Plone Training Solr Documentation

Release 1.0

Timo Stollenwerk

May 17, 2017

Contents

Contents:

CHAPTER 1

Set up Plone and Solr

Buildout

Bootstrap project:

```
$ mkdir plone-training-solr
$ cd plone-training-solr
$ wget https://bootstrap.pypa.io/bootstrap-buildout.py
$ wget https://raw.githubusercontent.com/collective/collective.solr/json-api/solr.cfg
```

Create Buildout (buildout.cfg):

```
[buildout]
extends =
    http://dist.plone.org/release/4.3.6/versions.cfg
    solr.cfg
parts += instance

[instance]
recipe = plone.recipe.zope2instance
http-address = 8080
user = admin:admin
eggs =
    Plone
    collective.solr

[versions]
zope.interface = 4.0.5
zc.buildout = 2.3.1
setuptools = 8.0.4
```

Run buildout:

```
$ python2.7 bootstrap-buildout.py
$ bin/buildout
```

Start Plone:

```
$ bin/instance fg
```

Start Solr:

```
$ bin/solr-instance fg
```

Solr Buildout

Buildout parts:

```
[buildout]
parts +=
    solr-download
    solr-instance
```

Base Solr Settings:

```
[settings]
solr-host = 127.0.0.1
solr-port = 8983
solr-min-ram = 128M
solr-max-ram = 256M
```

Solr Download:

```
[solr-download]
recipe = hexagonit.recipe.download
strip-top-level-dir = true
url = https://archive.apache.org/dist/lucene/solr/4.10.4/solr-4.10.4.tgz
md5sum = 8ae107a760b3fc1ec7358a303886ca06
```

Solr Instance:

```
[solr-instance]
recipe = collective.recipe.solrinstance
solr-location = ${solr-download:location}
host = ${settings:solr-host}
port = ${settings:solr-port}
basepath = /solr
# autoCommitMaxTime = 900000
max-num-results = 500
section-name = SOLR
unique-key = UID
logdir = ${buildout:directory}/var/solr
default-search-field = default
default-operator = and
unique-key = UID
java_opts =
  -Dcom.sun.management.jmxremote
  -Djava.rmi.server.hostname=127.0.0.1
  -Dcom.sun.management.jmxremote.port=8984
  -Dcom.sun.management.jmxremote.ssl=false
```

```
  -Dcom.sun.management.jmxremote.authenticate=false
  -server
  -Xms${settings:solr-min-ram}
  -Xmx${settings:solr-max-ram}
```

Index:

```
index =
    name:allowedRolesAndUsers   type:string stored:false multivalued:true
    name:created                type:date stored:true
    name:Creator                type:string stored:true
    name:Date                   type:date stored:true
    name:default                type:text indexed:true stored:false multivalued:true
    name:Description            type:text copyfield:default stored:true
    name:description            type:text copyfield:default stored:true
    name:effective              type:date stored:true
    name:exclude_from_nav       type:boolean indexed:false stored:true
    name:expires                type:date stored:true
    name:getIcon                type:string indexed:false stored:true
    name:getId                  type:string indexed:false stored:true
    name:getRemoteUrl           type:string indexed:false stored:true
    name:is_folderish           type:boolean stored:true
    name:Language               type:string stored:true
    name:modified               type:date stored:true
    name:object_provides        type:string stored:false multivalued:true
    name:path_depth             type:integer indexed:true stored:false
    name:path_parents           type:string indexed:true stored:false multivalued:true
    name:path_string            type:string indexed:false stored:true
    name:portal_type            type:string stored:true
    name:review_state           type:string stored:true
    name:SearchableText         type:text copyfield:default stored:false
    name:searchwords            type:string stored:false multivalued:true
    name:showinsearch           type:boolean stored:false
    name:Subject                type:string copyfield:default stored:true␣
↪multivalued:true
    name:Title                  type:text copyfield:default stored:true
    name:Type                   type:string stored:true
    name:UID                    type:string stored:true required:true
```

- name: Name of the field
- type: Type of the field (e.g. “string”, “text”)
- indexed: searchable
- stored: returned as metadata
- copyfield: copy content to another field, e.g. copy title, description, subject and SearchableText to default.

