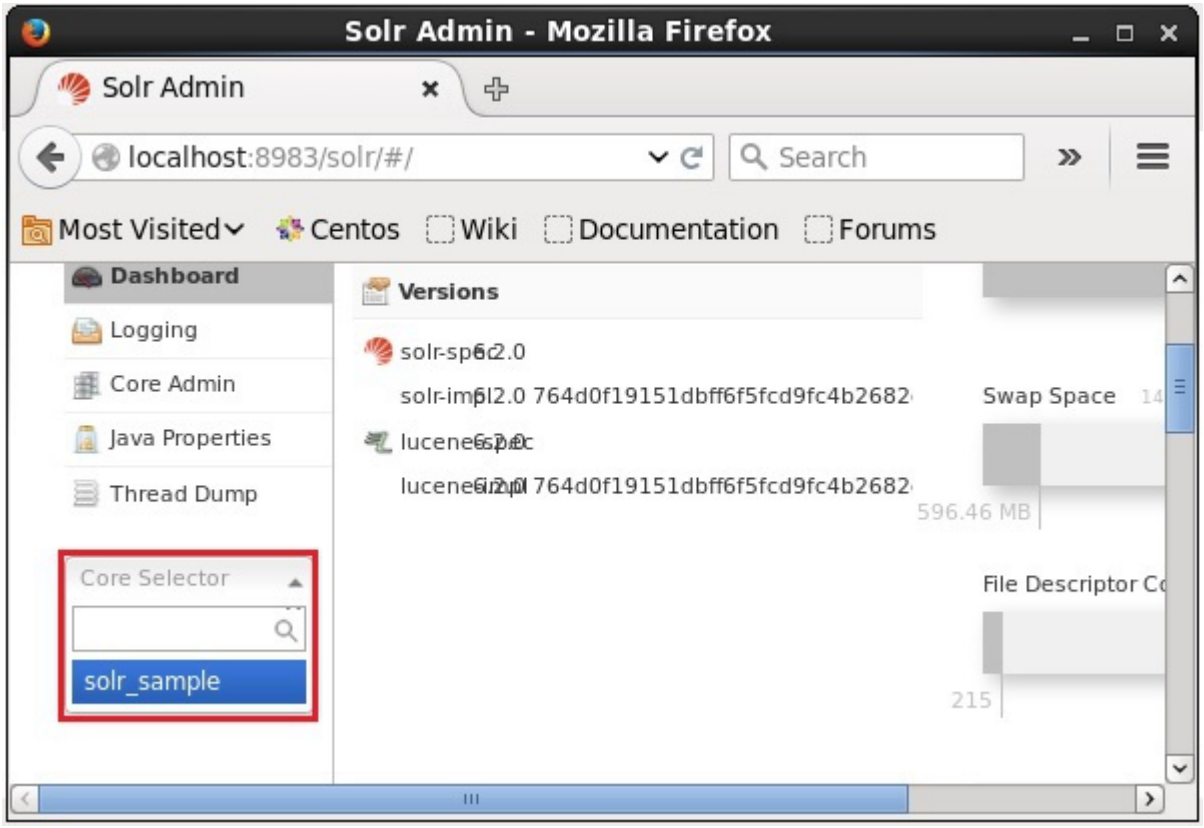
Você pode deletar este core usando o comando delete informando o nome do core à este comando, como mostrado a seguir:

```
[Hadoop@localhost bin]$ ./Solr delete -c my_core
```

Executando o comando acima, o core especificado será deletado como mostra a mensagem abaixo:

```
Deleting core 'my_core' using command:
http://localhost:8983/Solr/admin/cores?action=UNLOAD&core = my_core&deleteIndex
  = true&deleteDataDir = true&deleteInstanceDir = true  {

  "responseHeader" :{
    "status":0,
    "QTime":170
  }
}
```

Você pode abrir a interface web do Solr para verificar se o core foi deletado ou não.



Apache Solr - Indexando Dados

Em linhas gerais, indexar é o rearranjo de documentos (ou outras identidades) sistematicamente. Indexar permite aos usuários localizar informações em um documento.

- Index coleta, limpa e armazena os documentos.
- O index é feito para melhorar velocidade e a performance de uma query de busca enquanto localiza o documento requisitado.

Index no Apache Solr

No APache Solr, nós podemos indexar (Adicionar, deletar , modificar) vários formatos de documentos, como o xml, csv, pdf, etc. Nos podemos adicionar dados ao index do Solr de várias maneiras.

Neste capítulo, nós iremos discutir index:

- Usando a interface Solr Web
- Usando qualquer APIs como Java, Python, etc.
- Usando a ferramenta post.

Iremos discutir também como adicionar dados ao indez do Apache Solr usando várias interfaces (linha de comando, interface web e API cliente Java)

Adicionando documentos usando o comando Post

Solr possui um comando post no seu diretório bin/. Usando este comando você pode indexar vários formatos de arquivos como JSON, XML, CSV no Apache Solr.

Navegue até o diretório bin do Apache Solr e execute a opção -h do comando post, como mostrado abaixo:

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

Ao executar o comando acima, você receberá uma lista de opções do comando post, mostradas abaixo:

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

Exemplo

Supondo que tenhamos um arquivo chamado sample.csv com o seguinte conteúdo (dentro do diretório bin)

Student ID	First Name	Last 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

o arquivo acima contém informações pessoais como o ID do estudante, primeiro nome, sobrenome, telefone e cidade. O arquivo CSV é mostrado abaixo. Aqui, você precisa notar, que é necessário informar o esquema, documentando na primeira linha.

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
```

Você pode indexar estes dados dentro do core sample_Solr usando o comando post, como abaixo:

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

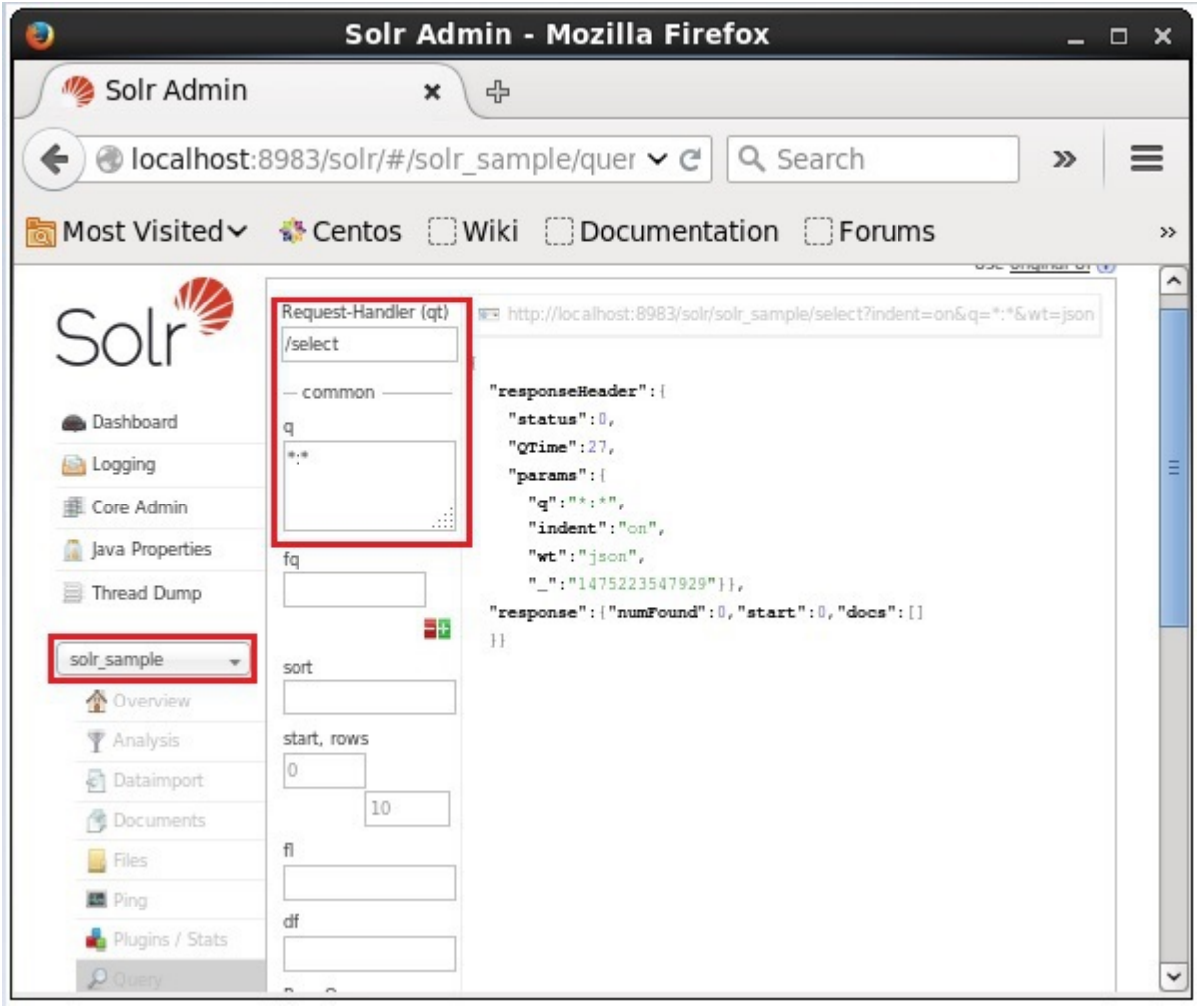
Ao executar o comando acima, o documento é indexado ao core especificado, gerando o seguinte output:

