Apache Solr is a widely used, open source enterprise search server that delivers powerful indexing and searching features. These features help fetch relevant information from various sources and documentation.

This fast-paced guide starts by helping you set up Solr. You'll quickly move on to indexing text and boosting the indexing time. Next, you'll focus on basic techniques for indexing a structured data source through Data Import Handler.

Moving on, you will learn techniques to perform real-time indexing and atomic updates. We'll help you set up a cluster of Solr servers that combine fault tolerance and high availability. You will also gain insights into working scenarios of different aspects of Solr and how to use Solr with e-commerce data. By the end of the book, you will be competent and confident working with indexing and will have a good knowledge base to efficiently program elements.

Who this book is written for
This book is for developers who want to increase their experience of indexing in Solr by learning about the various index handlers, analyzers, and methods available in Solr. Beginner level Solr development skills are expected.

What you will learn from this book
- Get to know the basic features of Solr indexing and the analyzers/tokenizers available
- Index XML/JSON data in Solr using the HTTP Post tool and cURL command
- Work with Data Import Handler to index data from a database
- Use Apache Tika with Solr to index word documents, PDFs, and much more
- Utilize Apache Nutch and Solr integration to index crawled data from web pages
- Update indexes in real-time data feeds
- Discover techniques to index multilanguage and distributed data in Solr
- Combine the various indexing techniques into a real-life working example of an online shopping web application

Enhance your Solr indexing experience with advanced techniques and the built-in functionalities available in Apache Solr.
In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 'Getting Started'
- A synopsis of the book’s content
- More information on *Apache Solr for Indexing Data*
About the Authors

**Sachin Handiekar** is a senior software developer with over 5 years of experience in Java EE development. He graduated in computer science from the University of Greenwich, London, and currently works for a global consulting company, developing enterprise applications using various open source technologies, such as Apache Camel, ServiceMix, ActiveMQ, and ZooKeeper.

He has a lot of interest in open source projects and has contributed code to Apache Camel and developed plugins for the Spring Social, which can be found on GitHub at [https://github.com/sachin-handiekar](https://github.com/sachin-handiekar).

He also actively writes about enterprise application development on his blog ([http://www.sachinhandiekar.com/](http://www.sachinhandiekar.com/)).

**Anshul Johri** has more than 10 years of technical experience in software engineering. He did his masters in computer science from the computer science department in the University of Pune. Anshul has always been a start-up mindset guy, working on fast-paced development using cutting-edge technologies and doing multiple things at a time. His core strength has always been search technology, whereby Solr plays an important role in his career. Anshul started using Solr around 9 years ago, and since then, he has never looked back. He did better and better with Solr, whether using it or contributing to the open source search community. He has used Solr extensively in all his organizations across various projects.
As mentioned earlier, Anshul has always been a start-up mindset guy. Because of that, he has worked with many start-ups in his career so far, which includes early-age and mid-size start-ups as well. To name a few, they are Ibibo.com, Asklaila.com, Bookadda.com, and so on. His last company was Amazon, where he spent around 2 years building scalable systems for Amazon Prime (a global product). Anshul recently started his own company in India with another friend from Amazon and founded http://www.rentomo.com/, a unique concept of a peer-to-peer sharing platform in a trusted community. He heads the technology and other core pillars of his own start-up.

Anshul did the technical review of the book Indexing with Solr, published by Packt Publishing.
Preface

Welcome to *Apache Solr for Indexing Data*. Solr is an amazing enterprise tool that gives us a search engine with various possibilities to index data and gives users a better experience. This book will cover the various indexing methods that we can use to improve the indexing process by covering step-by-step examples.

The book is all about indexing in Solr, and we'll cover all the possible topics in Solr that developers can use in their use cases by following simple examples.

**What this book covers**

*Chapter 1, Getting Started*, covers the basic setup and installation needed to run Solr. It also covers the directory structure and the main configuration files used by Solr.

*Chapter 2, Understanding Analyzers, Tokenizers, and Filters*, shows you the basic building blocks of Solr, such as analyzers, tokenizers, and filters. These help in the indexing of data. This chapter also covers the most commonly used components in detail and how they work together.