https://wiki.apache.org/solr/SchemaXml#Common_field_options

Plone and Solr

Activate Solr in Plone:

```
- Create Plone instance with collective.solr installed
- Go to: "Configuration" -> "Solr Settings"
- Check: "Active", click "Save"
```

```
- Go to: http://localhost:8080/Plone/@@solr-maintenance/reindex
- Search for "Plone"
```

CHAPTER 2

Solr GUI and Query

Access Solr Gui

- Go to: http://localhost:8983/solr/#/
- Select Core “collection1”
- Go to: “Schema Browser”
- Select “Title”
- Click: “Load Term Info”
- Click on term “nachrichten”

Solr Query

Solr Query Parameters:

Query “q”:

```
Title:"nachrichten"
*:"nachrichten"
```

Filter Query “fq”:

```
is_folderish:true
```

Sorting “sort”:

```
"Date asc"
"Date desc"
```

Filter List “fl”:

```
Title,Type
```

This parameter can be used to specify a set of fields to return, limiting the amount of information in the response.

Response Writer “wt”:

```
"json"
```

A Response Writer generates the formatted response of a search.

Solr Query via URL

Copy query from Solr GUI, e.g.:

```
http://localhost:8983/solr/collection1/select?q=Title%3A%22termine%22&wt=json&
↪indent=true
```

Advanced Solr Query Syntax

Simple Query:

```
"fieldname:value"
```

Operators:

```
"Title:Foo AND Description:Bar"
```

“AND”, “OR”, “+”, “-“

Range Queries:

```
"[* TO NOW]"
```

Boost Terms:

“people^4”

Fuzzy Search:

```
"house0.6"
```

Proximity Search:

```
"'apache solr'2"
```

Questions

- What do we get by using Solr instead of the Plone search?
- Why don’t we query Solr directly in Plone?
- What does collective.solr do?

CHAPTER 3

Collective Solr

Collective Solr Control Panel

Basic Configuration:

- Active
- Host
- Port
- Base
- ...

Query Configuration:

```
+(Title:{value}^5 OR
  Description:{value}^2 OR
  SearchableText:{value} OR
  SearchableText:({base_value}
)
OR searchwords:({base_value})^1000) +showinsearch:True
```

Base Functionality

- Patches the ZCatalog
- Some queries are faster in Solr some are not
- Indexes and Metadata duplicated
- Full text search with SearchableText

Transactions

Solr is not transactional aware or supports any kind of rollback or undo. We therefor only sent data to Solr at the end of any successful request. This is done via collective.indexing, a transaction manager and an end request transaction hook. This means you won't see any changes done to content inside a request when doing Solr searches later on in the same request.

Querying Solr with collective.solr

ZCatalog Query:

```
catalog(SearchableText='Foo', portal_type='Document')
```

Result is a Solr Object.

Direct Solr Queries:

```
solr_search = solrSearchResults(
    SearchableText=SearchableText,
    spellcheck='true',
    use_solr='true',
)
```

You can pass Solr query params directly to Solr and force a Solr response with "use_solr='true"'.

Though, you have to make sure the response also contains

Mangler

collective.solr has a mangleQuery function that translates / mangles ZCatalog query parameters to replace zope specifics with equivalent constructs for Solr.

https://github.com/collective/collective.solr/blob/master/src/collective/solr/mangler.py#L96

CHAPTER 4

Solr Buildout Configuration

Solr Multi Core

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance:mc
cores =
  collection1
  collection2
  collection3
  testing
default-core-name = collection1
```

Stopwords

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance
filter =
    text solr.StopFilterFactory ignoreCase="true" words="${buildout:directory}/etc/
↪stopwords.txt"
```

stopwords.txt:

```
der
die
das
und
oder
```

http://svn.apache.org/repos/asf/lucene/dev/trunk/lucene/analysis/common/src/resources/org/apache/lucene/analysis/snowball/german_stop.txt