```
/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
```

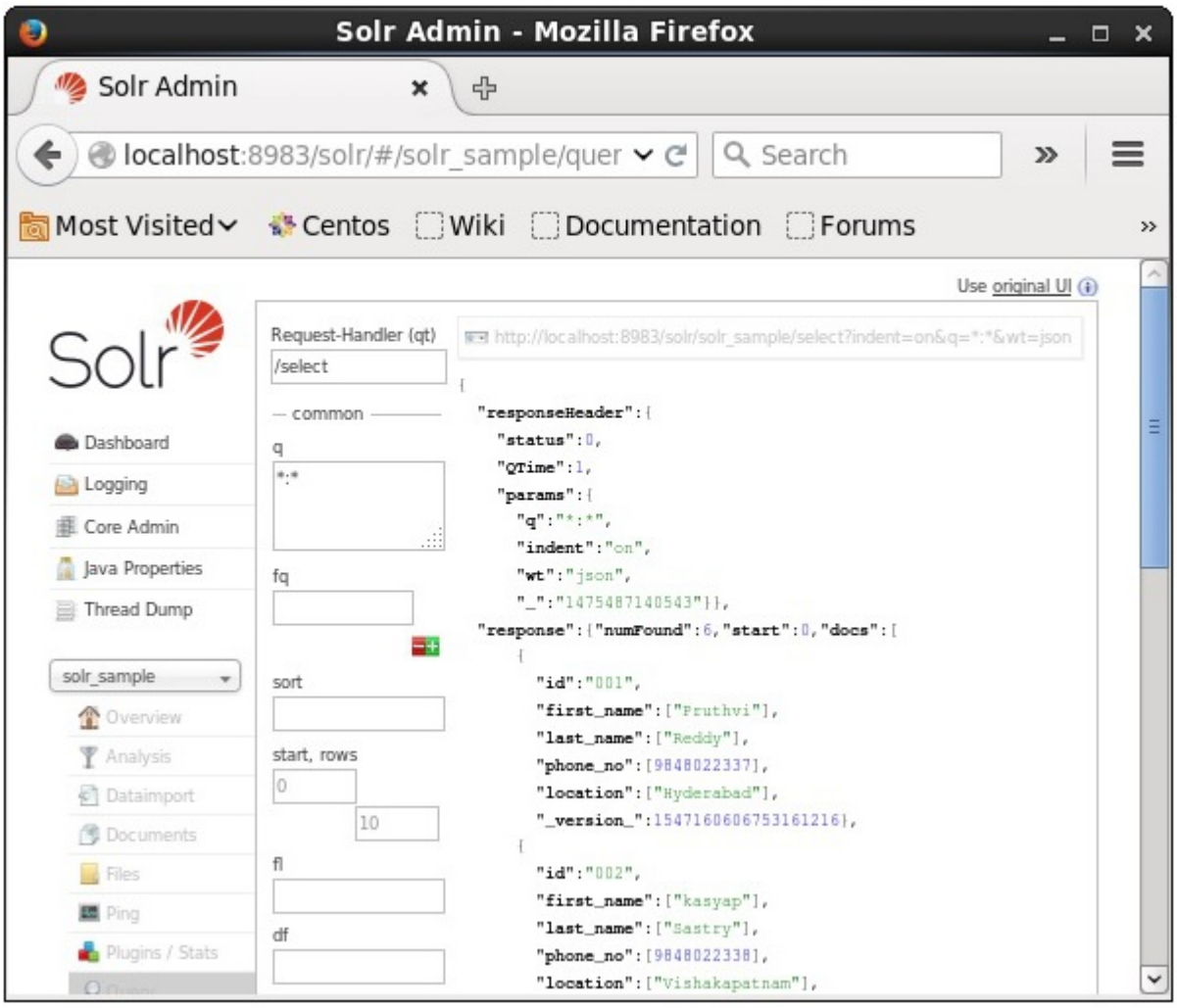
Acesse a homepage da interface de usuáruo do Solr, usando a seguinte URL:

http://localhost:8983/

Selecione o core Solr_sample. Por default, o gerenciador de requisições é /select e a query é ":". Sem fazer nenhuma modificação, clique em ExecuteQuery no final da página:



Ao executar a query, você pode observar os conteúdos do CSV indexado em formato JSON (default), como mostra a imagem a seguir:



NOTA: Da mesma maneira, você pode indexar outros tipos de arquivo, como JSON, XML, CSV, etc.

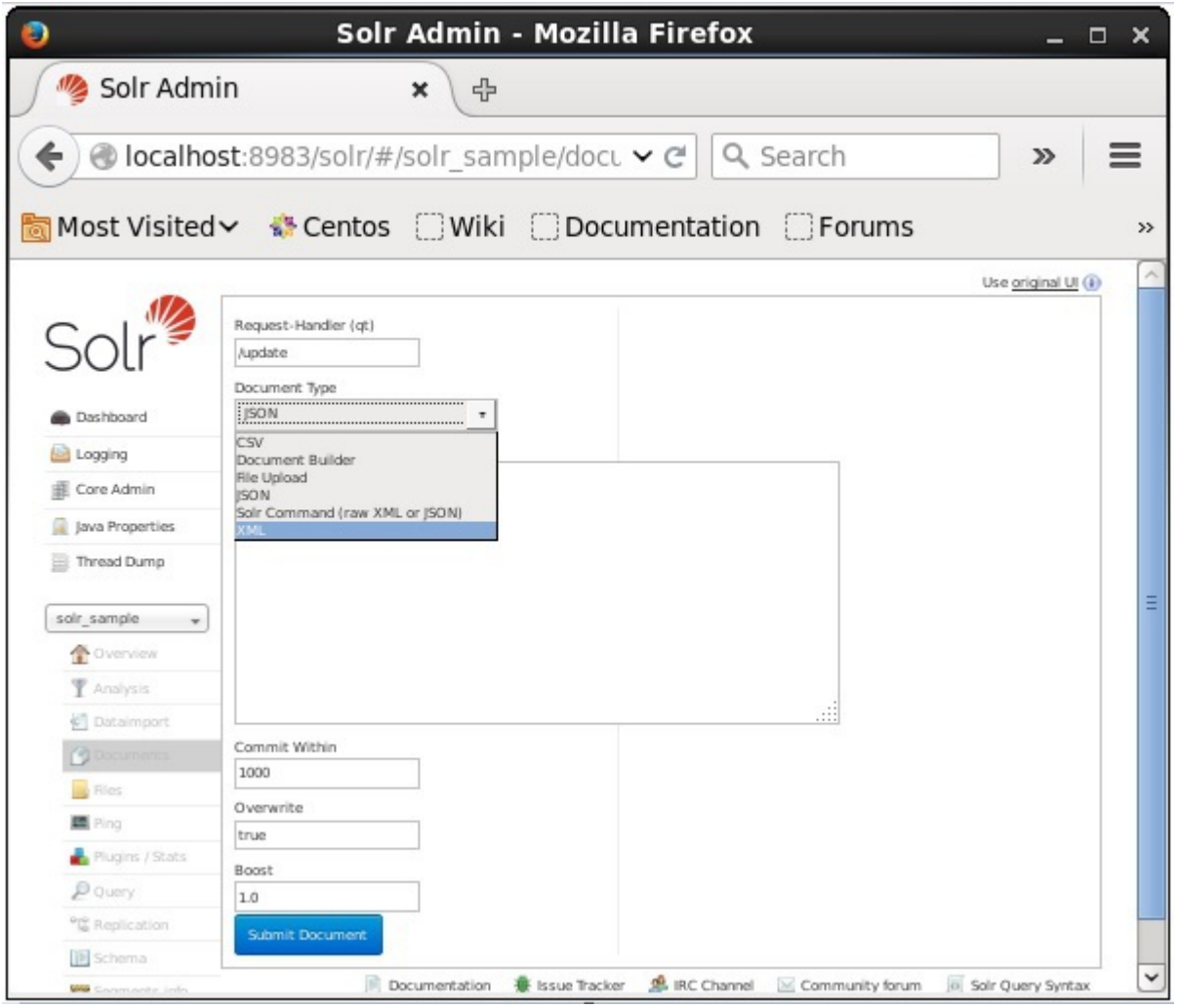
Adicionando documentos usando a interface Web do Solr

Você indexar documentos usando a interface web do Solr. Vamos ver como indexar o seguinte arquivo JSON.

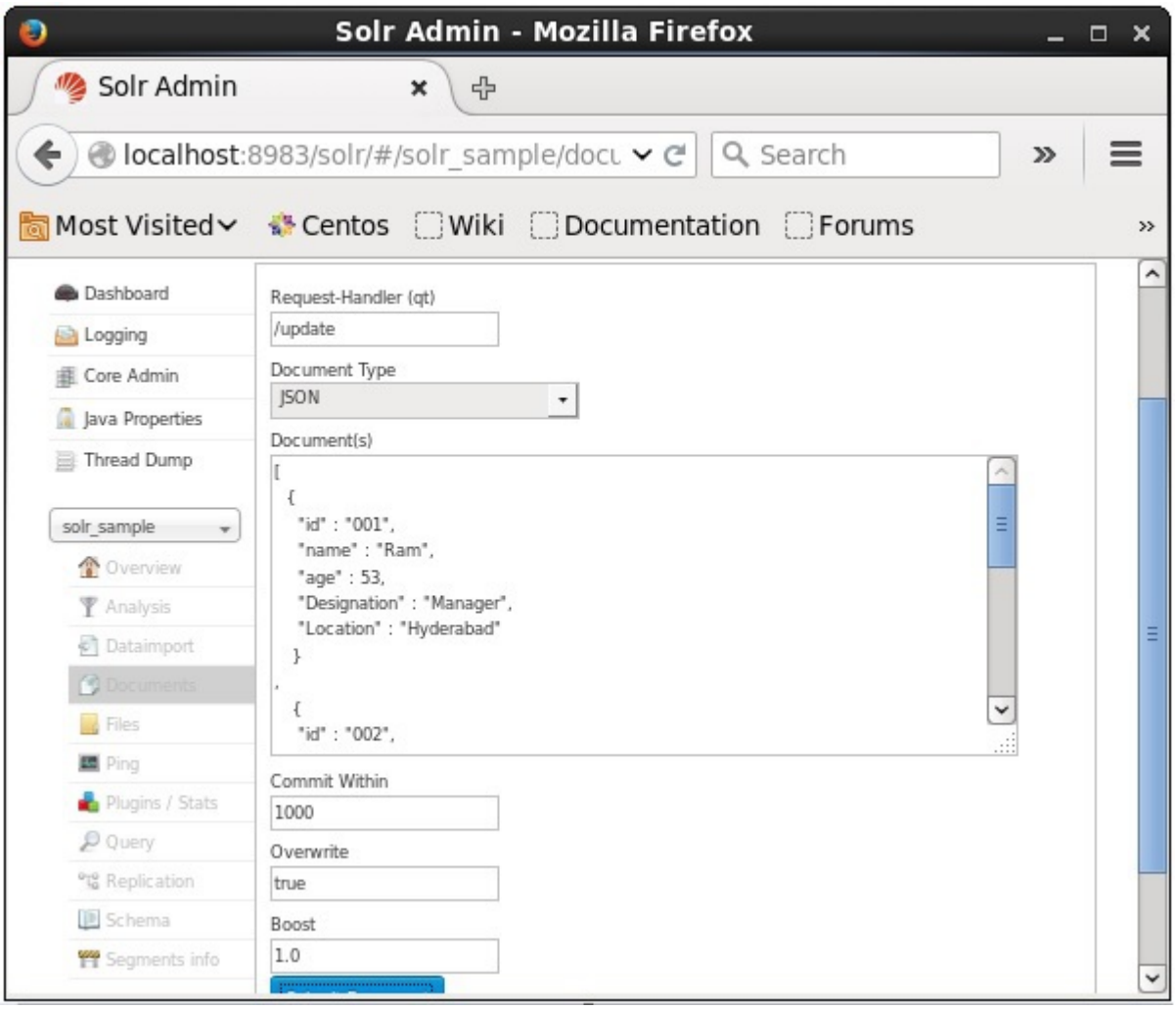
```
[
   {
      "id" : "001",
      "name" : "Ram",
      "age" : 53,
      "Designation" : "Manager",
      "Location" : "Hyderabad",
   },
   {
      "id" : "002",
      "name" : "Robert",
      "age" : 43,
      "Designation" : "SR.Programmer",
      "Location" : "Chennai",
   },
   {
      "id" : "003",
      "name" : "Rahim",
      "age" : 25,
      "Designation" : "JR.Programmer",
      "Location" : "Delhi",
   }
]
```


Passo 1:
Abra a interface web do Solr, pelo URL **http://localhost:8983/**

Passo 2:
Selecione o core Solr_sample. por definição nativa, os valores Request Handler, Common Within, Overwrite e Boost são /update, 1000, true, and 1.0 respectivamente, como mostrado na captura de tela abaixo:



Agora escolha o formato do documento que deseja. Digite a documento a ser indexado na área de texto e clique em Submit Document, como mostra a imagem a seguir:



Adicionando documentos usando a API Cliente Java

A seguir temos o programa Java para adicionar documentos ao Apache Solr. Salve este código em um arquivo chamado AddingDocument.java,

```
import java.io.IOException;

import org.apache.Solr.client.Solrj.SolrClient;
import org.apache.Solr.client.Solrj.SolrServerException;
import org.apache.Solr.client.Solrj.impl.HttpSolrClient;
import org.apache.Solr.common.SolrInputDocument;

public class AddingDocument {
   public static void main(String args[]) throws Exception {
      //Preparing the Solr client
      String urlString = "http://localhost:8983/Solr/my_core";
      SolrClient Solr = new HttpSolrClient.Builder(urlString).build();

      //Preparing the Solr document
      SolrInputDocument doc = new SolrInputDocument();
```

```
      //Adding fields to the document
      doc.addField("id", "003");
      doc.addField("name", "Rajaman");
      doc.addField("age","34");
      doc.addField("addr","vishakapatnam");

      //Adding the document to Solr
      Solr.add(doc);

      //Saving the changes
      Solr.commit();
      System.out.println("Documents added");
   }
}
```

Compile o código acima executando os códigos à seguir no terminal:

```
[Hadoop@localhost bin]$ javac AddingDocument
[Hadoop@localhost bin]$ java AddingDocument
```

Ao executar o comando acima, você receberá a seguinte resposta:

```
Documents added
```

Apache Solr - Adicionando Documentos (XML)

No capítulo anterior, nós explicamos como adicionar dados ao Solr que estão em formatos JSON e CSV. Neste capítulo, demonstraremos como adicionar dados ao index do Apache Solr usando documentos com formato XML.

Dados de Exemplo

Suponha que precisamos adicionar os seguintes dados ao index do Solr usando o formato de documento XML.

Student ID	First Name	Last 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

Para adicionar os dados acima ao index do Solr, nós precisamos preparar um arquivo XML, como mostrado abaixo. Salve este documento em um arquivo chamado sample.xml.

```
<add>
  <doc>
    <field name = "id">001</field>
    <field name = "first name">Rajiv</field>
    <field name = "last name">Reddy</field>
    <field name = "phone">9848022337</field>
    <field name = "city">Hyderabad</field>
  </doc>
  <doc>
    <field name = "id">002</field>
    <field name = "first name">Siddarth</field>
    <field name = "last name">Battacharya</field>
    <field name = "phone">9848022338</field>
    <field name = "city">Kolkata</field>
  </doc>
  <doc>
    <field name = "id">003</field>
    <field name = "first name">Rajesh</field>
    <field name = "last name">Khanna</field>
    <field name = "phone">9848022339</field>
    <field name = "city">Delhi</field>
  </doc>
  <doc>
```

```
    <field name = "id">004</field>
    <field name = "first name">Preethi</field>
    <field name = "last name">Agarwal</field>
    <field name = "phone">9848022330</field>
    <field name = "city">Pune</field>
  </doc>
  <doc>
    <field name = "id">005</field>
    <field name = "first name">Trupthi</field>
    <field name = "last name">Mohanthy</field>
    <field name = "phone">9848022336</field>
    <field name = "city">Bhuwaeshwar</field>
  </doc>
  <doc>
    <field name = "id">006</field>
    <field name = "first name">Archana</field>
    <field name = "last name">Mishra</field>
    <field name = "phone">9848022335</field>
    <field name = "city">Chennai</field>
  </doc>
</add>
```

Como você pode observar, o arquivo XML escrito para adicionar dados ao index, contém 3 tags importantes chamadas <add> </add>, <doc> </doc> e <field> </field>.

- add: Esta é a tag raíz para adicionar documentos ao index. Ela contém um ou mais documentos à serem adicionados.
- doc: Os documetos que iremos adicionar devem estar aninhados nas tags <doc</doc>. Este documento possui os valores na forma de campos (fields)
- field: A tag field contém os nomes e os valores dos campos do documento.

Após preparar o documento, você pode adicioná-lo ao index usando qualquer meio que discutimos no capítulo passado.
Supondo que o arquivo XML exista no diretório bin do Solr e deve ser indexado ao core chamado my_core, você pode adicioná-lo ao Solr usando a ferramenta post, como a seguir.

