Table of Contents

About the author ........................................................................................................ 3
Summary .................................................................................................................... 4
About Solr ............................................................................................................... 5
Installing and Running Solr ................................................................................... 6
Using curl ............................................................................................................... 7
The Solr Configuration Files ................................................................................ 8
The schema.xml File ............................................................................................ 9
Important Data Types .......................................................................................... 10
Analyzers .............................................................................................................. 11
New Fields ........................................................................................................... 12
Existing "Solr Cell" Fields .................................................................................... 13
The solrconfig.xml ............................................................................................. 14
PHP Options ......................................................................................................... 15
Indexing ............................................................................................................... 16
Connecting to the Server ..................................................................................... 17
Iterating over Files ............................................................................................. 18
Omitting Elements From the Index .................................................................... 19
Creating the Index ............................................................................................... 20
Viewing Data ....................................................................................................... 21
Searching ............................................................................................................. 22
Steps in Searching ............................................................................................... 23
Search Code ......................................................................................................... 24
Search Result ....................................................................................................... 25
Refining Searches ............................................................................................... 26
Boosting Documents .......................................................................................... 27
Undervalued Document Example ..................................................................... 28
Corrected by Boosting ......................................................................................... 29
Searching Multiple Fields .................................................................................. 30
Revised Multi-field Search .................................................................................. 31
What’s Different .................................................................................................. 32
About the author

Peter Lavin is a technical writer who has been published in a number of print and online magazines. He is the author of Object Oriented PHP, published by No Starch Press and a contributor to PHP Hacks by O'Reilly Media.
Summary

• This talk describes how to search the HTML files of a specific domain.

• The use case is fairly simple. A single-core installation of Solr is used to search about four thousand documents.

• create configuration files

• create an index (excluding some elements such as navigation headers and footers)

• use this index

• adjust the configuration to deal with some specific issues
About Solr

Apache project that implements the Lucene search library.

Requirements:

• Java runtime environment

• A servlet container such as Jetty or Tomcat

• A recent version of Apache Solr (3.1 or higher)

• A current PHP distribution

• Manage your own RESTful interface or use one of a number of existing Solr clients
Installing and Running Solr

Check that the server is running by entering http://localhost:8983/solr/admin/ into the address bar of your browser.

Figure 1. Solr admin interface
Using curl

Searching from the command line:

curl http://localhost:8983/solr/select/?q=*:*

Deleting the index:

curl http://localhost:8983/solr/update -H "Content-Type: text/xml" \ 
--data-binary '<delete><query>*:*</query></delete>'
curl http://localhost:8983/solr/update -H \ 
"Content-Type: text/xml" --data-binary '<commit/>'

The Solr Configuration Files

The configuration files are:

- **schema.xml** – The file that determines how your data is organized. It is found in the `<SolrHome>/conf` directory.

- **solrconfig.xml** – This file is the primary Solr configuration file, found in the `<SolrHome>/conf` directory. Among other things, this file contains the request handlers, that is the various query types. These are defined by `<requestHandler>` tags.

- **solr.xml** – The principle purpose of this file is for configuring Solr with multi-cores.
The `schema.xml` File

This is the file that determines how your is data organized.

The major elements of this file are:

- `<field>`
- `<fieldType>`
- `<analyzer>`
- `<uniqueKey>`
- `<defaultSearchField>text</defaultSearchField>`
Important Data Types

The data types that are important for our application:

- `string` – plain text field
- `text` – processed (by an analyzer)
- `textgen` – A general text field, the principal difference from the `text` type is that it is unstemmed.
Analyzers

What is an analyzer?

- A processor of text.
- Analyzers are defined within the fieldTypes in `schema.xml`.
- Made up of tokenizers and filters. Tokenizers split up text and filters transform.
- There are index and search analyzers.
New Fields

The fields added to the `schema.xml` for our application are as follows:

**Figure 2. Fields**

```xml
<field name="id" type="string" indexed="false" stored="true" required="true" />
<field name="book" type="string" indexed="false" stored="true"/>
<field name="category" type="text" indexed="true" stored="true"/>
```

The `<field>` attributes are as follows:

- `stored` – if you want to display a field
- `indexed` – if you want to search on a field then this must be `true`. 
Existing "Solr Cell" Fields

The other fields that we are going to use are already defined in the schema.xml as metadata fields. They are as follows:

• title
• links
• content_type
• content (text)

<!-- catchall field, containing all other searchable text fields (implemented via copyField further on in this schema -->

<field name="text" type="text" indexed="true" stored="false" multiValued="true"/>
The **solrconfig.xml**

This file specifies high level configuration options.