Stemming

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance
...
filter =
#    text solr.GermanMinimalStemFilterFactory  # Less aggressive
#    text solr.GermanLightStemFilterFactory  # Moderately aggressiv
#    text solr.SnowballPorterFilterFactory language="German2"  # More aggressive
    text solr.StemmerOverrideFilterFactory dictionary="${buildout:directory}/etc/
↪stemdict.txt" ignoreCase="false"
```

stemdict.txt:

```
# english stemming
monkeys monkey
otters  otter

# some crazy ones that a stemmer would never do
dogs    cat

# german stemming
gelaufen    lauf
lief        lauf
risiken     risiko
```

Synonyms

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance
...
filter-index =
# The recommended approach for dealing with synonyms is to expand the synonym
# when indexing. See: http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters
↪#solr.SynonymFilterFactory
    text solr.SynonymFilterFactory synonyms="${buildout:directory}/etc/synonyms.txt"␣
↪ignoreCase="true" expand="true"
```

synonyms.txt:

```
#Explicit mappings match any token sequence on the LHS of "=>"
#and replace with all alternatives on the RHS.  These types of mappings #ignore the␣
↪expand parameter in the schema.
#Examples:
i-pod, i pod => ipod,
sea biscuit, sea biscit => seabiscuit
```

```
#Equivalent synonyms may be separated with commas and give #no explicit mapping.  In␣
↪this case the mapping behavior will #be taken from the expand parameter in the␣
↪schema.  This allows #the same synonym file to be used in different synonym␣
↪handling strategies.
#Examples:
ipod, i-pod, i pod
foozball , foosball
universe , cosmos

# If expand==true, "ipod, i-pod, i pod" is equivalent to the explicit mapping:
ipod, i-pod, i pod => ipod, i-pod, i pod # If expand==false, "ipod, i-pod, i pod" is␣
↪equivalent to the explicit mapping:
ipod, i-pod, i pod => ipod

#multiple synonym mapping entries are merged.
foo => foo bar
foo => baz
#is equivalent to
foo => foo bar, baz
```

Autocomplete

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance
...
additional-schema-config =
  <copyField source="Title" dest="title_autocomplete" />
  <copyField source="Description" dest="description_autocomplete" />
  <copyField source="Title" dest="title_suggest" />

extra-field-types =
  <fieldType class="solr.TextField" name="text_auto">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ShingleFilterFactory" maxShingleSize="4" outputUnigrams=
↪"true"/>
      <filter class="solr.EdgeNGramFilterFactory" maxGramSize="20" minGramSize="1"/>
     </analyzer>
  </fieldType>
  <fieldType class="solr.TextField" name="text_desc">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ShingleFilterFactory" maxShingleSize="4" outputUnigrams=
↪"true"/>
      <filter class="solr.EdgeNGramFilterFactory" maxGramSize="20" minGramSize="1"/>
     </analyzer>
   </fieldType>


# Solr Config => parts/solr/solr/collection1/conf/solrconfig.xml
additional-solrconfig =
```

```
  <!-- ================================================================= -->
  <!-- AUTOCOMPLETE                                                      -->
  <!-- ================================================================= -->

  <requestHandler name="/autocomplete" class="solr.SearchHandler">
    <lst name="defaults">

      <!-- defType: a reference to the query parser that is used.
           The 'edismax' query parser adds features to enhance search relevancy.
           https://wiki.apache.org/solr/ExtendedDisMax -->
      <str name="defType">edismax</str>

      <!-- rows: maximum number of documents included in the response
           https://wiki.apache.org/solr/CommonQueryParameters#rows -->
      <str name="rows">10</str>

      <!-- fl: field list to be returned in the response. -->
      <str name="fl">description_autocomplete,title_autocomplete,score</str>

      <!-- qf: query fields list with 'boosts' that are associated with each
           field.
           https://wiki.apache.org/solr/ExtendedDisMax#qf_.28Query_Fields.29
           -->
      <str name="qf">title_autocomplete^30 description_autocomplete^50.0</str>

      <!-- pf: phrase fields list to 'boost' the score (after 'fq' and 'qf')
           of documents where terms in 'q' appear in close proximity.
           https://wiki.apache.org/solr/ExtendedDisMax#pf_.28Phrase_Fields.29
           -->
      <str name="pf">title_autocomplete^30 description_autocomplete^50.0</str>