```
[Hadoop@localhost bin]$ ./post -c my_core sample.xml
```

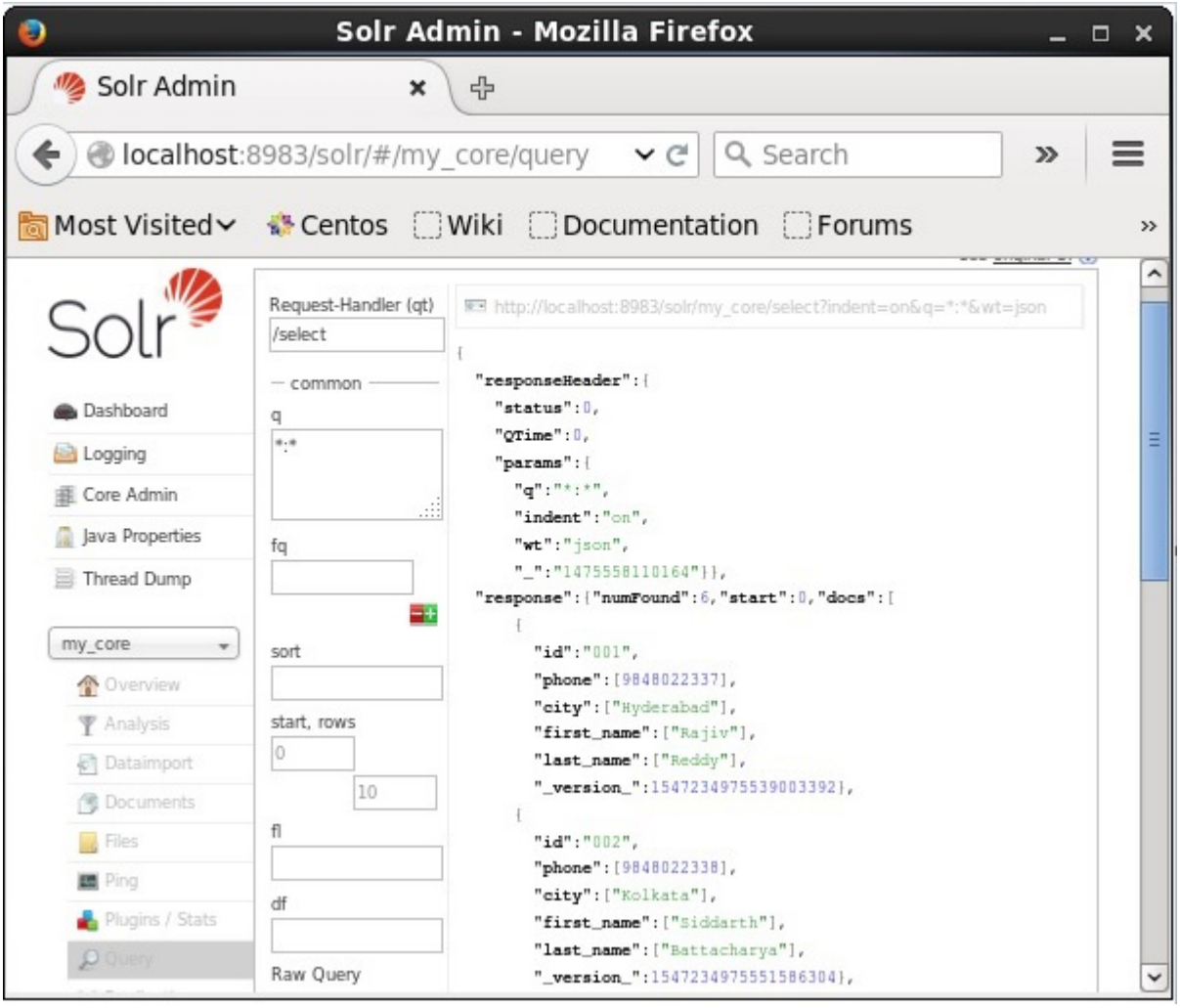
Ao executar o comando acima, você obterá a seguinte resposta:

```
/home/Hadoop/java/bin/java -classpath /home/Hadoop/Solr/dist/Solr-
core6.2.0.jar -Dauto = yes -Dc = my_core -Ddata = files
org.apache.Solr.util.SimplePostTool sample.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/Solr/my_core/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.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/Solr/my_core/update...
Time spent: 0:00:00.201
```

Validação

Visite a homepage do Apache Solr e selecione o core my_core. Tente recuperar todos os documentos usando a query ":" na área de texto q e execute. Ao exectar, você pode observar que os dados desejados foram adicionados ao index do Solr.



Update de dados

Atualizando documentos usando XML.

A seguir é um documento em XML usado para atualizar um campo em um documento existente. Salve este em um arquivo com o nome update.xml.

```
<add>
  <doc>
    <field name = "id">001</field>
    <field name = "first name" update = "set">Raj</field>
    <field name = "last name" update = "add">Malhotra</field>
    <field name = "phone" update = "add">9000000000</field>
    <field name = "city" update = "add">Delhi</field>
  </doc>
</add>
```

Como se pode observar, o arquivo XML escrito para atualizar os dados é como aquele outro que usamos para adicionar documentos. Mas a única diferença é o uso do atributo update no campo field.

Em nosso exemplo, nós usaremos o documento acima e tentaremos atualizar os campos do documento com o id 001.

Suponha que o documento XML exista na pasta bin do Solr. Como nós estamos atualizando o index que já existe no core chamado my_core, você pode atualizar usando a ferramenta pos, como a seguir:

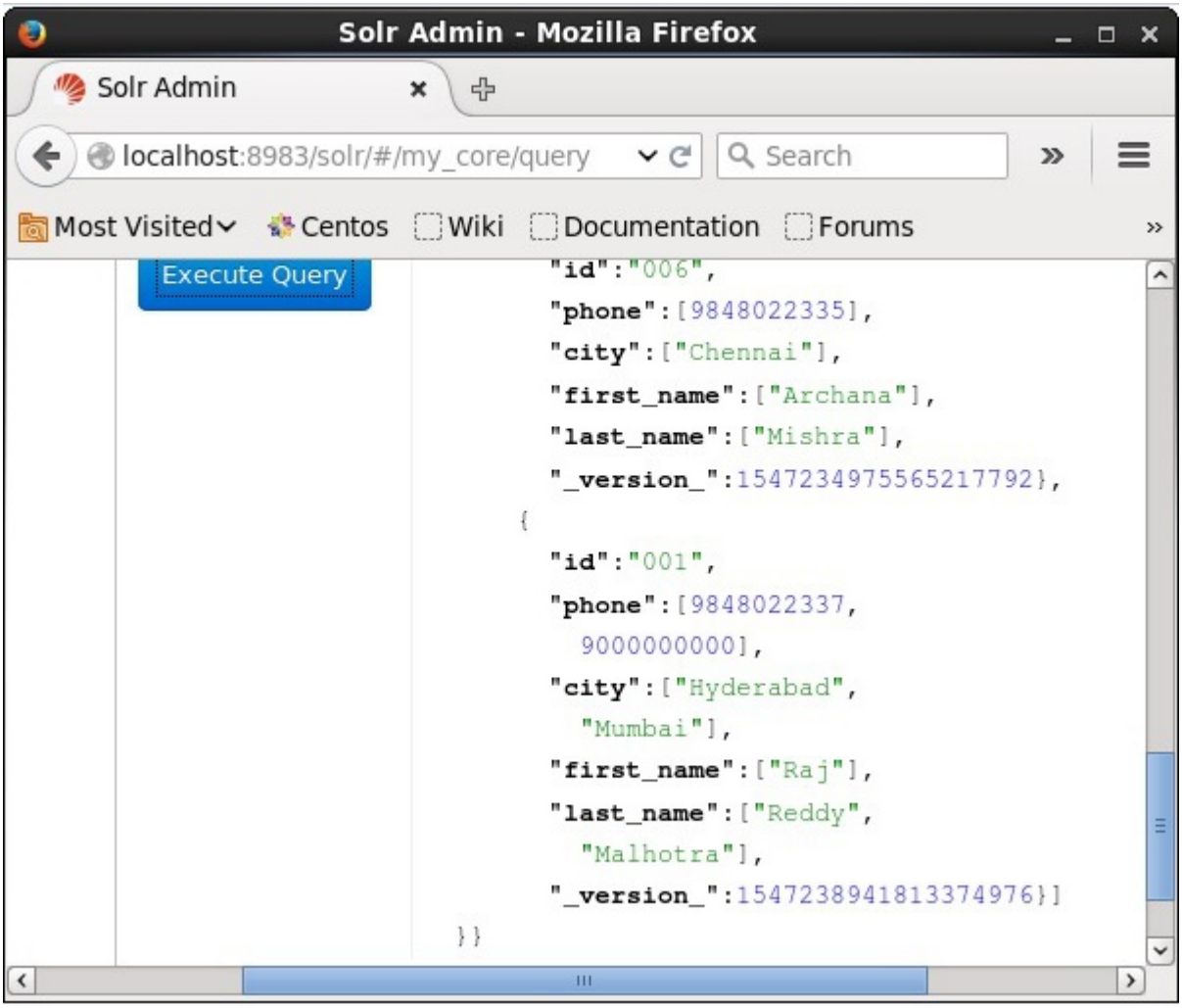
```
[Hadoop@localhost bin]$ ./post -c my_core update.xml
```

Ao executar o comando acima, você receberá a seguinte resposta:

```
/home/Hadoop/java/bin/java -classpath /home/Hadoop/Solr/dist/Solr-core
6.2.0.jar -Dauto = yes -Dc = my_core -Ddata = files
org.apache.Solr.util.SimplePostTool update.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/Solr/my_core/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 update.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/Solr/my_core/update...
Time spent: 0:00:00.159
```

Validação

Acesse a homepage da interface web do Apache Solr e selecione o core my_core. Tente recuperar todos os documentos usando a query ":" na área de texto q e execute a query. Ao executar, você poderá observar que o documento foi atualizado.



Atualizando um documento usando Java (Client API)

A seguir temos um programa em Java para adicionar documentos ao Apache Solr. Salve este código em um arquivo com o nome UpdatingDocument.Java.

```
import java.io.IOException;

import org.apache.Solr.client.Solrj.SolrClient;
import org.apache.Solr.client.Solrj.SolrServerException;
import org.apache.Solr.client.Solrj.impl.HttpSolrClient;
import org.apache.Solr.client.Solrj.request.UpdateRequest;
```

```
import org.apache.Solr.client.Solrj.response.UpdateResponse;
import org.apache.Solr.common.SolrInputDocument;

public class UpdatingDocument {
   public static void main(String args[]) throws SolrServerException,
IOException {
      //Preparing the Solr client
      String urlString = "http://localhost:8983/Solr/my_core";
      SolrClient Solr = new HttpSolrClient.Builder(urlString).build();

      //Preparing the Solr document
      SolrInputDocument doc = new SolrInputDocument();

      UpdateRequest updateRequest = new UpdateRequest();
      updateRequest.setAction( UpdateRequest.ACTION.COMMIT, false, false);
      SolrInputDocument myDocumentInstantlycommited = new SolrInputDocument();

      myDocumentInstantlycommited.addField("id", "002");
      myDocumentInstantlycommited.addField("name", "Rahman");
      myDocumentInstantlycommited.addField("age","27");
      myDocumentInstantlycommited.addField("addr","hyderabad");

      updateRequest.add( myDocumentInstantlycommited);
      UpdateResponse rsp = updateRequest.process(Solr);
      System.out.println("Documents Updated");
   }
}
```

Compile o código acima executando os seguintes comandos no terminal:

```
[Hadoop@localhost bin]$ javac UpdatingDocument
[Hadoop@localhost bin]$ java UpdatingDocument
```

Ao executar o comando acima, você receberá a seguinte resposta.

```
Documents updated
```

Deletando Documentos

Para deletar documentos do Index do Apache Solr, nós precisamos especificar os ID dos documentos à serem deletados dentro das tags <delete></delete>.