The ExtractingRequestHandler is also known as Solr Cell (originally, Tika)

The default configuration (this is from Solr 1.4 and differs slightly in later versions) is as follows:

**Figure 3. Solr Cell**

```xml
<!-- Solr Cell: http://wiki.apache.org/solr/ExtractingRequestHandler -->
<requestHandler name="/update/extract" class="org.apache.solr.handler.extraction.ExtractingRequestHandler">
  
  <list name="defaults">
    <!-- All the main content goes into "text"... if you need to return the extracted text or do highlighting, use a stored field. -->
    <str name="fmap.content">text</str>
    <str name="lowernames">true</str>
    <str name="uprefix">ignored</str>
  
  </list>

  <!-- capture link hrefs but ignore div attributes -->
  <str name="captureAttr">true</str>
  <str name="fmap.a">links</str>
  <str name="fmap.div">ignored</str>
</requestHandler>
```
There are a variety of ways that you can use PHP with Solr. Some of the options are listed below:

- Do it yourself
- PecI Solr – http://pecl.php.net/package/solr
- Solarium – http://www.solarium-project.org/
Indexing

The steps in indexing a page are as follows:

• Create a connection to the Solr server
• Iterate over the different directories
• Remove web page elements that shouldn't appear in the index
• Identify the stored parameters, such as id (URL), book, category.
• Index the document (web page).
Connecting to the Server

Connect to the Solr server:

**Figure 4. Server connection**

```php
7
8    // Create a new Solr service instance - host, port, and webapp
9    // path which defaults to Solr Home plus solr
10   $server = 'localhost';
11   $port = 8983;
12   $solr = new Apache_Solr_Service($server, $port, 'solr/');
13
14   if (! $solr->ping()) {
15     echo 'Solr service not responding.';
16     exit(1);
17   }
```
Iterating over Files

All web directories are found within the online directory. Recursively iterate over all the directories found here and process each file as follows:

Figure 5. Iterating over a directory

```php
$dir = 'online';
foreach (new RecursiveIteratorIterator(new RecursiveDirectoryIterator($dir)) as $file) {
    // only index PHP files.
    if (pathinfo($file, PATHINFO_EXTENSION) == 'php') {
        // create DOM object.
        $dom = new DomDocument();
        clean_up($file, $dom);
        // strip off top level directory.
        $file = trim($file, $dir . '/');
        echo 'Indexing file ' . $file . $n';
        // determine category.
        $category = create_category($file);
        // determine book.
        $book = determine_book($file);
        // index the file.
        index_document($file, $category, $book, $dom, $solr);
    }
}
```
Omitting Elements From the Index

We know that there are elements of a web page that we don't want to index. In the following example, $dom is a DomDocument and the <div> with the class name navheader is a navigation header that we don't want to index.

Figure 6. Omitting elements

```php
if (!$dom->loadHTMLFile($file)) {
    echo "Failed to load $file";
    exit(1);
}
//create XPath object.
$xp = new DOMXPath($dom);
//preprocess the file by .
//removing the offending div .
$div = $xp->query("//div[@class='navheader']");
//there's only one, remove it.
//but check that it is set .
if ($div->item(0)){
    $div->item(0)->parentNode->removeChild($div->item(0)); .
}
//remove the offending footer .
$div = $xp->query("//div[@class='navfooter']");
if ($div->item(0)){
    $div->item(0)->parentNode->removeChild($div->item(0)); .
}
```

This code removes any <div>s of the class navheader prior to indexing.
Creating the Index

Figure 7. Indexing

```php
function index_document($file, $category, $book, $dom, $solr)
{
    $params['literal.id'] = $file;
    if ($category != '') {
        $params['literal.category'] = $category;
    }
    $params['literal.book'] = $book;
    //we already map the content to "text".
    //in the solrconfig.xml file so need not do it here.
    //params['map.content'] = "text";
    //Solr Cell/Tika will detect content-type automatically.
    //but be explicit.
    $params['stream.contentType'] = "text/html";
    /*
     * Check here if there is a doctype declaration and remove it..
     * Solr Client chokes if there is a doctype and it.
     * doesn’t match “loose” so remove the doctype declaration..
     * You can tell that the doctype is there if “html” has a parent..
     */
    $xp = new DOMXPath($dom);
    $html = $xp->query("//html");
    if ($html->item(0)->parentNode){
        //remove doctype, header and footer already gone.
        $cleaned = $dom->saveXML($html->item(0));
    } else {
        //get the xml with the header and footer removed.
        $cleaned = $dom->saveXML();
    }
    try{
        // use extract from string, no point writing back to file.
        // and re-reading it!
        /*
         * public function extractFromString($string, $params = array(), $document = null, ...
         * $mime-type = 'application/octet-stream')
         */
        $solr->extractFromString($cleaned, $params);
        //add other parameters here after creating?
    } catch (Exception $e){
        echo $e->__toString();
    }
}
```
Viewing Data

An easy way to view the fields defined in the schema.xml file is to navigate from the admin screen to Schema Browser, Fields screen. Find below the details of the book field.