      <!-- result grouping:
           https://wiki.apache.org/solr/FieldCollapsing#Request_Parameters -->
      <str name="group">true</str>
      <str name="group.field">title_autocomplete</str>
      <str name="group.field">description_autocomplete</str>
      <str name="sort">score desc</str>
      <str name="group.sort">score desc</str>

    </lst>
  </requestHandler>
```

Suggest

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance
...

additional-solrconfig =

    <!-- ================================================================= -->
    <!-- SUGGEST (INCLUDED IN THE DEFAULT SOLR SELECT REQUEST HANDLER)       -->
```

```
    <!-- ================================================================= -->

    <searchComponent name="spellcheck" class="solr.SpellCheckComponent">
    <str name="queryAnalyzerFieldType">title</str>
    <lst name="spellchecker">
      <str name="name">direct</str>
      <str name="field">title_suggest</str>
      <str name="classname">solr.DirectSolrSpellChecker</str>
      <str name="distanceMeasure">internal</str>
      <float name="accuracy">0.2</float>
      <int name="maxEdits">2</int>
      <int name="minPrefix">1</int>
      <int name="maxInspections">5</int>
      <int name="minQueryLength">3</int>
      <!--<float name="maxQueryFrequency">0.01</float>-->
    </lst>
    </searchComponent>

    <requestHandler name="/select" class="solr.SearchHandler"
    startup="lazy">
    <lst name="defaults">
      <!-- Solr Default Select Request Handler -->
      <str name="echoParams">explicit</str>
      <int name="rows">500</int>
      <!-- Suggest -->
      <str name="df">title_suggest</str>
      <str name="spellcheck.dictionary">direct</str>
      <str name="spellcheck">on</str>
      <str name="spellcheck.extendedResults">true</str>
      <str name="spellcheck.count">5</str>
      <str name="spellcheck.collate">true</str>
      <str name="spellcheck.collateExtendedResults">true</str>
    </lst>
    <arr name="last-components">
      <str>spellcheck</str>
    </arr>
    </requestHandler>
```

Solr Import Handler

solr.cfg:

```
[solr]
recipe = collective.recipe.solrinstance:mc
additional-solrconfig =
  <!-- Generate a unique key when creating documents in solr -->
  <requestHandler name="/update" class="solr.UpdateRequestHandler">
    <lst name="defaults">
      <str name="update.chain">uuid</str>
    </lst>
  </requestHandler>

  <!-- Generate a unique key when importing documents from csv in solr -->
  <requestHandler name="/update/csv" class="solr.UpdateRequestHandler">
    <lst name="defaults">
```

```
      <str name="update.chain">uuid</str>
    </lst>
  </requestHandler>

  <updateRequestProcessorChain name="uuid">
    <processor class="solr.UUIDUpdateProcessorFactory">
      <str name="fieldName">id</str>
    </processor>
    <processor class="solr.RunUpdateProcessorFactory" />
  </updateRequestProcessorChain>


[solr-geolocations-import]
recipe = collective.recipe.template
input = inline:
  #!/bin/sh
  # Delete all data
  curl http://${settings:solr-host}:${settings:solr-port}/solr/solr-core-geospatial/
↪update?commit=true -H "Content-Type: text/xml" --data-binary '<delete><query>*:*</
↪query></delete>'
  # Import data
  curl http://${settings:solr-host}:${settings:solr-port}/solr/solr-core-geospatial/
↪update/csv?commit=true --data-binary @etc/geolocations.csv -H 'Content-type:text/
↪csv; charset=utf-8'
output = ${buildout:directory}/bin/solr-geolocations-import
mode = 755
```

geolocations.csv:

```
"location","geolocation"
"01067 Dresden","51.057379, 13.715954"
"01069 Dresden","51.04931, 13.744873"
"01097 Dresden","51.060424, 13.745002"
...
```

Geospatial Search (with Autocomplete)

Works just when querying Solr directly. collective.solr needs some minor fixes. See https://github.com/collective/collective.solr/tree/spatial-filters.