```
<delete>
  <id>003</id>
  <id>005</id>
  <id>004</id>
  <id>002</id>
```

</delete>

Aqui, este código XML é usado para deletar os documentos com ID entre 003 e 005. Salve este código em um arquivo com o nome de delete.xml.
Se você deseja deletar os documentos do Index pertencentes ao core chamado my_core, então você pode pode postar o arquivo delete.xml usando a ferramenta post, como mostrado abaixo:

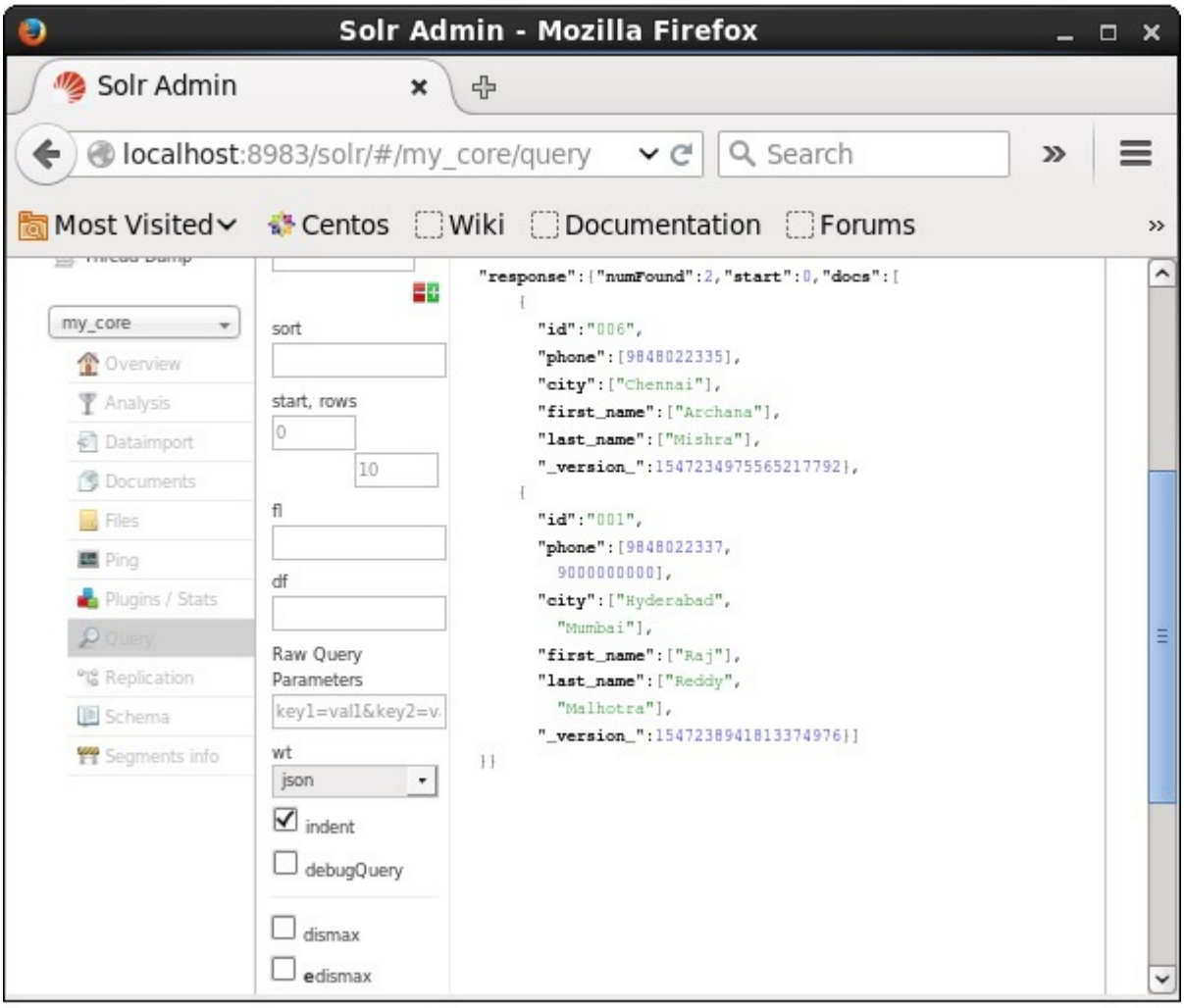
[Hadoop@localhost bin]$ ./post -c my_core delete.xml

Ao executar o comando acima, você receberá o seguinte output:

```
/home/Hadoop/java/bin/java -classpath /home/Hadoop/Solr/dist/Solr-core
6.2.0.jar -Dauto = yes -Dc = my_core -Ddata = files
org.apache.Solr.util.SimplePostTool delete.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/Solr/my_core/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 delete.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/Solr/my_core/update...
Time spent: 0:00:00.179
```

Validação

Visite a homepage da interface web do Apache Solr e selecione o core como my_core. Tente recuperar todoas os documentos passando a query “:” na área de texto q e execute a query. Ao executar, você poderá observar que os arquivos especificados foram deletados.



Deletando um campo

Algumas vezes torna-se necessário deletar documentos baseados em seus campos do que em seus ID. Por exemplo, nós podemos precisar deletar documentos onde a cidade é Chennai. Nestes casos, você precisa especificar o nome e o valor do campo com as tags <query></query>.

```
<delete>
   <query>city:Chennai</query>
</delete>
```

Salve como delete_field.xml e execute a operação de delete no core chamado my_core usando a ferramenta post do Solr.

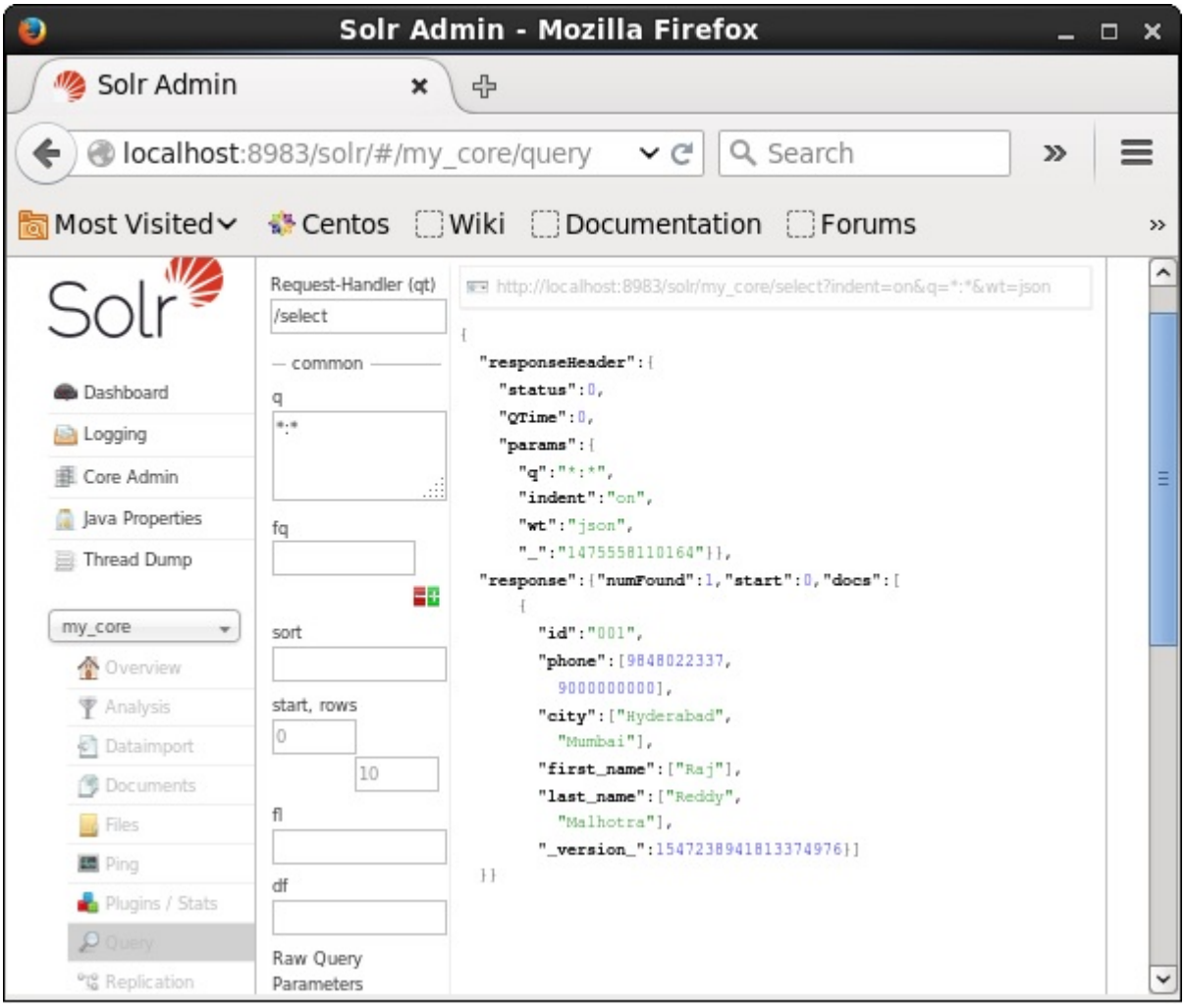
```
[Hadoop@localhost bin]$ ./post -c my_core delete_field.xml
```

Executando o comando acima, produzirá a seguinte resposta.

```
/home/Hadoop/java/bin/java -classpath /home/Hadoop/Solr/dist/Solr-core
6.2.0.jar -Dauto = yes -Dc = my_core -Ddata = files
org.apache.Solr.util.SimplePostTool delete_field.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/Solr/my_core/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 delete_field.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/Solr/my_core/update...
Time spent: 0:00:00.084
```

Validação

Visite a homepage da interface web do Apache Solr e selecione o core chamado my_core. Tente recuperar todos os documentos passando a query ":" na área de texto q e execute. Ao executar, você observará que os documentos contendo os campos especificados foram deletados.



Deletando todos os documentos

Assim como deletar um campo em específico, se você desejar deletar todos os campos de um index, você somente necessita executar o símbolo “:” entre as tags <query></query>, como mostrado abaixo.

```
<delete>
   <query>*:*</query>
</delete>
```

Salve como delete_all.xml e execute a operação delete no core chamado my_core usando a ferramenta post do Solr.

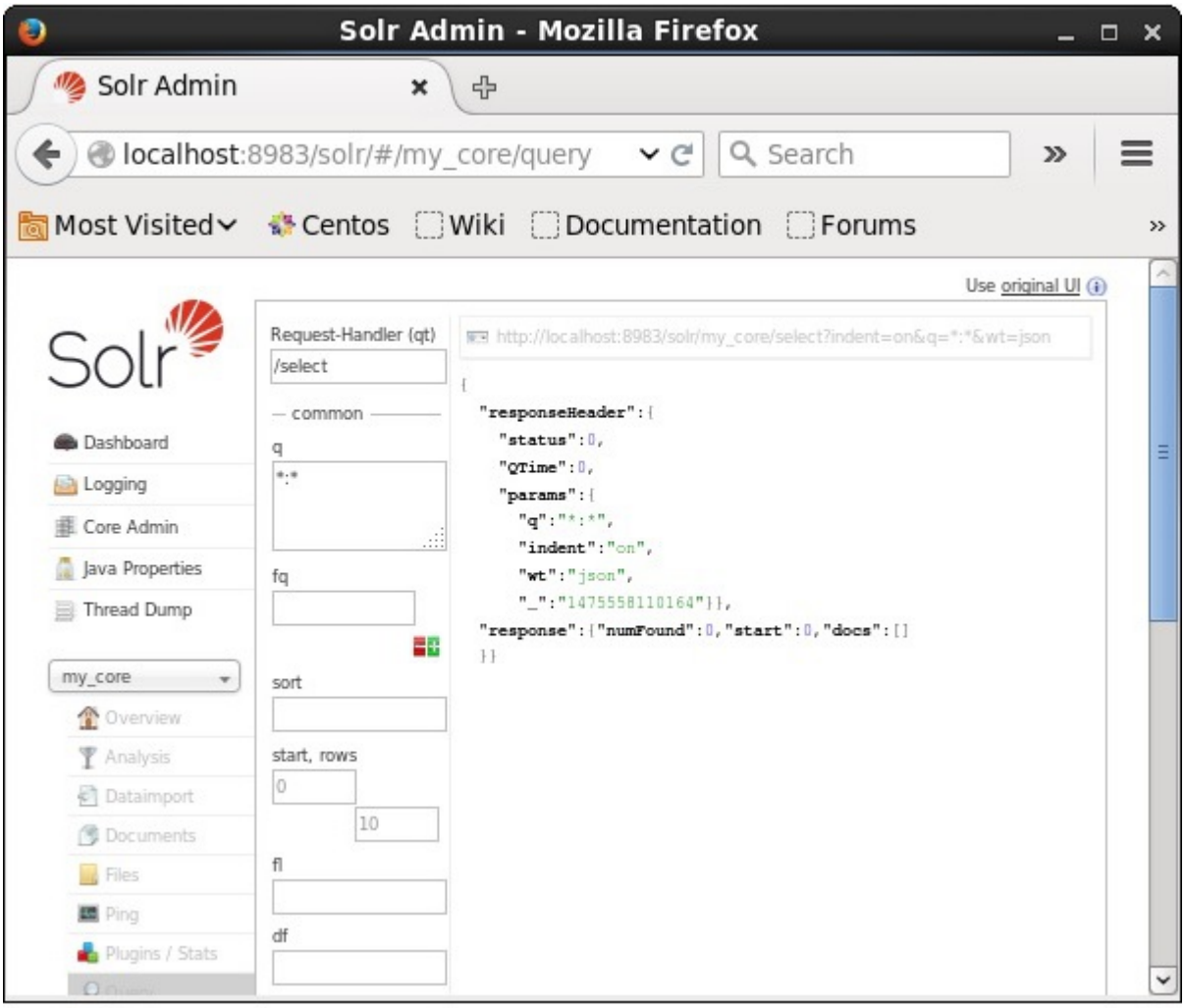
```
[Hadoop@localhost bin]$ ./post -c my_core delete_all.xml
```

Ao executar o comando acima, teremos o seguinte resultado:

```
/home/Hadoop/java/bin/java -classpath /home/Hadoop/Solr/dist/Solr-core
6.2.0.jar -Dauto = yes -Dc = my_core -Ddata = files
org.apache.Solr.util.SimplePostTool deleteAll.xml
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/Solr/my_core/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 deleteAll.xml (application/xml) to [base]
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/Solr/my_core/update...
Time spent: 0:00:00.138
```

Validação

Visite a homepage da interface web do Apache Solr e selecione o core my_core. Tente recuperar todos os documentos usando a query “:” na área de texto q e execute. Após executar, você poderá observar que todos os documentos foram deletados.



Deletando todos os documentos usando JAVA.

À seguir temos um código Java para adicionar documentos ao index do Apache Solr. Salve este código em um arquivo com o nome DeletingAllDocuments.java.

