SolrTutorial.com

Basic Solr Concepts

In this document, we'll cover the basics of what you need to know about Solr in order to use it.

Indexing

Solr is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead.

This is like retrieving pages in a book related to a keyword by scanning the index at the back of a book, as opposed to searching every word of every page of the book.

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

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

How Solr represents data

In Solr, a **Document** is the unit of search and index.

An index consists of one or more Documents, and a Document consists of one or more Fields.

In database terminology, a Document corresponds to a table row, and a Field corresponds to a table column.

Schema

Before adding documents to Solr, you need to specify the schema, represented in a file called **schema.xml**. *It is not advisable to change the schema after documents have been added to the index.*

The schema declares:

- what kinds of fields there are

- which field should be used as the unique/primary key
- which fields are required
- how to index and search each field

Field Types

In Solr, every field has a **type**. Solr expands the variety of field types available in Lucene.

Examples of basic field types available in Solr include:

- float
- long
- double
- date
- text

Solr also allows you to define new field types, by combining filters and tokenizers, for example:

```
<fieldtype name="phonetic" stored="false" indexed="true"
class="solr.TextField" >
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.DoubleMetaphoneFilterFactory" inject="false"/>
  </analyzer>
</fieldtype>
```

Defining a field

Here's what a field declaration looks like:

```
<field name="id" type="text" indexed="true" stored="true"
multiValued="true"/>
```

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

The **indexed** and **stored** attributes are important and warrant a little explanation.

Analysis

When data is added to Solr, it goes through a series of transformations before being added to the index. This is called the **analysis** phase. Examples of transformations include lower-casing, removing word stems etc. The end result of the analysis are a series of **tokens** which are then added to the index. **Tokens, not the original text, are what are searched when you perform a search query.**

indexed fields are fields which undergo an analysis phase, and are added to the index.

If a field is not indexed, it cannot be searched on. What use is it then?

Term Storage

Well, when we displaying search results to users, they generally expect to see the original document, not the machine-processed tokens (which may bear very little resemblance to the source text).

That's the purpose of the **stored** attribute: to tell Solr to store the original text in the index somewhere.

Sometimes, there are fields which aren't searched, but need to displayed in the search results. You accomplish that by setting the field attributes to **stored=true** and **indexed=false**.

So, why wouldn't you store all the fields all the time?

Because storing fields increases the size of the index, and **the larger the index, the slower the search**. In terms of physical computing, we'd say that a larger index requires more disk seeks to get to the same amount of data.

What about Lucene, by the way?

Solr is powered by Lucene, a powerful open-source full-text search library, under the hood.

The relationship between Solr and Lucene, is like that of the relationship between a car and its engine.

For the purpose of this introduction, we haven't differentiated between the two, just as to most people, the distinction between a car and its engine is not terribly important when learning how to drive a car.

However, to a mechanic, the distinction is a very important one. Similarly, when we dive deeper under the bonnet of Solr, we'll explore the distinctions between

Lucene and Solr in detail.

Getting Started

- Home
- Solr in 5 Minutes
- Basic Concepts
- Solr Query Syntax
- Lucene Query Builder

Configuring

- Configuring Solr
- schema.xml
- solrconfig.xml

Integrating Solr

- SolrJ

Intermediate

- Search Relevancy
- Boost documents by age

Advanced

- Creating FunctionQueries

Overviews

- Solr vs Elastic Search
- Solr for Managers
- Solr for SysAdmins

Misc

- Need help with Solr?
- About Me

Siblings

- LuceneTutorial.com

- ElasticSearchTutorial.com
- Solr-vs-ElasticSearch.com

© Copyright 2015 Kelvin Tan - Lucene and Solr consultant
Sponsored by Flexile Vertical Search Crawler