Opzet gebruik SOLR

# SOLR conventies

SOLR gebruikt een schema file waarin de velden zijn gedefnieerd.

Een input document ziet er bijvoorbeeld als volgt uit:

<doc>

<field name="id">SOLR1000</field>

<field name="name">Solr, the Enterprise Search Server</field>

<field name="manu">Apache Software Foundation</field>

<field name="cat">software</field>

<field name="cat">search</field>

<field name="features">Advanced Full-Text Search Capabilities using Lucene</field>

<field name="features">Optimized for High Volume Web Traffic</field>

<field name="features">Standards Based Open Interfaces - XML and HTTP</field>

<field name="features">Comprehensive HTML Administration Interfaces</field>

<field name="features">Scalability - Efficient Replication to other Solr Search Servers</field>

<field name="features">Flexible and Adaptable with XML configuration and Schema</field>

<field name="features">Good unicode support: h&#xE9;llo (hello with an accent over the e)</field>

<field name="price">0.0</field>

<field name="popularity">10</field>

<field name="inStock">true</field>

<field name="incubationdate\_dt">2006-01-17T00:00:00.000Z</field>

</doc>

Wat er met een veld moet worden gedaan staat in schema.xml, onder dezelfde ‘name=’ value.

Let op: neem een beslissing over UTF-8 in SOLR! Lijkt me gewenst.

## Field name convention

Om het schema simpel te houden worden field name conventies gebruikt. Hieronder staan de conventies die ik heb gehanteerd:

|  |  |
| --- | --- |
| \*\_i | Integer veld, niet herhaald |
| \*\_is | Herhaald integer veld |
| \*\_s | String veld, in z’n geheel geïndexeerd |
| \*\_ss | Herhaald string veld, elke occurence in z’n geheel geïndexeerd |
| \*\_t | Tekst veld, wordt woord voor woord geïndexeerd |
| \*\_txt | Herhaald tekst veld, elke occurence woord voor woord geïndexeerd |
| \*\_dt | Datum veld, |
| \*\_dts | Herhaald datumveld |

# Werkwijze

Er zijn meerdere input bestanden: BIB, LBD en LHR, THES

Voorstel: maak voor elk inputbestand 1 tabel in de database D. De tabelstructuur is dan:

database D:

table: D.bib

column: D.bib.ocnPlus //main key

column: D.bib.ocn //indexed

column: D.bib.data //een titelbeschrijving in SOLR XML

table: D.LBD

column: D.bib.ocnPlus //main key

column: D.LBD.ocn //indexed

column: D.LBD.data //LBD fields in SOLR XML

**OF, wanneer LBD een eigen num heeft:**

table: D.LBD

column: D.LBD.lbdnumPlus //main key

column: D.LBD.ocn //indexed

column: D.LBD.data //LBD fields in SOLR XML

table: D.LHR

column: D.LHR.lhrnumPlus //main key

column: D.LHR.ocn //indexed

column: D.LHR.data //LHR fields in SOLR XML

table: D.THES

column: D.THES.ppnPlus //main key

column: D.THES.ppn //indexed

column: D.THES.data //THES fields in SOLR XML

table: D.links

column: D.links.fromOcnPlus //indexed

column: D.links.toOcnPLUS //indexed

column: D.links.linktype //indexed

main key (if required): fromOcnPlus + linktype + toOcnPlus

Belangrijke voorbereiding: maak het schema.

Voorbeeld:

## Werkwijze stap voor stap:

Belangrijk: de data zijn meestal ok, maar kunnen inconsistent, incompleet en fout zijn….

1. Lees elk inputbestand in in z’n tabel. Doet dat eventueel batchgewijs (splits het inputbestand in brokken).
2. Converteer per record MARC XML naar SOLR XML. Daarvoor is een ‘traverse’ nodig van de MARC XML en een XPath functie om bij een veld een ander veld te zoeken. Gebruik bijvoorbeeld simpleXML of DOM.
3. Voor BIB
   1. Behandel het OCN. Kijk of het OCN eerder is ingelezen.