solr.cfg:

```
[solr-core-geospatial]
max-num-results = 10
unique-key = id
index =
  name:id type:uuid indexed:true stored:true multivalued:false required:true
  name:location type:text indexed:true stored:true
  name:geolocation type:location indexed:true stored:true
  name:autocomplete type:text_auto indexed:true stored:true multivalued:true

additionalFieldConfig =
  <dynamicField name="*_coordinate"  type="tdouble" indexed="true"  stored="false"/>

extra-field-types =
  <fieldType name="uuid" class="solr.UUIDField" indexed="true" />
```

```
  <fieldType class="solr.TextField" name="text_auto">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ShingleFilterFactory" maxShingleSize="4" outputUnigrams=
↪"true"/>
      <filter class="solr.EdgeNGramFilterFactory" maxGramSize="20" minGramSize="1"/>
     </analyzer>
  </fieldType>

# Copy field city -> autocomplete
additional-schema-config =
  <copyField source="location" dest="autocomplete" />

additional-solrconfig =
  <!-- Generate a unique key when creating documents in solr -->
  <requestHandler name="/update" class="solr.UpdateRequestHandler">
    <lst name="defaults">
      <str name="update.chain">uuid</str>
    </lst>
  </requestHandler>

  <!-- Generate a unique key when importing documents from csv in solr -->
  <requestHandler name="/update/csv" class="solr.UpdateRequestHandler">
    <lst name="defaults">
      <str name="update.chain">uuid</str>
    </lst>
  </requestHandler>

  <updateRequestProcessorChain name="uuid">
    <processor class="solr.UUIDUpdateProcessorFactory">
      <str name="fieldName">id</str>
    </processor>
    <processor class="solr.RunUpdateProcessorFactory" />
  </updateRequestProcessorChain>

filter =
    text solr.LowerCaseFilterFactory
```

CHAPTER 5

collective.solr JSON API

Checkout json-api branch of collective.solr.

buildout.cfg:

```
[buildout]
...
extensions = mr.developer
auto-checkout = collective.solr

...

[sources]
collective.solr = git https://github.com/collective/collective.solr.git␣
↪pushurl=git@github.com:collective/collective.solr.git branch=json-api
```

Run buildout:

```
$ bin/buildout
```

Start Plone and Solr:

```
$ bin/instance fg
$ bin/solr-instance fg
```

JSON Search API

URL:

```
http://localhost:8080/Plone/@@search?format=json&SearchableText=Plone
```

Javascript:

```
GET http://localhost:8080/Plone/@@search?SearchableText=Plone
Accept: application/json
```

Response:

```
{
    "data":
[
        {
            "description": "",
            "id": "front-page",
            "portal_type": "Document",
            "title": "Willkommen bei Plone",
            "url": "http://localhost:8080/Plone/front-page"
        }
    ],
    "suggestions": [ ]
}
```