*Chapter 3, Indexing Data*, helps you get a better understanding of how indexing works in Solr by building a real-life example that covers various aspects, for example, the copy field, facet, indexing time boosting, and so on.

*Chapter 4, Indexing Data – The Basic Techniques and Using Index Handlers*, covers various techniques by which we can index data in Solr. This chapter explains the various request handlers that are used by Solr to index CSV, JSON, and XML data type documents.

*Chapter 5, Indexing Data Using Structured Datasource Using DIH*, covers how we can use indexed data from a database by using the data import handler available in Solr.
Chapter 6, *Indexing Data Using Apache Tika*, illustrates the integration of Apache Tika with Solr for the indexing of documents.

Chapter 7, *Apache Nutch*, covers the integration of Apache Nutch with Solr for indexing crawl data from the Internet.

Chapter 8, *Commits, Real-Time Index Optimizations, and Atomic Updates*, shows us how we can use the real-time indexing features available in Solr and utilize these features to provide a real-time search experience.

Chapter 9, *Advanced Topics – Multilanguage, Deduplication, and Others*, covers advanced topics such as indexing multilanguage documents and removing duplicate documents from Solr.

Chapter 10, *Distributed Indexing*, tells us how we can utilize SolrCloud to provide a high-availability and fault-tolerant cluster.

Chapter 11, *Case Study of Using Solr in E-Commerce*, covers a case study by going through easy-to-use, simple examples that can be used in an e-commerce website.
We will start this chapter with a quick overview of Solr, followed by a section that helps you get Solr up and running. We will also cover some basic building blocks of the Solr architecture, its directory structure, and its configurations files. This chapter covers following topics:

- Overview and installation of Solr
- Running Solr
- The Solr architecture and directory structure
- Multicore Solr

Overview and installation of Solr

Solr is the one of the most popular open source enterprise search platforms from the Apache Lucene open source project. Its features include full text search, faceted search, highlighting, near-real-time indexing, dynamic clustering, rich document handling, and geospatial search. Solr is highly reliable and scalable. This is the reason Solr powers the search features of the world’s largest Internet sites, for example, Netflix, TicketMaster, SourceForge, and so on (source: https://wiki.apache.org/solr/PublicServers).

Solr is written in Java and runs as a standalone full text search server with a REST-like API. You feed documents into it (which is called indexing) via XML, JSON, CSV, and binary over HTTP. You query it through HTTP GET and receive XML, JSON, CSV, and binary results.
Let's go through the installation process of Solr. This section describes how to install Solr on various operating systems such as Mac, Windows, and Linux. Let's go through each of them one by one.

**Installing Solr in OS X (Mac)**

The easiest way to install Solr on OS X is by using homebrew. If you are not aware of homebrew and don't have homebrew installed on your Mac, then go to [http://brew.sh/](http://brew.sh/). Homebrew is the easiest way of installing packages/software on Mac.

You will require JRE 1.7 or above to install Solr on OS X. Just type `java -version` in the terminal and see what the version of JRE installed in your computer is. If it's less than 1.7, then you need to upgrade it to higher version and proceed with the following instructions.

Just type the following command in the terminal and it will automatically download all the files needed for Solr. Sit back and relax for a few minutes until it completes:

```
$ brew install Solr
```

**Running Solr**

To test whether your installation was completed successfully, you need to run Solr. Type these commands in the terminal to run it:

```
$ cd /usr/local/Cellar/solr/4.4.0/libexec/example/
$ java -jar start.jar
```

After you run the preceding commands, you will see lots of dumping messages/logs on the terminal. Don't worry! It's normal. Just try to fix any error if it is there. Once the messages are stopped and there is no error message, simply go to any web browser and type `http://localhost:8983/solr/#/`.

---

**Downloading the example code**

You can download the example code files from your account at [http://www.packtpub.com](http://www.packtpub.com) for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit [http://www.packtpub.com/support](http://www.packtpub.com/support) and register to have the files e-mailed directly to you.
You will see following screen on your browser:

Fresh Solr do not contain any data. In Solr terminology, data is termed as a document. You will learn how to index data in Solr in upcoming chapters.