Zo nee: voer het OCN in in D.bib.ocnPlus en in D.bib.ocn

Zo ja: voer het OCN + volgnummer in in D.bib.ocnPlus en het OCN in D.bib.ocn. Gebruik SELECT ocn FROM D.bib WHERE ocn=OCN om het aantal eerder ingevoerde records met hetzelfde OCN te weten.

* 1. Start het SOLR XML record, bijvoorbeeld:

<field name="id\_s">ocnPlus</field>

<field name="ocn\_s">ocn</field>

Merk op dat het start element <doc> ontbreekt, dat komt later.

* 1. Loop door de MARC XML velden. Er zijn meerdere mogelijkheden:
     1. Neem over in SOLR XML, bijvoorbeeld:

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

* + 1. Verbeter en neem over in SOLR XML, bijvoorbeeld:

<field name="language\_txt">Taal voluit</field>

* + 1. Neem over en maak een apart veld met een verbeterde versie:

<field name="year\_txt">ca. 1930</field>

<field name="year\_dt">1930</field>

* + 1. Combineer met andere velden:
    2. Neem links op in de tabel links
  1. Sla het SOLR XML record in z’n geheel op in bib.data

1. Voor LBD: doe analoog hetzelfde, maar:
   1. Zorg dat eventuele PPN’s apart bewaard worden:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

1. Voor LHR: doe analoog hetzelfde
2. Voor THES: doe analoog hetzelfde, hopelijk is de export ook in een XML format
3. Loop door D.bib, stel per record het definitieve SOLR XML record samen t.b.v. invoer:
   1. Start het definitieve SOLR XML record:

<doc>

* 1. Neem het XML record uit D.bib.data over:

<doc>

<field name="id\_s">ocnPlus</field>

<field name="ocn\_s">ocn</field>

<field name="title\_txt">De titel</field>

<field name="title\_txt">Subtitel</field>

<field name="title\_txt">Nog een subtitel</field>

<field name="year\_txt">ca. 1930</field>

<field name="year\_dt">1930</field>

…

* 1. Zoek het bijbehorende LBD record (eventueel meer) m.b.v. het ocn en plak de data uit D.lbd.data aan het definitieve SOLR XML record
  2. Doe hetzelfde met de LHR’s
  3. Verrijk de thesaurus termen, ga voor elk veld van de vorm:

<field name="thes\_ppn\_s">PPN van een thesaurus term</field>

Het ppn opzoeken in D.thes en plak de inhoud van D.thes.data in het definitieve SOLR XML record.

* 1. Maak de inhoud van het SOLR veld text:

Verzamel alle relevante veldinhouden, voeg ze samen (gescheiden door een spatie) en voeg toe:

<field name="text">verzamelde tekst</field>

* 1. Maak de inhoud van het SOLR veld listEntry:

Verzamel alle relevante veldinhouden, voeg ze samen in de vorm

title:”de titel”;author:”de author”;…

en voeg toe:

<field name="listEntry">

title:”de titel”;author:”de author”;…

</field>

* 1. Maak de inhoud van het SOLR veld detailEntry:

Analoog aan listEntry. Maar met meer velden.

* 1. Behandel de eventuele links:

SELECT \* FROM D.links WHERE fromOcnPlus=OCN

Voeg voor elk resultaat van deze query toe:

<field name="D.links.linktype".’\_ss’”>D.links.toOcnPLUS</field>

Voorbeeld:

<field name="chapInBook\_ss">123456</field>

SELECT \* FROM D.links WHERE toOcnPLUS=OCN

Voeg voor elk resultaat van deze query toe:

<field name="inverse(D.links.linktype).’\_ss’">D.links.fromOcnPlus</field>

Voorbeeld:

<field name="bookHasChap\_ss">654321</field>

* 1. GEAVANCEERD: Zorg ervoor dat je alle VIAF nummers bewaard (analoog aan 4.1). Ga voor elke VIAF link de VIAF database bevragen (VIAF\_Service.php) en voeg relevante content toe aan het definitieve SOLR XML record.
  2. Sluit het definitieve SOLR XML record af:

</doc>

* 1. Stuur het record naar SOLR

# Het SOLR schema

Het SOLR schema ziet er globaal als volgt uit:

**<schema>**

**<types>**

**<fields>**

**<uniqueKey>**

**<copyField>**

**</schema>**

De file ../server/solr/configsets/\_default/conf/managed-schema is een uitgewerkt én goed gedocumenteerd schema. Het idee is dat deze file gekopieerd wordt naar schema.xml en wordt aangepast.