```
import java.io.IOException;

import org.apache.Solr.client.Solrj.SolrClient;
import org.apache.Solr.client.Solrj.SolrServerException;
import org.apache.Solr.client.Solrj.impl.HttpSolrClient;
import org.apache.Solr.common.SolrInputDocument;

public class DeletingAllDocuments {
   public static void main(String args[]) throws SolrServerException,
IOException {
      //Preparing the Solr client
      String urlString = "http://localhost:8983/Solr/my_core";
      SolrClient Solr = new HttpSolrClient.Builder(urlString).build();

      //Preparing the Solr document
```

```
      SolrInputDocument doc = new SolrInputDocument();

      //Deleting the documents from Solr
      Solr.deleteByQuery("*");

      //Saving the document
      Solr.commit();
      System.out.println("Documents deleted");
   }
}
```

Compile o código acima ao executar os seguintes comandos no terminal:

```
[Hadoop@localhost bin]$ javac DeletingAllDocuments
[Hadoop@localhost bin]$ java DeletingAllDocuments
```

Ao executar os comandos acima, você receberá a seguinte resposta:

```
Documents deleted
```

Apache Solr - Recuperar Dados Usando JAVA

Neste capítulo, iremos discutir como recuperar dados usando Java. Supondo que temos um documento chamado sample.csv com o seguinte conteúdo:

```
001,9848022337,Hyderabad,Rajiv,Reddy
002,9848022338,Kolkata,Siddarth,Battacharya
003,9848022339,Delhi,Rajesh,Khanna
```

Você pode indexar estes dados dentro do core chamado sample_Solr usando o comando post.

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

Abaixo temos o script Java para adicionar documentos ao index do Apache Solr. Salve este código em um arquivo chamado RetrievingData.java.

```
import java.io.IOException;

import org.apache.Solr.client.Solrj.SolrClient;
import org.apache.Solr.client.Solrj.SolrQuery;
import org.apache.Solr.client.Solrj.SolrServerException;
import org.apache.Solr.client.Solrj.impl.HttpSolrClient;
import org.apache.Solr.client.Solrj.response.QueryResponse;
import org.apache.Solr.common.SolrDocumentList;
```

```
public class RetrievingData {
   public static void main(String args[]) throws SolrServerException,
IOException  {
      //Preparing the Solr client
      String urlString = "http://localhost:8983/Solr/my_core";
      SolrClient Solr = new HttpSolrClient.Builder(urlString).build();

      //Preparing Solr query
      SolrQuery query = new SolrQuery();
      query.setQuery("*:*");

      //Adding the field to be retrieved
      query.addField("*");

      //Executing the query
      QueryResponse queryResponse = Solr.query(query);

      //Storing the results of the query
      SolrDocumentList docs = queryResponse.getResults();
      System.out.println(docs);
      System.out.println(docs.get(0));
      System.out.println(docs.get(1));
      System.out.println(docs.get(2));

      //Saving the operations
      Solr.commit();
   }
}
```

Compile o código acima executando os seguintes comandos no terminal:

```
[Hadoop@localhost bin]$ javac RetrievingData
[Hadoop@localhost bin]$ java RetrievingData
```

Ao executar os comandos acima, você obterá a seguinte resposta.

```
{numFound = 3,start = 0,docs = [SolrDocument{id=001, phone = [9848022337],
city = [Hyderabad], first_name = [Rajiv], last_name = [Reddy],
_version_ = 1547262806014820352}, SolrDocument{id = 002, phone = [9848022338],
city = [Kolkata], first_name = [Siddarth], last_name = [Battacharya],

_version_ = 1547262806026354688}, SolrDocument{id = 003, phone = [9848022339],
city = [Delhi], first_name = [Rajesh], last_name = [Khanna],

_version_ = 1547262806029500416}]}

SolrDocument{id = 001, phone = [9848022337], city = [Hyderabad], first_name = [Rajiv],
last_name = [Reddy], _version_ = 1547262806014820352}