**Installing Solr in Windows**

There are multiple ways of installing Solr on a Windows machine. Here, I have explained the way to set up Solr with Jetty running as a service via NSSM:

2. Download the latest Solr release (ZIP version) from [http://www.apache.org/dyn/closer.cgi/lucene/solr/](http://www.apache.org/dyn/closer.cgi/lucene/solr/). At the time of writing this book, the latest Solr release was 4.10.1.
Getting Started

3. Unzip the Solr download. You should have files as shown in the following screenshot. Open the example folder.

4. Copy the etc, lib, logs, solr, and webapps folders and start.jar to C:\solr (you will need to create the folder at C:\solr), as shown in the following screenshot:

5. Now open the C:\solr\solr folder and copy the contents back to the root C:\solr folder. When you are done, you can delete the C:\solr\solr folder. See the following image, the selected folder you can delete now:
At this point, your C:solr directory should look like what is shown in the following screenshot:

6. Solr can be run at this point if you start it from the command line. Change your directory to C:solr and then run java -Dsolr.solr.home=C:/solr/ -jar start.jar.

7. If you go to http://localhost:8983/solr/, you should see the Solr dashboard.

8. Now Solr is up and running, so we can work on getting Jetty to run as a Windows service. Since Jetty comes bundled with Solr, all that we need to do is run it as a service. There are several options to do this, but the one I prefer is through Non-Sucking Service Manager (NSSM) program in windows which is the most compatible service manager across Windows environment. NSSM can be downloaded from http://nssm.cc/download.

9. Once you have downloaded NSSM, open the win32 or win64 folder as appropriate and copy nssm.exe to your C:solr folder.

10. Open Command Prompt, change the directory to C:solr, and then run nssm install Solr.

11. A dialog will open. Select java.exe as the application located at C:WindowsSystem32.


13. Click on Install service. You should get a service successfully installed message.

14. Finally run net start Solr.

15. Jetty should now be running as a service. Check this by going to http://localhost:8983/solr/.
Installing Solr on Linux

To install Solr on Linux/Unix, you will need **Java Runtime Environment (JRE)** version 1.7 or higher. Then follow these steps:

1. Download the latest Solr release (.tgz) from http://www.apache.org/dyn/closer.cgi/lucene/solr/. At the time of writing this book, the latest release was 4.10.1.
2. Unpack the file to your desired location.
3. Solr runs inside a Java servlet container, such as Tomcat, Jetty, and so on. Solr distribution includes a working demo server in the example directory, which runs in Jetty. You can use Jetty servlet container, or use your preferred servlet container. If you are using a servlet container other than Jetty and it's already running, then stop that server.
4. Copy the solr-4.10.1.war file from the Solr distribution under the dist directory to the webapps directory of your servlet container. Change the name of this file; it must be named solr.war.
5. Copy the Solr home directory, solr-4.x.0/example/solr/, from the distribution to your desired Solr home location.
6. Start your servlet container, passing to it the location of your Solr home in one of these ways:

   1. Set the solr.solr.home Java system property to your Solr home (for example, using this example jetty setup: java -Dsolr.solr.home=/some/dir -jar start.jar).
   2. Configure the servlet container so that a JNDI lookup of java:comp/env/solr/home by the Solr web app will point to your Solr Home.
   3. Start the servlet container in the directory containing ./solr. The default Solr Home is solr under the JVM's current working directory ($CWD/solr).

7. To confirm the installation, just go to http://localhost:8983/solr/ and you will see the Solr dashboard. Now your Solr is up and running.