Zie Github voor een door mij alvast enigszins aangepaste schema.xml

# Zoeken

Gebruik PHP’s SolrClient (<https://www.php.net/manual/en/class.solrclient.php>)

## Het zoekscherm

Het zoekscherm bestaat uit de volgende onderdelen:

1. Query input met de query zoals ingevoerd door de gebruiker
2. Header, met aantal gevonden, volgende/vorige pagina, etc.
3. Facets
4. Details

Bij elke zoekactie genereert SOLR een zoekresultatenlijst met alle gegevens. Dus ook **listEntry’s** en **detailEntry’s** worden aangeleverd.

De listEntry’s zijn handig voor het snel opmaken van de lijst.

Wanneer op een element in de lijst wordt geklikt kan het detailscherm worden opgemaakt door de bijbehorende detailEntry te vertonen.

## Dynamische verrijking lijst en details

Als het SOLR zoekresultaat binnen is, dan moeten er via de PHP libraries *Availability\_Service* en *WorldCat\_KB\_Service* gegevens uit WMS worden opgehaald. Dit moet voor elk item in de lijst gebeuren. Om dit redelijk snel te kunnen doen lijken mij Ajax calls vanuit de pagina handig. Het scherm kan dan worden opgebouwd en elke keer als een antwoord van WMS binnenkomt kan dat worden toegevoegd.

Werkwijze:

require\_once './Availability\_Service.php';

$avail = new Availability\_Service('keys\_availability.php');

$holding = $avail->get\_circulation\_info($ocn)

Analyseer de $holding array om gegevens te vinden over de fysieke beschikbaarheid.

require\_once './WorldCat\_KB\_Service.php';

$KB = new WorldCat\_KB\_Service('keys\_worldcat\_kb.php');

$found = $KB->search\_kb\_record($ocn);

//delete the previous line if only the link is needed

$href = $KB->getlink($ocn);

// complete response in $KB->kb\_record

Als de $href een lege string is dan is er geen digitale versie geregistreerd.

## Zoekschem class

Het lijkt mij handig om een class te maken voor het zoekscherm:

<?php

class SearchPage {

public $userQuery = ''; //of "\*.\*"

private $parsedQuery = '';

public $start = 0;

public $numRows = 10;

private $solrOptions = array (

'hostname' => SOLR\_SERVER\_HOSTNAME,

'login' => SOLR\_SERVER\_USERNAME,

'password' => SOLR\_SERVER\_PASSWORD,

'port' => SOLR\_SERVER\_PORT,

);

private $facetFields = array(

);

private $solrClient = nulll //of type SolrClient

private $solrQuery = null; //of type SolrQuery

public $searchResults = array();

public function \_\_construct() {

$client = new SolrClient($this->solrOptions);

$query = new SolrQuery();

}

public function prepareSearch($userInput) {

$this->userQuery = $userInput;

$this->parsedQuery = $this->parse($userInput);

$this->query->setQuery($this->parsedQuery);

$this->query->setStart($this->start);

$this->query->setRows($this->numRows);

//etc

$this->search();

}

private function search() {

$query\_response = $this->client->query($this->query);

$query\_response->setParseMode(SolrQueryResponse::PARSE\_SOLR\_DOC); //??

$this->searchResults = $query\_response->getResponse();

$this->

}

public function nextPage() {

$this->start = $this->start + $this->numRows;

if ($this->start < $this->searchResults['response']['numFound']) $this->search();

}

public function prevPage() {

$this->start = $this->start - $this->numRows;

if ($this->start > 0) $this->search();

}

//etc

}

TODO complete this chapter