Figure 8. Fields admin screen
Searching

The web search interface is very simple:

**Figure 9. Web interface**

Users can specify a query and they can filter their query by the *book* field.

They can use Lucene syntax as specified at [http://lucene.apache.org/core/3_6_0/queryparsersyntax.html](http://lucene.apache.org/core/3_6_0/queryparsersyntax.html)
Steps in Searching

The steps for coding a search are as follows:

1. Create a client
2. Define a query
3. Define search parameters
4. Get result set
5. Iterate over the result set
Search Code

You create a search client in the same way that you create an index client. Basic search is very straightforward:

Figure 10. Searching

```php
$params['fl'] = '*', 'score';
$filter = get_section($section);
$params['fq'] = $filter;
$results = $solr->search($query, 0, $limit, $params);
```
Search Result

What does the result look like? Let's look at a single item.

```json
{
    "responseHeader": {
        "status": 0,
        "QTime": 8,
        "params": {
            "fq": "book:*",
            "defType": "dismax",
            "rows": "100",
            "q": "\"read on startup by\"",
            "start": "0",
            "wt": "json",
            "json.nl": "map",
            "qf": "title text category",
            "fl": "title,book,id,category,score"
        }
    },
    "response": {
        "start": 0,
        "maxScore": 0.22452901,
        "numFound": 1,
        "docs": [
            {
                "category": "Ingestor",
                "score": 0.22452901,
                "book": "Message Scope Reference",
                "id": "web-message-scope/msc.ingestor.php",
                "title": "Chapter 8. Configuring the Ingestor"
            }
        ]
    }
}
```
Refining Searches

We're going to tweak search results to compensate for the following kinds of irregularities:

• Compensate for overvalued (or undervalued) pages by boosting

• Search on more than one field

• Adjust for specialized vocabularies
Boosting Documents

• Ranked by score

• For the algorithm see Lucene Similarity Class.

• Document size distortion

• TOC that references a topic may score higher than the actual document that deals with the topic

• Boost field to correct this distortion
Undervalued Document Example

Figure 11. No boost

Search Results For: counter

There were 80 hits.

- `msys.counter.unlink` (Lua, Momentum Reference Manual (3.x)) 202.28%
- `msys.counter.reset` (Lua, Momentum Reference Manual (3.x)) 189.21%
- `msys.counter.inc` (Lua, Momentum Reference Manual (3.x)) 152.91%
- `msys.counter.read` (Lua, Momentum Reference Manual (3.x)) 151.71%
- `msys.counter.add` (Lua, Momentum Reference Manual (3.x)) 148.93%
- `counter` (Eckelty Console Commands, Momentum Reference Manual (3.x)) 145.61%
- `ec_inc_counter` (Sieve, Momentum Reference Manual (3.x)) 139.9%
- `msys.counter.open` (Lua, Momentum Reference Manual (3.x)) 136.16%
- `ec_inc_counter` (Sieve, Momentum Reference Manual 2.2) 115.59%
- `thread.mutex` (Lua, Momentum Reference Manual (3.x)) 108.12%
- `ec_tarpit` (Sieve, Momentum Reference Manual (3.x)) 71.52%
- `ec_tarpit` (Sieve, Momentum Reference Manual 2.2) 71.52%
- `msys.shareGet` (Lua, Momentum Reference Manual (3.x)) 64.62%
- `msys.shareSet` (Lua, Momentum Reference Manual (3.x)) 64.62%
- `B.2. Per-group (binding) Metrics, 1.3.6.1.4.1.19552.1.3` (Momentum Reference Manual (3.x)) 62.56%

The result that appears in sixth place is the result that should appear at the top of the list.
Corrected by Boosting

Searching the same documents where the title field has been boosted:

Figure 12. With title boost

Search Results For: counter

There were 80 hits.