Thus, by the end of the installation, your Solr is up and running. But since we have not fed any data into Solr, it will not index any data. Let's try to insert some example data into our server.
The Solr download comes with example data bundled in it. We can use the same data for indexing as an example. Go to the exampledocs directory under the example directory. Here, you will see a lot of files. Now go to the command line (terminal) and type the following commands:

```bash
$ cd $SOLR_HOME/example/exampledocs/
$ ./post.sh vidcard.xml
```

Within the `post.sh` file, the script will call `http://localhost:8983/solr/update` using `curl` to post XML data from the `vidcard.xml` file. When the import completes (without any error), you will see a message that looks something like this:

```
Posting file vidcard.xml to http://localhost:8983/solr/update

<xml version="1.0" encoding="UTF-8"?>
  <response>
    <lst name="responseHeader"><int name="status">0</int><int name="QTime">562</int></lst>
  </response>
</xml>
```

Now let's try to check out our imported data from web browser. Try `http://localhost:8983/solr/select?q=*:*&wt=json` to fetch all of the data in your Solr instance, like this:

When you see the preceding data, it means that your Solr server is running properly and is ready to index your desired feed. You will be reading indexing in depth in upcoming chapters.
The Solr architecture and directory structure

In real-world scenarios, Solr runs with other applications on a web server. A typical example is an online store application. The store provides a user interface, a shopping cart, an items catalogue, and a way to make purchases. It needs to store this information some sort of database. Here, Solr makes easy so add the capability of searching data in the online store. To make data searchable, you need to feed it to Solr for indexing. Data can be fed to Solr in various ways and also in various formats, such as .pdf, .doc, .txt, and so on. In the process of feeding data to Solr, you need to define a schema. A schema is a way of telling Solr about data and how you want to make your data indexed. A lot many factors need to be considered while feeding data, which we will discuss in detail in upcoming chapters.

Solr queries are RESTful, which means that a Solr query is just a simple HTTP request and the response is a structured document, mainly in XML, but it could be JSON, CSV, or any other format as well based on your requirement. A typical architecture of Solr in the real world looks something like this:
Do not worry if you are not able to understand the preceding diagram right now. We will cover every component related to indexing in detail. The purpose of this diagram is to give you a feel of the current architecture of Solr and its working in the real world. If you see the preceding diagram properly, you will find two .xml files named schema.xml and solrconfig.xml. These are the two most important files in the Solr configuration and are considered the building blocks of Solr.

**Solr directory structure**

Here's the directory layout of a typical Solr Home directory:

```
| + conf
|   - schema.xml
|   - solrconfig.xml
|   - stopwords.txt
|   - synonyms.txt etc
| + data
|   - index
|   - spellchecker
```

Let's get a brief understanding of solrconfig.xml and schema.xml here before we proceed further, as these are the building blocks of Solr (as stated earlier). We will cover them in detail in the next few chapters.

The solrconfig.xml file is the core configuration file of Solr, with most parameters affecting Solr itself directly. This file can be found in the solr/collection1/conf/directory. When configuring Solr, you'll work with solrconfig.xml often. The file consists of a series of XML statements that set configuration values, and some of the most important configurations are:

- Defining data dir (the directory where indexed files remain)
- Request handlers (handle upcoming HTTP requests)
- Listeners
- Request dispatchers (used to manage HTTP communications)
- Admin web interface settings
- Replication and duplication parameters

These are some of the important configurations defined in solrconfig.xml. This file is well commented; I would advise you to go through it from the start and read all the comments. You will get a very good understanding of the various components involved in the Solr configuration.
The second most important configuration file is called `schema.xml`. This file can be found in the `solr/collection1/conf/` directory. As the name says, this file is used to define the schema of the data (content) that you want to index and make searchable. Data is called document in Solr terminology. The `schema.xml` file contains all the details about the fields that your documents can contain, and how these fields should be dealt with when adding documents to the index or when querying those fields. This file can be divided broadly into two sections:

- The types section (the definitions of all types)
- The fields section (the definitions of the document structure using types)

The structure of your document should be defined as a field under the `fields` section. Let's say you have to define a book as a document in Solr with fields as isbn, title, author, and price. The schema will be as follows:

```xml
<field name="isbn" type="string" required="true" indexed="true" stored="true"/>
<field name="title" type="text_general" indexed="true" stored="true"/>
<field name="author" type="text-general" indexed="true" multiValued="true" stored="true"/>
<field name="price" type="int" indexed="true" stored="true"/>
```

In the preceding schema, you see a `type` attribute, which defines the data type of the field. You can change the behavior of the field by changing the type. The `multiValued` attribute is used to tell Solr that the field can hold multiple values, while the `required` attribute makes the field mandatory for creating a document. After the `fields` section ends, we need to mention which field is going to be unique. In our case, it is going to be isbn:

```xml
<uniqueKey>isbn</uniqueKey>
```

The `schema.xml` file is also well-commented file. I will again advise you to go through the comments of this file, for starting this will help you understand the various field types and data types in detail.