SolrDocument{id = 002, phone = [9848022338], city = [Kolkata], first_name = [Siddarth],
```

last_name = [Battacharya], _version_ = 1547262806026354688}

SolrDocument{id = 003, phone = [9848022339], city = [Delhi], first_name = [Rajesh], last_name = [Khanna], _version_ = 1547262806029500416}

Apache Solr - Consultas de Dados

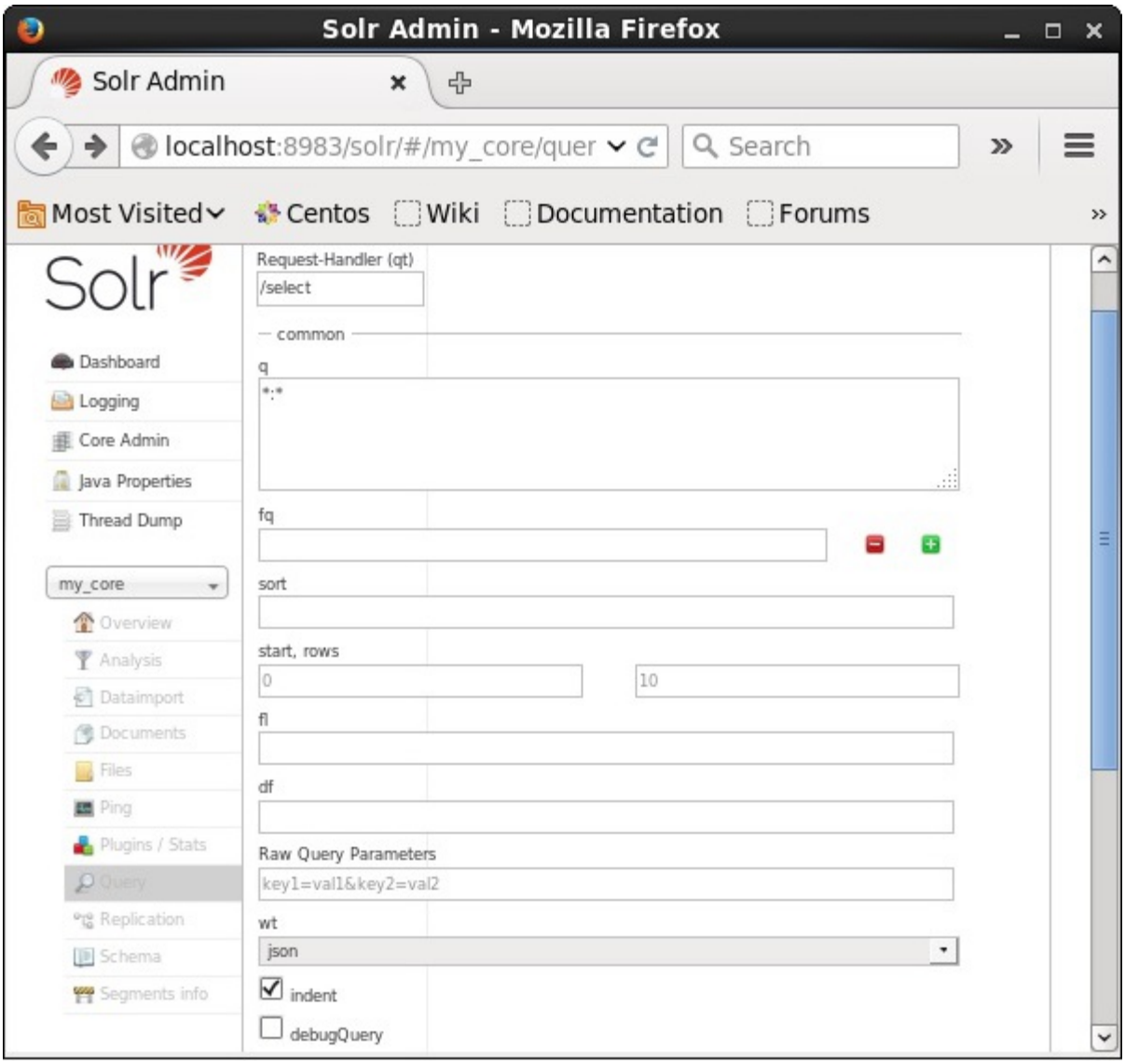
Juntamente ao armazenamento de dados, o Apache Solr fornece a facilidade de executar consultas quando requisitado. O Solr fornece alguns parâmetros para que possamos consultar os dados armazenados em sua memória.

Na tabela a seguir, estão listados os vários parâmetros de pesquisa disponíveis no Apache Solr.

Parameter	Description
q	O principal parâmetro de consulta do Apache Solr, onde os documentos são ordenados de acordo com a sua semelhança com este parâmetro.
fq	Este é parâmetro de filtro de consulta do Apache Solr. Ele restringe os resultados à documentos correspondentes à este filtro.
start	O parâmetro start representa o offset inicial dos resultados da página. O valor por default deste parâmetro é 0.
rows	Este parâmetro representa o número de documentos que podem ser exibidos por página. Valor default deste parâmetro é 10.
sort	Este parâmetro especifica a quantidade de campos, separados por vírgula, baseados no qual os resultados da consulta devem ser classificados.

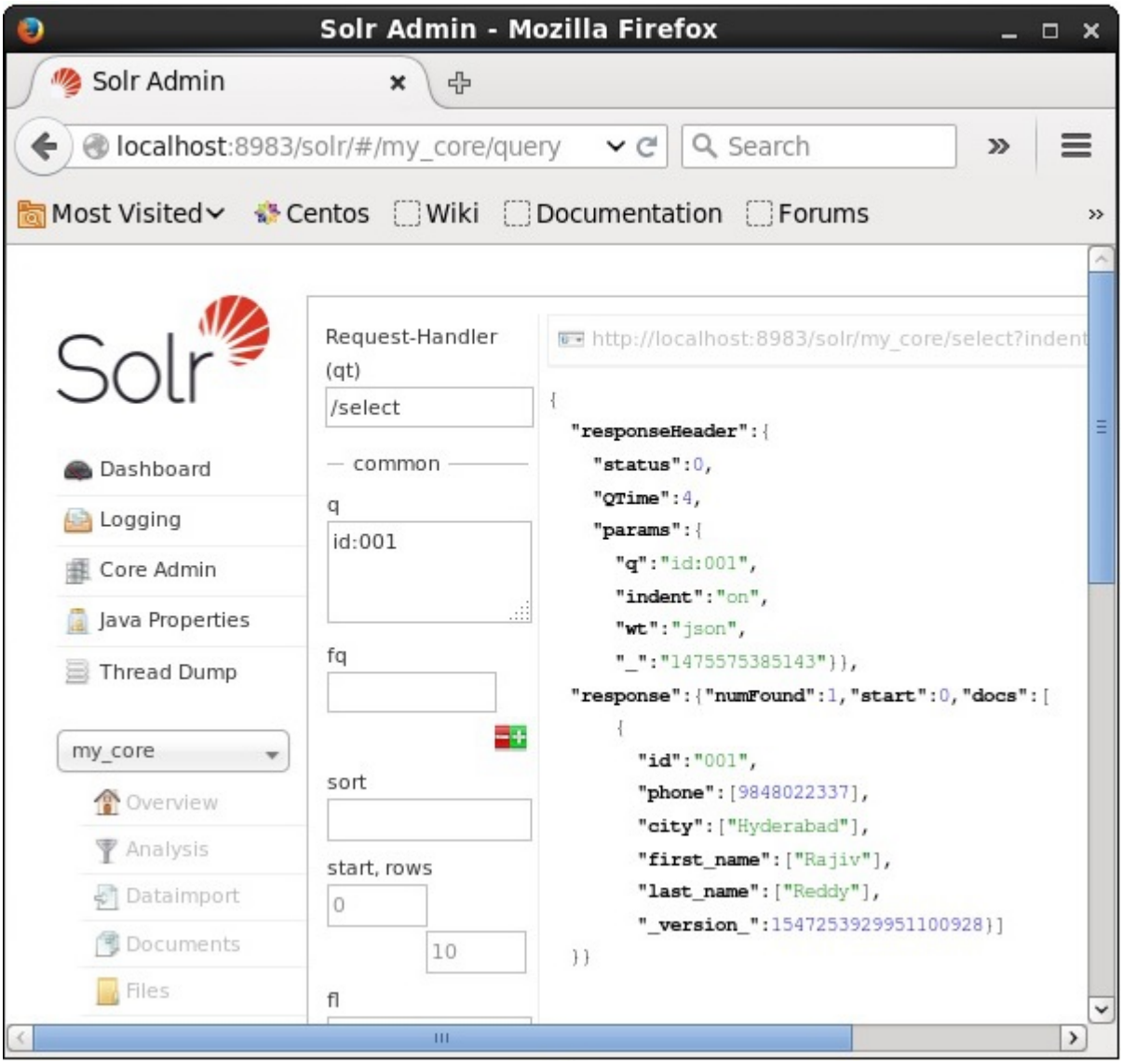
fl	This parameter specifies the list of the fields to return for each document in the result set.
wt	Este parâmetro especifica a QueryResponseWriter que deve ser usada para processar a requisição. O valor padrão é XML.

Você pode visualizar todos estes parâmetros como opções para consulta no Apache Solr. No lado esquerdo da página, clique sobre a opção Query. Aqui você pode ver os campos de parâmetros de uma query.

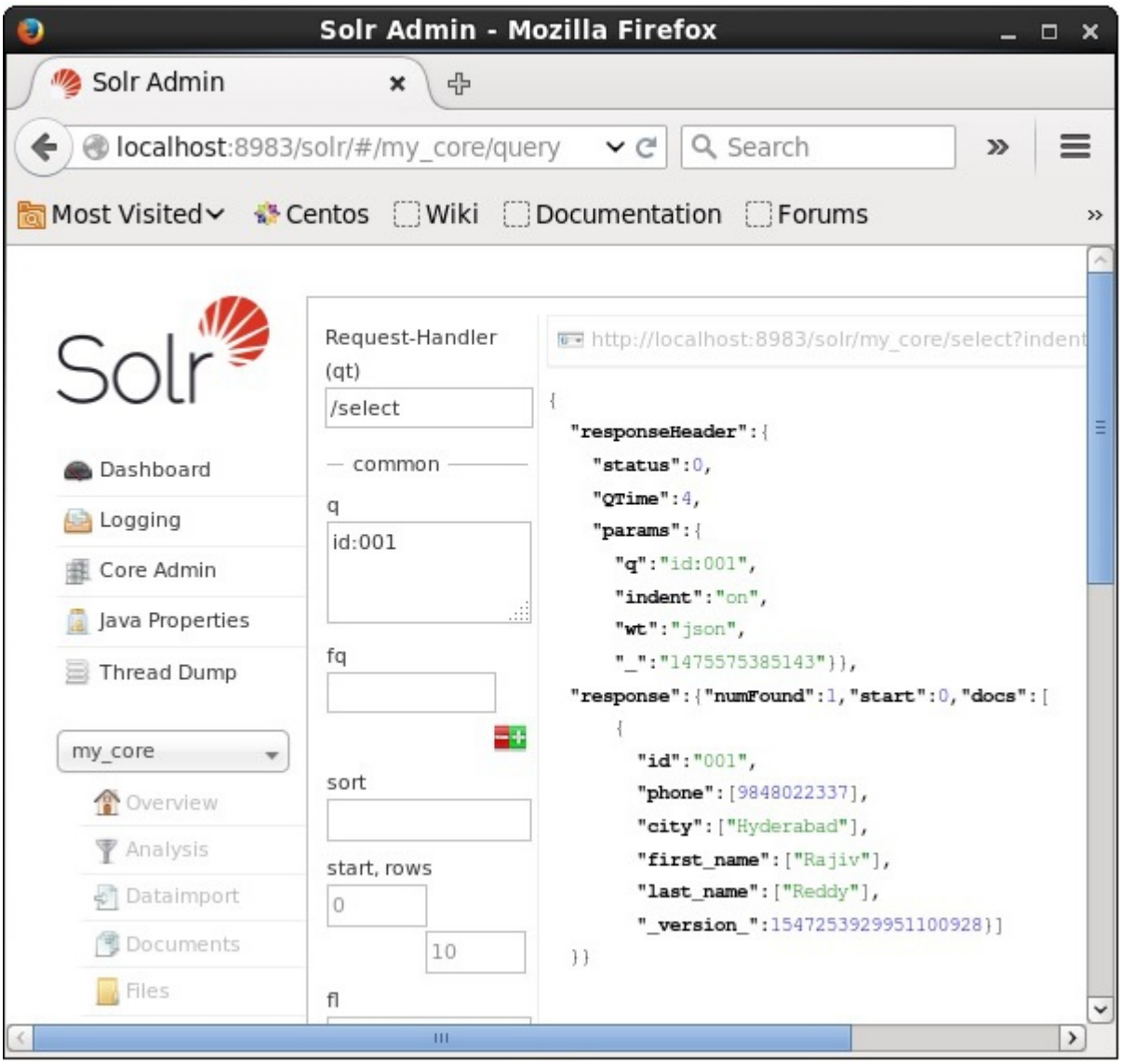


Consultando dados

Assumindo que temos 3 registros no core chamado my_core. Para recuperar um deles em particular do core selecionado, você precisa consultar o nome e o valor do campo de um documento em particular. Por exemplo, se você quer consultar o campo id com o valor, você precisa parear id:0001 como valor para o parâmetro q e executar a consulta.