JSON Suggest API

Solr Configuration (solr.cfg):

```
[solr-instance]
recipe = collective.recipe.solrinstance
...
additional-solrconfig =

  <!-- ================================================================= -->
  <!-- SUGGEST (INCLUDED IN THE DEFAULT SOLR SELECT REQUEST HANDLER)     -->
  <!-- ================================================================= -->

  <searchComponent name="spellcheck" class="solr.SpellCheckComponent">
    <str name="queryAnalyzerFieldType">title</str>
    <lst name="spellchecker">

      <!-- The DirectSolrSpellChecker is a spell checker that doesn't require␣
↪building a separate, parallel index in order.
      https://wiki.apache.org/solr/DirectSolrSpellChecker -->

      <!--
          Optional, it is required when more than one spellchecker is configured.
          Select non-default name with spellcheck.dictionary in request handler.
      -->
      <str name="name">direct</str>

      <!--
          Load tokens from the following field for spell checking,
          analyzer for the field's type as defined in schema.xml are used
      -->
      <str name="field">Title</str>
      <str name="classname">solr.DirectSolrSpellChecker</str>

      <!-- the spellcheck distance measure used, the default is the internal␣
↪levenshtein -->
```

```
      <str name="distanceMeasure">internal</str>

      <!-- minimum accuracy needed to be considered a valid spellcheck suggestion -->
      <float name="accuracy">0.2</float>

      <!-- the maximum #edits we consider when enumerating terms: can be 1 or 2 -->
      <int name="maxEdits">2</int>

      <!-- the minimum shared prefix when enumerating terms -->
      <int name="minPrefix">1</int>

      <!-- maximum number of inspections per result. -->
      <int name="maxInspections">5</int>

      <!-- minimum length of a query term to be considered for correction -->
      <int name="minQueryLength">3</int>

      <!-- maximum threshold of documents a query term can appear to be considered␣
↪for correction -->
      <!--<float name="maxQueryFrequency">0.01</float>-->

      <!-- uncomment this to require suggestions to occur in 1% of the documents
        <float name="thresholdTokenFrequency">.01</float>
      -->

    </lst>
  </searchComponent>

  <!-- Include the suggest search component into the default '/select' request
       handler.

       See https://wiki.apache.org/solr/SpellCheckComponent#Request_Parameters for␣
↪all spellcheck component request parameters.
  -->

  <requestHandler name="/select" class="solr.SearchHandler"
  startup="lazy">
    <lst name="defaults">
      <!-- Solr Default Select Request Handler -->
      <str name="echoParams">explicit</str>
      <int name="rows">500</int>

      <!-- Suggest -->
      <str name="df">Title</str>

      <!-- The name of the spellchecker to use. -->
      <str name="spellcheck.dictionary">direct</str>

      <!-- Turn on or off spellcheck suggestions for this request. -->
      <str name="spellcheck">on</str>

      <!-- Provide additional information about the suggestion, such as the frequency␣
↪in the index. -->
      <str name="spellcheck.extendedResults">false</str>

      <!-- The maximum number of suggestions to return. -->
      <str name="spellcheck.count">5</str>
```

```
      <!-- A collation is the original query string with the best suggestions for␣
↪each term replaced in it. -->
      <str name="spellcheck.collate">false</str>

      <!-- If true, returns an expanded response format detailing collations found. --
↪>
      <str name="spellcheck.collateExtendedResults">false</str>

    </lst>
    <arr name="last-components">
      <str>spellcheck</str>
    </arr>
  </requestHandler>
```

URL:

http://localhost:8080/Plone/@@search?format=json&SearchableText=Plane

Javascript:

```
GET http://localhost:8080/Plone/@@search?SearchableText=Plane
Accept: application/json
```

Response:

```
{
    "data": [ ],
    "suggestions":
    {
        "plane":
        {
            "endOffset": 87,
            "numFound": 1,
            "startOffset": 82,
            "suggestion":
                [
                    "plone"
                ]
            }
        }
    }
}
```

JSON Autocomplete API

Solr Configuration (solr.cfg):

```
[solr-instance]
recipe = collective.recipe.solrinstance
...
name:title_autocomplete     type:text_autocomplete indexed:true stored:true

additional-schema-config =
  <!-- Additional field for autocomplete -->
  <copyField source="Title" dest="title_autocomplete" />
```

```
extra-field-types =
  <!-- Custom autocomplete filter for the autocomplete field -->
  <fieldType class="solr.TextField" name="text_autocomplete">
    <analyzer>

      <!-- Creates tokens of characters separated by splitting on whitespace. -->
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>

      <!-- Creates tokens by lowercasing all letters and dropping non-letters. -->
      <filter class="solr.LowerCaseFilterFactory"/>

      <!-- A ShingleFilter constructs shingles (token n-grams) from a token stream.␣
↪In other words, it creates combinations of tokens as a single token. For example,␣
↪the sentence "please divide this sentence into shingles" might be tokenized into␣
↪shingles "please divide", "divide this", "this sentence", "sentence into", and
↪"into shingles". -->
      <filter class="solr.ShingleFilterFactory" maxShingleSize="4" outputUnigrams=
↪"true"/>

      <!-- Create n-grams from the beginning edge of a input token: e.g.
      Nigerian => "ni", "nig", "nige", "niger", "nigeri", "nigeria", "nigeria",
↪"nigerian" -->
      <filter class="solr.EdgeNGramFilterFactory" maxGramSize="20" minGramSize="2"/>

    </analyzer>
  </fieldType>
```

URL:

```
http://localhost:8080/Plone/@@solr-autocomplete?term=Pl
```

Response:

```
[
    {
        "value": "Willkommen bei Plone",
        "label": "Willkommen bei Plone"
    }
]
```

CHAPTER 6

Solr Testing

collective.solr comes with a few test fixtures that make it easier to test Solr.