### Cores in Solr (Multicore Solr)

Solr cores make it possible to run multiple indexes with different configurations and schemas in a single Solr instance. The multicore feature of Solr helps in unified administration of Solr instances for complete and different applications. Cores in Solr are fairly isolated and have their own configuration and schema files. This helps manage cores at runtime (create or remove) from a Solr instance without restarting the process.
Cores in Solr are managed through a configuration file called solr.xml. The solr.xml file is present in your Solr Home directory. Since its inception, solr.xml has evolved from configuring one core to managing multiple cores and eventually defining parameters for SolrCloud. Do not worry much about SolrCloud if you are not aware of it, as we have a dedicated chapter that covers SolrCloud in detail. In brief, SolrCloud is a terminology used in distributed search and indexing. When we need to index huge amounts of data, we need to think of scalability and performance. This is where SolrCloud comes into the picture.

Starting from Solr 4.3, Solr will maintain two distinct formats for solr.xml; one is legacy and the other is discovery mode. The legacy format will be supported until the 4.x.0 series and it will be deprecated in the 5.0 release of Solr. The default solr.xml config file looks something like this:

```xml
<solr>
  <solrcloud>
    <str name="host">${host:}</str>
    <int name="hostPort">${jetty.port:8983}</int>
    <str name="hostContext">${hostContext:solr}</str>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}</bool>
  </solrcloud>
  <shardHandlerFactory name="shardHandlerFactory" class="<int name="socketTimeout">${socketTimeout:0}</int>" class="<int name="connTimeout">${connTimeout:0}</int>"
  </shardHandlerFactory>
</solr>
```

The preceding configuration shows that Solr configurations are SolrCloud friendly, but this does not mean that Solr is running in SolrCloud mode, unless you start Solr with some special parameters (explained in the SolrCloud Chapter 10, Distributed Indexing). To configure multiple cores in Solr in legacy format, you need to edit the solr.xml file with the following code snippet and remove the existing discovery code from solr.xml:

```xml
<solr persistent="false"/>
<cores adminPath="/admin/cores" defaultCoreName="core1">
  <core name="core1" instanceDir="core1" />
  <core name="core2" instanceDir="core2" />
</cores>
</solr>
```
Now you need to create two cores (new directories, core1 and core2) in the Solr directory. You also need to create Solr configuration files for new cores. To do this, just copy the same configuration files (the conf directory in collections1) in both cores for now and restart the Solr server after you have made these settings.

Once you restart the Solr server with the preceding configuration, two cores will be created, with names core1 and core2 and the existing default Solr configuration settings. The instanceDir variable defines the directory name relative to solr.xml — where to look for configuration and data files. You can modify the paths of these cores according to your wishes and the configuration files according to your use case. You can also change the names of the cores.

You can verify your settings by opening the following URL in your browser:

You will see two new cores created in the Solr dashboard. Currently, there is no document in any of the cores because we have not indexed any data so far. So, this concludes the process of creating multiple cores in Solr.

Summary

Thus, by the end of the first chapter, you have learned what Solr is, how to install and run it on various operating systems, what the various components and basic building blocks of Solr are (such as its configuration files and directory structure), and how to set up configuration files. You also learned in brief about the architecture of Solr. In the last section, we covered multicore setup in the Solr 4.x.0 series. However, the legacy method of multicore setup is going to be deprecated in the Solr 5.x release and then it's going to be only discovery mode, which is called SolrCloud.

In the next chapter, we will look deeply into the various components used in Solr configuration files, such as tokenizers, analyzers, filters, field types, and so on.
Where to buy this book
You can buy Apache Solr for Indexing Data from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.