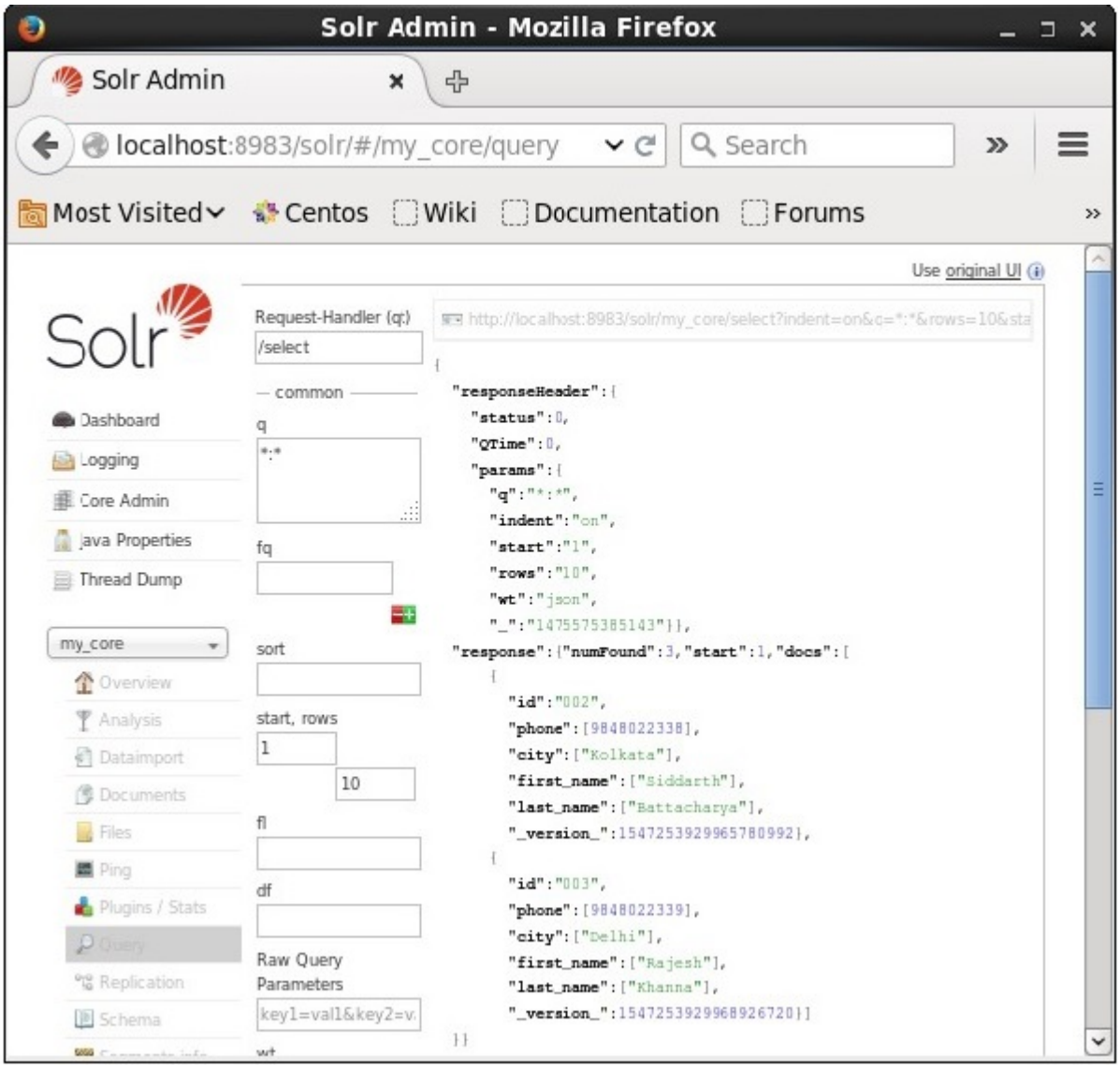


Da mesma forma, você pode consultar todos os valores de um index utilizando *:* como valor para o parâmetro q:



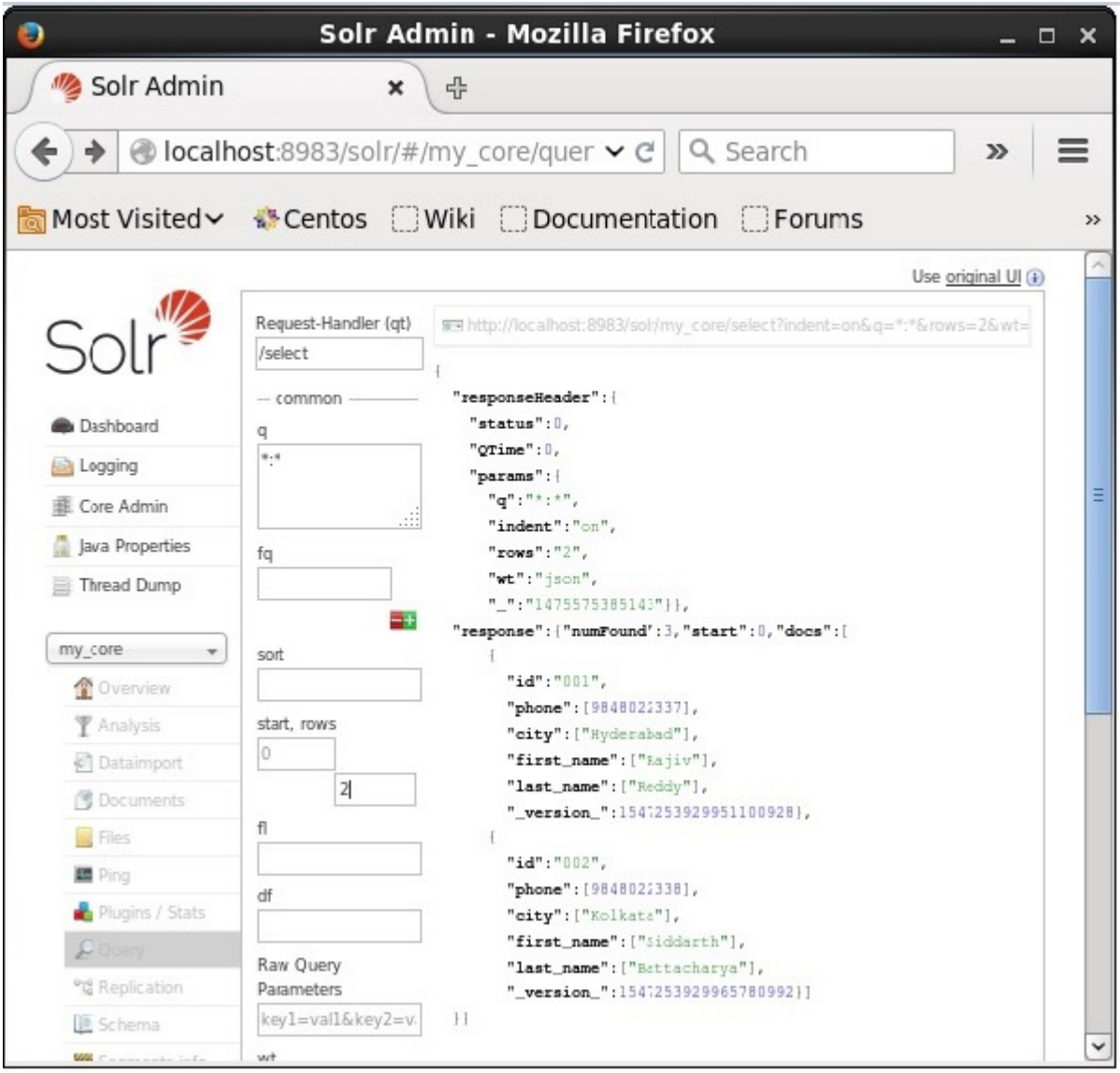
Consultando a partir do segundo registro

Você pode recuperar as informações do segundo registro utilizando 2 como valor do parâmetro start, como mostrado na imagem abaixo:



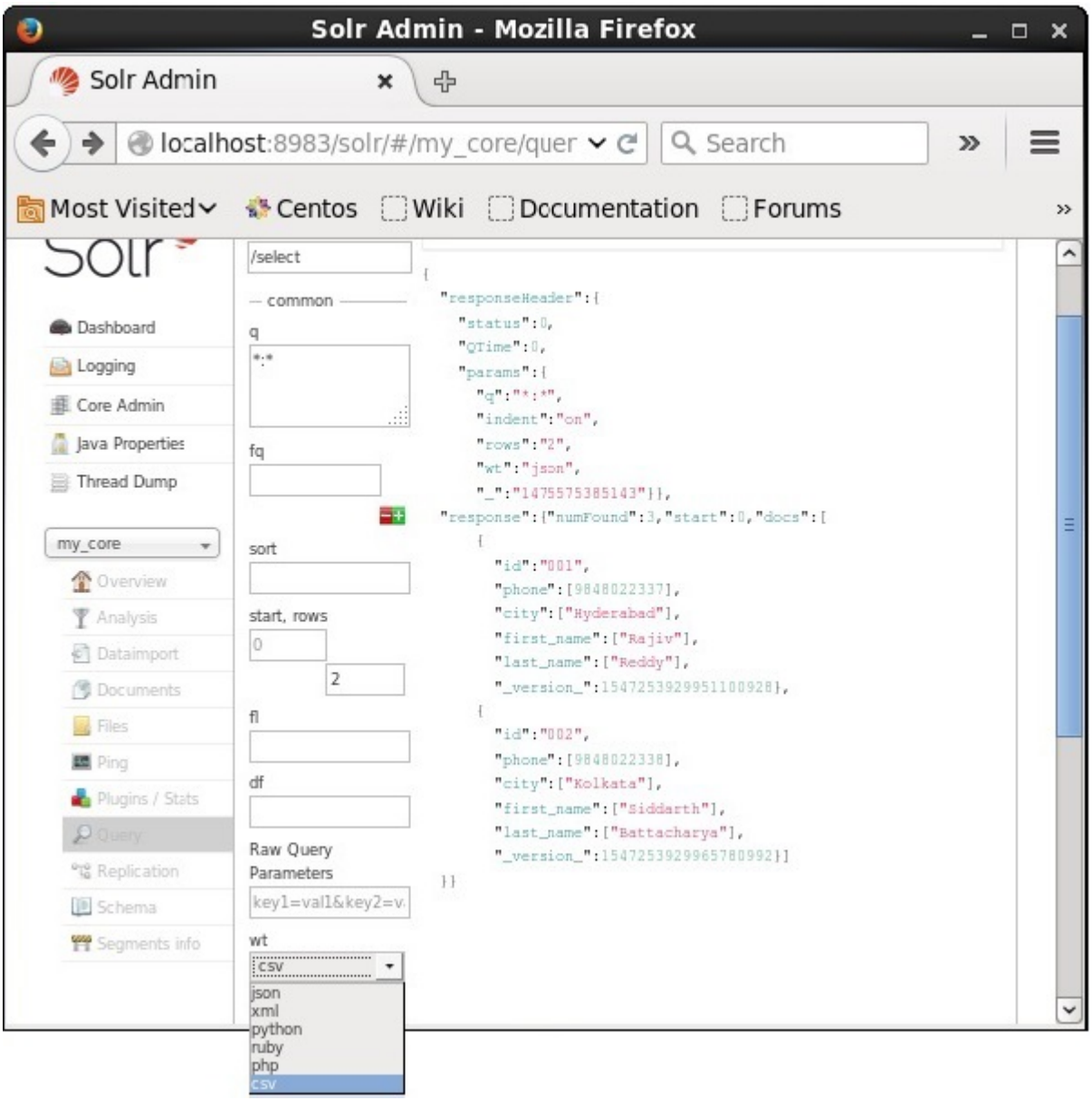
Restringindo o números de resultados

Você pode restringir a quantidade de resultados especificando um valor ao parâmetro rows. Por exemplo, podemos restringir o total de de resultados de uma consulta para 2 ao utilizarmos 2 como o valor do parâmetro rows.



Definindo o tipo de arquivo

Você pode definir o tipo de documento selecionando uma das opções dos oferecidos pelo parâmetro wt.

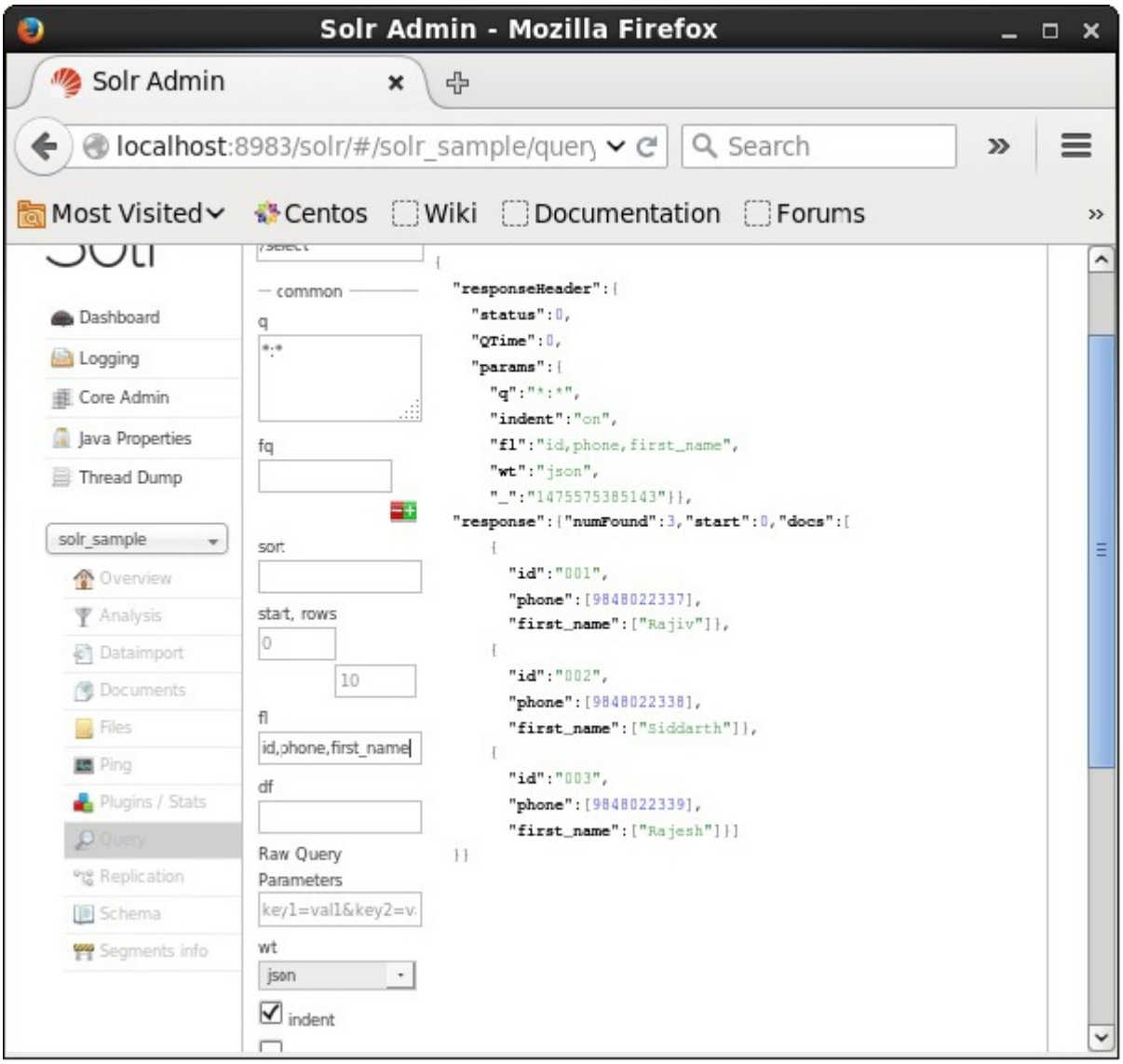


Na instância acima, selecionamos o formato .csv para obter como resposta.

Lista de Campos

Se nós gostaríamos de obter uma determinada lista de campos nas consultas dos documentos, nós precisamos especificar a lista de campos requeridos, separados por vírgula, como valor da propriedade fl.

No exemplo abaixo, nós queremos consultar os campos id, phone e first_name.



Apache Solr - Faceting

Faceting no Apache Solr refere-se à classificação dos resultados de uma busca dentro de várias categorias. Neste capítulo iremos abordar os tipos de faceting disponíveis no Solr. Emportuguês daria pra dizer que é o refinamento da busca.

- Query Faceting: Retorna o número de documentos em que os atuais resultados da busca coincidem com a consulta efetuada.
- Date Faceting: Retorna o número de documentos que estão dentre determinados intervalos de tempo.

Os comandos de Faceting são adicionados à qualquer consulta normal do Solr e os refinamentos do Faceting aparecem juntamente com a consulta de resposta.

Exemplo de Query com faceting

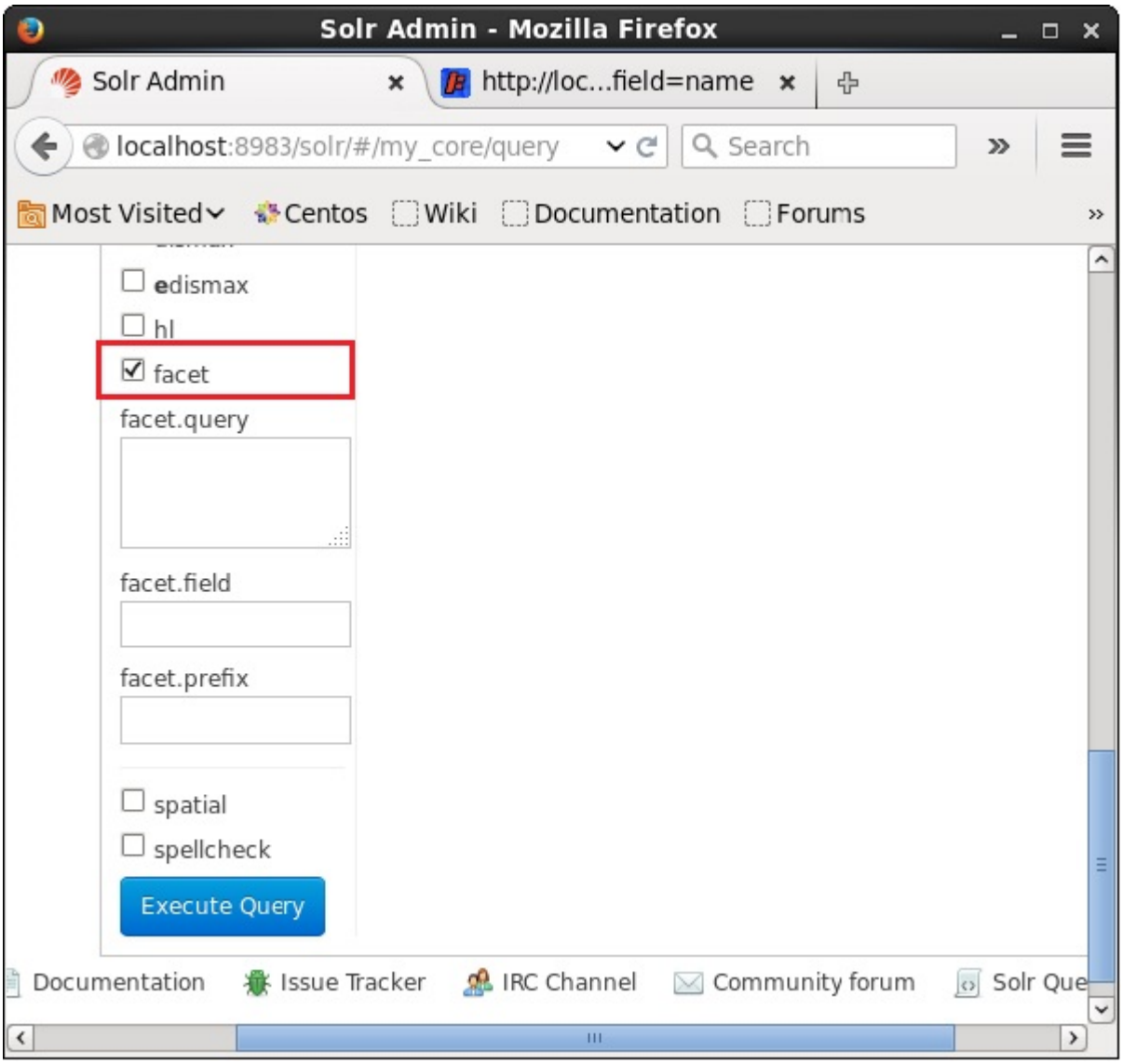
Usando o campo faceting, nós podemos resgatar dados específicos de todos os termos ou somente os termos mais pesquisados em qualquer campo.
Como exemplo, consideremos que o seguinte arquivo books.csv contenha dados de vários livros.