- counter (Ecelerity Console Commands, Momentum Reference Manual (3.x)) 832.48%
- msys.counter.unlink (Lua, Momentum Reference Manual (3.x)) 11.23%
- msys.counter.reset (Lua, Momentum Reference Manual (3.x)) 10.51%
- msys.counter.inc (Lua, Momentum Reference Manual (3.x)) 6.49%
- msys.counter.read (Lua, Momentum Reference Manual (3.x)) 6.43%
- msys.counter.add (Lua, Momentum Reference Manual (3.x)) 6.27%
- ec_inc_counter (Sieve, Momentum Reference Manual (3.x)) 7.77%
- msys.counter.open (Lua, Momentum Reference Manual (3.x)) 7.56%
- ec_inc_counter (Sieve, Momentum Reference Manual 2.2) 6.42%
- thread.mutex (Lua, Momentum Reference Manual (3.x)) 6.01%
- ec_tarpit (Sieve, Momentum Reference Manual (3.x)) 3.97%
- ec_tarpit (Sieve, Momentum Reference Manual 2.2) 3.97%
- msys.shareGet (Lua, Momentum Reference Manual (3.x)) 3.59%
- msys.shareSet (Lua, Momentum Reference Manual (3.x)) 3.59%
- B2_Per-group (binding) Metrics. 1.3.6.1.4.1.19552.1.3 (Momentum Reference Manual (3.x)) 3.47%

Here's the code that boosts fields at index time:

```
$params["boost.title"] = "1.2";
```

You can boost at index time or at query time. In our case it makes good sense to boost at index time.
Searching Multiple Fields

The query text box supports the Lucene query syntax so you can add an AND to your search. (See Figure 9, “Web interface”.)

Figure 13. Before

Search Results For: counter AND lua

- msys.counter.unlink (Lua, Momentum Reference Manual (3.x))
- msys.counter.read (Lua, Momentum Reference Manual (3.x))
- counter (Ecoleerly Console Commands, Momentum Reference Manual (3.x))
- msys.shareSet (Lua, Momentum Reference Manual (3.x))
- lua.ref.msys.dumper.Dump (Lua, Momentum Reference Manual (3.x))
- Momentum Reference Manual (3.x) (Momentum Reference Manual (3.x))
- 16.1. Lua Function Summary (Lua, Momentum Reference Manual (3.x))
- ADAPTIVE_API (C/C++ SDK, Momentum API Reference (3.x))
- ADAPTIVE_API (C/C++ SDK, Momentum API Reference 2.2)
- Momentum Reference Manual (3.x) (Momentum Reference Manual (3.x))
- 14.35. inbound_audit – inbound traffic analytics (Module, Momentum Reference Manual (3.x))
- 7.7. Data Replication (Clustering, Momentum Reference Manual (3.x))
- Chapter 2. C API Reference (C/C++ SDK, Momentum API Reference (3.x))
- Chapter 17. C API Reference (C/C++ SDK, Momentum Reference Manual 2.2)
- Part III. Extending Momentum (Momentum Reference Manual 2.2)
Revised Multi-field Search

Here's what a revised search that searches on the category field as well as the body of the web page.

Figure 14. After

The original search returned 15 results and the second returns 24 hits. Both searches are made against the same set of documents. The search didn't miss 9 results. It only did what it was told.
What's Different

The original search searched only the `text` field.

**Figure 15. Revised search**

```php
// get the score as well &fl=*,score, must be part of the params array.
// don't need to actually return the title field.
$params["fl"] = "title,book,id$category,category,score";
// this query parser is available as of Solr 3.1.
$params["defType"] = "edismax";
// title has been boosted at index time.
// could be boosted here.
$params["qf"] = "title text category";
$filter = get_section($section);
$params["fq"] = $filter;
$results = $solr->search($query, 0, $limit, $params);
```

To search on multiple fields you need to use the `dismax` (or `edismax`) query parser (`defType`). This is now the default but in earlier versions of Solr it was not. More results are being returned because an additional field is being queried (`qf`).
Specialized Vocabulary

Figure 16. Camel case

Searching for `mymethod` vs `myMethod` returns different results.
Fix for Camel Case Search

The standard Solr configurations are geared towards generalized use cases.

Figure 17. Analyzer and WordDelimiterFilterFactory

```xml
<fieldType name="text" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StopFilterFactory">
      <ignoreCase>true</ignoreCase>
      <words>stopwords.txt</words>
      <enablePositionInCrements>true</enablePositionInCrements>
    </filter>
    <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1"
      cAtenateWords="1" cAtenateNumbers="1" cAtenateAll="0" splitOnCaseChange="1"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory" language="English" protected="protwords.txt"/>
  </analyzer>
</fieldType>
```

The purpose of this analyzer is to both split and join compound words and one of the criteria for splitting is case change.

We need to turn off splitting on case change.

`splitOnCaseChange="0"`
Resources

The Apache Solr Website – http://lucene.apache.org/solr/


Dismax Queries – http://www.lucidimagination.com/

The ExtractingRequestHandler – http://wiki.apache.org/solr/ExtractingRequestHandler


Content Extraction with Tika – http://www.lucidimagination.com/Community/Hear-from-the-Experts/Articles/Content-Extraction-Tika

Solr Pecl – http://pecl.php.net/package/solr
Questions