SOLR_FIXTURE fires up and tears down a Solr instance. This fixture can be used to write unit tests for a Solr configuration.

test_solr_unit.py:

```
# -*- coding: utf-8 -*-
from collective.solr.testing import SOLR_FIXTURE
import unittest2 as unittest
import json
import requests

SOLR_BASE_URL = 'http://localhost:8090/solr/collection1'


class TestSuggesetSolrConfig(unittest.TestCase):

    layer = SOLR_FIXTURE

    def setUp(self):
        self.clear()

    def clear(self):
        headers = {'Content-type': 'text/xml', 'charset': 'utf-8'}
        requests.post(
            SOLR_BASE_URL + "/update",
            data="<delete><select>*:*</select></delete>",
            headers=headers)
        requests.post(
            "http://localhost:8090/solr/update",
            data="<commit/>",
            headers=headers)

    def add(self, payload):
        headers = {'Content-type': 'application/json'}
```

```
        request = requests.post(
            SOLR_BASE_URL + "/update/json?commit=true",
            data=json.dumps(payload),
            headers=headers
        )
        if request.status_code != 200:
            print "FAILURE"

    def select(self, select):
        return requests.get(
            SOLR_BASE_URL + '/select?wt=json&q=%s' % select)

    def test_suggest(self):
        self.add([{
            "UID": "1",
            "Title": "Krebs",
            "SearchableText": "Krebs",
        }])
        response = self.select("Krabs")

        self.assertEqual(response.status_code, 200)
        self.assertEqual(
            response.json()['spellcheck']['suggestions'][1]['numFound'],
            1,
            "Number of found suggestions should be 1."
        )
        self.assertEqual(
            response.json()['spellcheck']['suggestions'][1]
            ['suggestion'][0]['word'],
            u'Krebs'
        )
```

COLLECTIVE_SOLR_FIXTURE fires up and tears down a Solr instance. In addition it activates and configures the collective.solr connection.

test_solr_integration.py:

```
# -*- coding: utf-8 -*-
from collective.solr.browser.interfaces import IThemeSpecific
from collective.solr.testing import COLLECTIVE_SOLR_INTEGRATION_TESTING
from collective.solr.utils import activate
from plone.app.testing import TEST_USER_ID
from plone.app.testing import setRoles
from zope.component import getMultiAdapter
from zope.interface import directlyProvides

import json
import unittest


class JsonSolrTests(unittest.TestCase):

    layer = COLLECTIVE_SOLR_INTEGRATION_TESTING

    def setUp(self):
        self.portal = self.layer['portal']
        self.request = self.layer['request']
        self.app = self.layer['app']
```

```
        self.portal.REQUEST.RESPONSE.write = lambda x: x    # ignore output
        self.maintenance = \
            self.portal.unrestrictedTraverse('@@solr-maintenance')
        activate()
        self.maintenance.clear()
        self.maintenance.reindex()
        directlyProvides(self.request, IThemeSpecific)
        setRoles(self.portal, TEST_USER_ID, ['Manager'])

    def tearDown(self):
        activate(active=False)

    def afterSetUp(self):
        self.maintenance = self.portal.unrestrictedTraverse('solr-maintenance')

    def beforeTearDown(self):
        pass

    def test_search_view_returns_plone_app_search_view(self):
        view = getMultiAdapter(
            (self.portal, self.request),
            name="search"
        )
        self.assertTrue(view)

    def test_search_view_with_json_accept_header(self):
        self.request.response.setHeader('Accept', 'application/json')
        view = getMultiAdapter(
            (self.portal, self.request),
            name="search"
        )
        view = view.__of__(self.portal)
        self.assertEqual(json.loads(view())['data'], [])
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search