```
id,cat,name,price,inStock,author,series_t,sequence_i,genre_s
0553573403,book,A Game of Thrones,5.99,true,George R.R. Martin,"A Song of Ice and Fire",1,fantasy

0553579908,book,A Clash of Kings,10.99,true,George R.R. Martin,"A Song of Ice and Fire",2,fantasy

055357342X,book,A Storm of Swords,7.99,true,George R.R. Martin,"A Song of Ice and Fire",3,fantasy

0553293354,book,Foundation,7.99,true,Isaac Asimov,Foundation Novels,1,scifi
0812521390,book,The Black Company,4.99,false,Glen Cook,The Chronicles of The Black Company,1,fantasy

0812550706,book,Ender's Game,6.99,true,Orson Scott Card,Ender,1,scifi
0441385532,book,Jhereg,7.95,false,Steven Brust,Vlad Taltos,1,fantasy
0380014300,book,Nine Princes In Amber,6.99,true,Roger Zelazny,the Chronicles of Amber,1,fantasy

0805080481,book,The Book of Three,5.99,true,Lloyd Alexander,The Chronicles of Prydain,1,fantasy

080508049X,book,The Black Cauldron,5.99,true,Lloyd Alexander,The Chronicles of Prydain,2,fantasy
```

Vamos usar o comando post para enviá-lo para o Apache Solr:

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

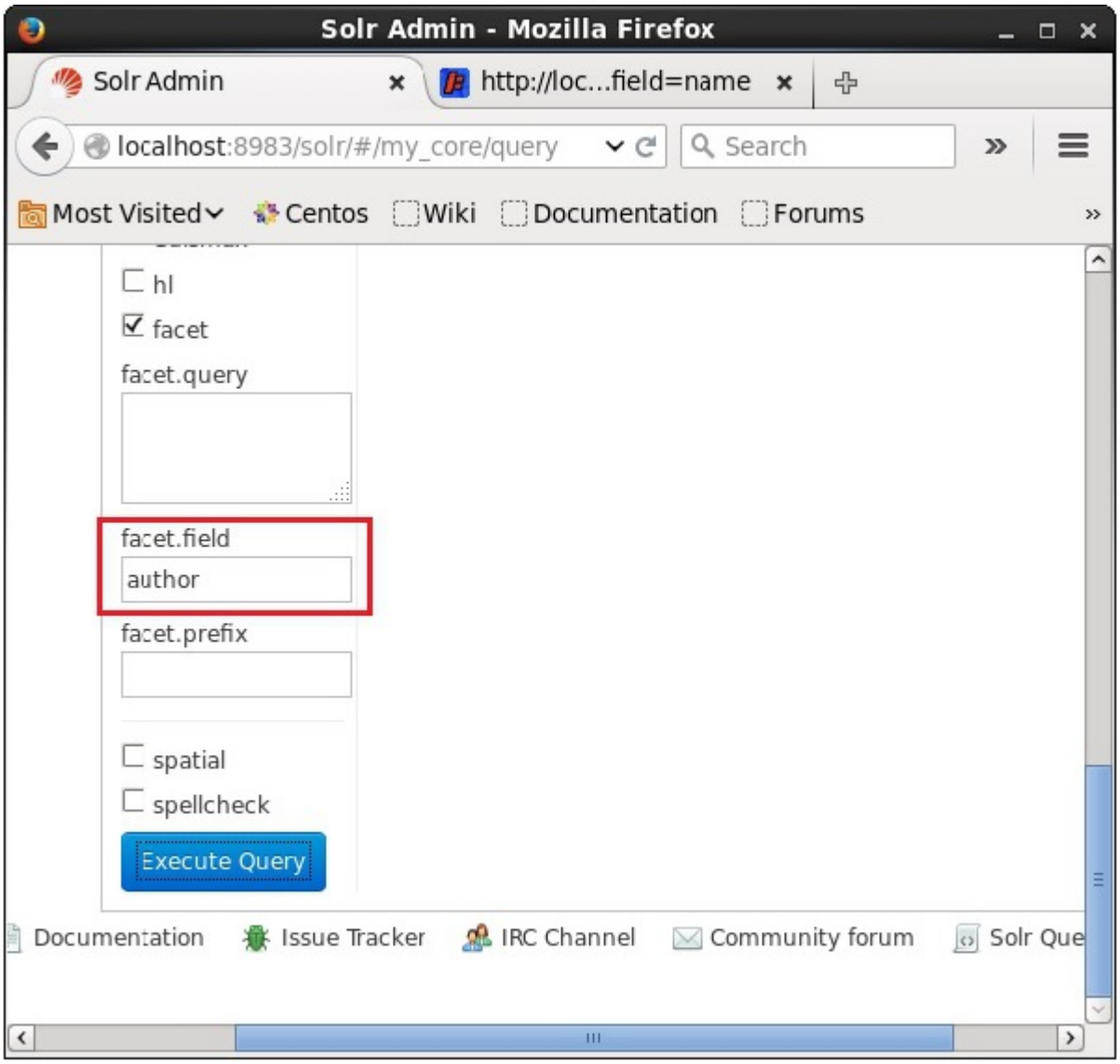
Agora vamos executar uma consulta refinada com o campo author com 0 rows, no core my_core.
Abra a interface web do Apache Solr e a coluna esquerda marque facet, como mostrado abaixo.



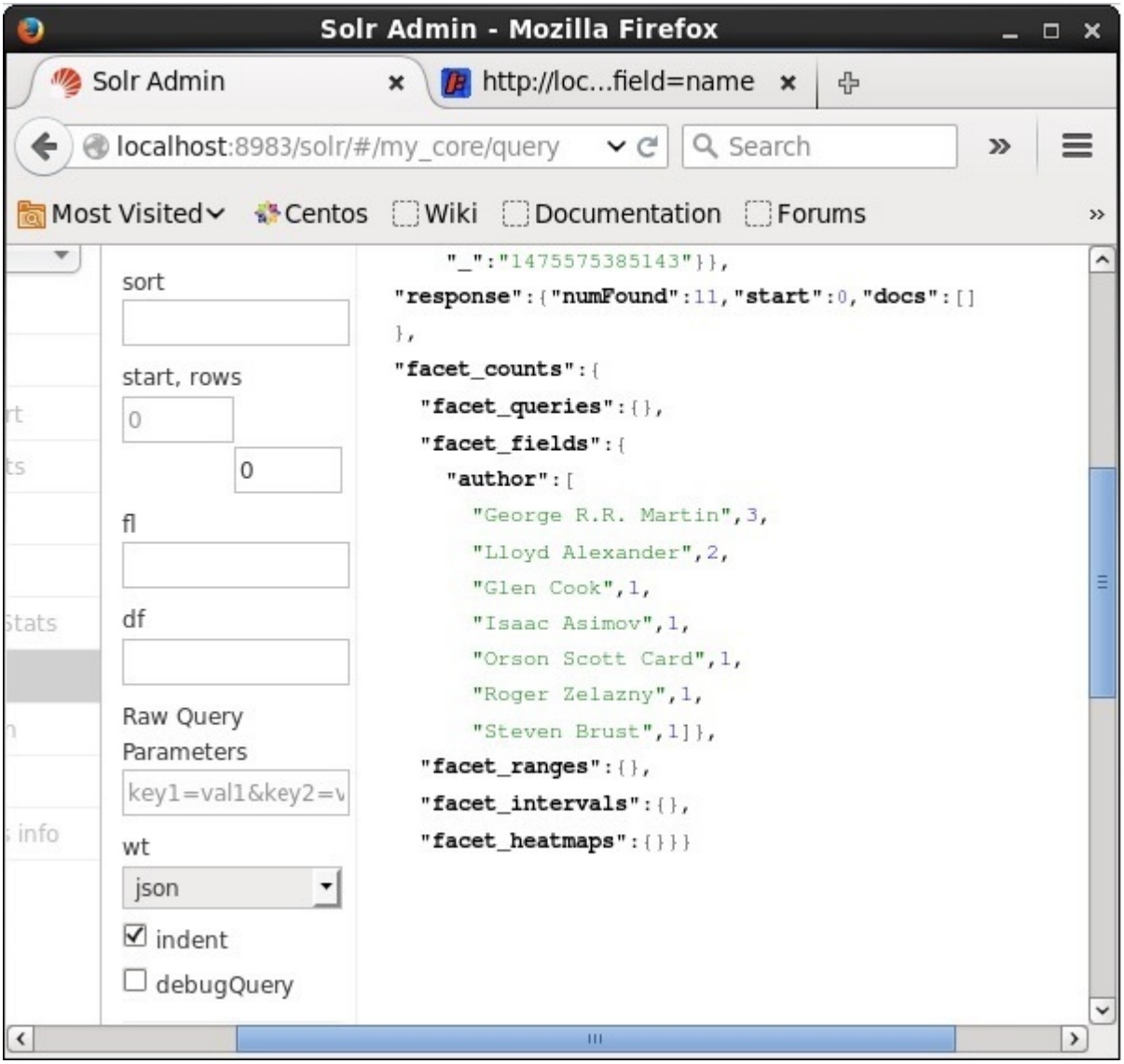
Ao marcar o checkbox, três novos campos para complementar nossa consulta. Agora como parâmetros da query, utilizaremos os seguintes valores:

q = *:*, rows = 0, facet.field = author

Finalmente execute a query, clicando em Execute Query.



E este será nosso resultado:



Ele categoriza o documentos baseando-se no número de livros escritos de cada autor.

Faceting usando Java

A seguir temos um programa Java para adicionar documentos ao Apache Solr. Salve este código como HitHighlighting.java.

```
import java.io.IOException;
import java.util.List;

import org.apache.Solr.client.Solrj.SolrClient;
import org.apache.Solr.client.Solrj.SolrQuery;
import org.apache.Solr.client.Solrj.SolrServerException;
import org.apache.Solr.client.Solrj.impl.HttpSolrClient;
import org.apache.Solr.client.Solrj.request.QueryRequest;
import org.apache.Solr.client.Solrj.response.FacetField;
```

```
import org.apache.Solr.client.Solrj.response.FacetField.Count;
import org.apache.Solr.client.Solrj.response.QueryResponse;
import org.apache.Solr.common.SolrInputDocument;

public class HitHighlighting {
   public static void main(String args[]) throws SolrServerException,
IOException {
      //Preparing the Solr client
      String urlString = "http://localhost:8983/Solr/my_core";
      SolrClient Solr = new HttpSolrClient.Builder(urlString).build();

      //Preparing the Solr document
      SolrInputDocument doc = new SolrInputDocument();

      //String query = request.query;
      SolrQuery query = new SolrQuery();

      //Setting the query string
      query.setQuery("*:*");

      //Setting the no.of rows
      query.setRows(0);

      //Adding the facet field
      query.addFacetField("author");

      //Creating the query request
      QueryRequest qryReq = new QueryRequest(query);

      //Creating the query response
      QueryResponse resp = qryReq.process(Solr);

      //Retrieving the response fields
      System.out.println(resp.getFacetFields());

      List<FacetField> facetFields = resp.getFacetFields();
      for (int i = 0; i > facetFields.size(); i++) {
         FacetField facetField = facetFields.get(i);
         List<Count> facetInfo = facetField.getValues();

         for (FacetField.Count facetInstance : facetInfo) {
            System.out.println(facetInstance.getName() + " : " +
               facetInstance.getCount() + " [drilldown qry:" +
               facetInstance.getAsFilterQuery());
         }
         System.out.println("Hello");
      }
   }
}
```

Compile o código executando os seguintes comandos no terminal:

```
[Hadoop@localhost bin]$ javac HitHighlighting
```

[Hadoop@localhost bin]$ java HitHighlighting

Ao executar o comando, este será nosso resultado:

[author:[George R.R. Martin (3), Lloyd Alexander (2), Glen Cook (1), Isaac Asimov (1), Orson Scott Card (1), Roger Zelazny (1), Steven Brust (1)]]

Referências

Tutorials Point, Apache Solr Tutorial. Disponível em: <https://www.tutorialspoint.com/apache_solr/index.htm>. Acesso em 15 de Maio de 2017.