SpellCheckerRequestHandler

The SpellCheckerRequestHandler is designed to process a word (or several words) as the value of the "q" parameter and returns a list of alternative spelling suggestions. The spellchecker used by this handler is the Lucene contrib SpellChecker. The SpellCheckerRequestHandler is still available in Solr, but is no longer recommended for use. See the SpellCheckComponent.

⚠️ Solr1.3

Term Source Configuration

When configuring the SpellCheckerRequestHandler in your SolrConfigXml, you should use the `termSourceField` config option to specify the field in your schema that you want to be able to build your spell index on. This should be a field that uses a very simple FieldType without a lot of Analysis (e.g. string):

```
<add>
  <doc>
    <field name="word">Accountant</field>
  </doc>
  <doc>
    <field name="word">Auditor</field>
  </doc>
  <doc>
    <field name="word">Solicitor</field>
  </doc>
</add>
```

In order to extract dictionary words from a field containing more than a single word (i.e. a text field), you should use the StandardTokenizer and StandardFilter which doesn't perform a great deal of processing on the field yet should provide acceptable results when used with the spell checker:

```
<fieldType name="spell" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
  </analyzer>
</fieldType>
```

⚠️ See SpellCheckingAnalysis for information on the importance of Analysis in Spell Checking. ⚠️

To automatically populate this field with the contents of another field when a document is added to the index, simply use a copyField:

```
<copyField source="content" dest="spell"/>
```

The default termSourceField is 'word'.

Core parameters

q

The word (or words) to be spell checked. **NOTE**: the way multiple words are handled depends on the setting of `sp.query.extendedResults`. If `false` (the default, compatible value), then words with spaces are treated as a single token to be spellchecked. If `true`, then each word is spellchecked separatedly.

qt

This must be set to 'spellchecker' in order to invoke the SpellCheckerRequestHandler

Dictionary-related parameters

termSourceField

(sp.dictionary.termSourceField in ⚠️ Solr1.3)

The field in your schema that you want to be able to build your spell index on. This should be a field that uses a very simple FieldType without a lot of Analysis (e.g. string):

```
<add>
  <doc>
    <field name="word">Accountant</field>
  </doc>
  <doc>
    <field name="word">Auditor</field>
  </doc>
  <doc>
    <field name="word">Solicitor</field>
  </doc>
</add>
```

In order to extract dictionary words from a field containing more than a single word (i.e. a text field), you should use the StandardTokenizer and StandardFilter which doesn't perform a great deal of processing on the field yet should provide acceptable results when used with the spell checker:

```
<fieldType name="spell" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
  </analyzer>
</fieldType>
```

To automatically populate this field with the contents of another field when a document is added to the index, simply use a copyField:

```
<copyField source="content" dest="spell"/>
```

The default field is 'word' and can be configured in SolrConfigXml.

spellcheckIndexDir

(sp.dictionary.indexDir in ⚠️ Solr1.3)

The directory where your spell checker index should live and defaults to 'spell' in SolrConfigXml. May be absolute or relative to the Solr "dataDir" directory. If this option is not specified, a RAM directory will be used.

sp.dictionary.threshold

Determines what terms will be used for creating the dictionary from the source field. The threshold is in terms of *document frequency*, i.e., what fraction of documents contain this term (not term frequency). This can be used to create a smaller, more accurate dictionary.

The default value is **`0`**. ⚠️ Solr1.3

cmd

There are currently two supported values for cmd: 'rebuild' and 'reopen':

In order to use SpellCheckerRequestHandler for the first time, you need to explicitly build the spelling index (see examples below):

If an external process is responsible for building the spell checker index, you must issue '&cmd=reopen' to force the spell checker index directory to be re-opened .

Query-related parameters

suggestionCount

(sp.query.suggestionCount in ⚠️ Solr1.3)

Determines how many spelling suggestions are returned. The default value is 1 but can be configured in SolrConfigXml. The order of the returned results is determined by both the Levenshtein distance (or accuracy) of the suggestion and the popularity (the frequency) of the suggested word in the termSourceField.

accuracy

(sp.query.accuracy in ⚠️ Solr1.3)

A float value between 1.0 and 0.0 on how close the suggested words should match the original word being checked (calculated using the Levenshtein distance algorithm). The default value is 0.5 but can be configured in SolrConfigXml.

onlyMorePopular

(sp.query.onlyMorePopular in ⚠️ Solr1.3)

When "onlyMorePopular" is set to true and the misspelled word exists in the user field, only words that occur more frequently in the termSourceField than the one given will be returned. The default value is false.

sp.query.extendedResults

Whether to use the extended response format, which is more complicated but richer. Returns the document frequency for each suggestion and returns one suggestion block for each term in the query string.

The default value is **`false`**. ⚠️ Solr1.3

Examples

```
Build the spelling index for the first time:
  http://localhost:8983/solr/select/?q=macrosoft&qt=spellchecker&cmd=rebuild

A simple call to the spell check handler:
  http://localhost:8983/solr/select/?q=windaws&qt=spellchecker

Return a list of suggestions that appear more frequently in the termSourceField that the word 'aft'
  http://localhost:8983/solr/select/?q=aft&qt=spellchecker&onlyMorePopular=true

Return 5 suggestions with a accuracy value of 0.7:
  http://localhost:8983/solr/select/?q=linix&qt=spellchecker&suggestionCount=5&accuracy=0.7
```

using extendedResults

Query: `q=pithon+programming&extendedResults=true&...`

```
<response>
    <lst name="responseHeader">
        <int name="status">0</int>
        <int name="QTime">173</int>
    </lst>
    <lst name="result">
        <lst name="pithon">
            <int name="frequency">5</int>
            <lst name="suggestions">
                <lst name="python">
                    <int name="frequency">18785</int>
                </lst>
            </lst>
        </lst>
        <lst name="progremming">
            <int name="frequency">0</int>
            <lst name="suggestions">
                <lst name="programming">
                    <int name="frequency">70997</int>
                </lst>
                <lst name="progressing">
                    <int name="frequency">1930</int>
                </lst>
                <lst name="programing">
                    <int name="frequency">597</int>
                </lst>
                <lst name="progamming">
                    <int name="frequency">113</int>
                </lst>
                <lst name="reprogramming">
                    <int name="frequency">344</int>
                </lst>
            </lst>
        </lst>
    </lst>
</response>
```

Example of the `extendedResults=true` output in python format:

```
{
  'responseHeader': {
    'status':0,
    'QTime':16
  },
  'result':{
    'pithon':{
      'frequency':5,
      'suggestions':['python',{'frequency':18785}]
    },
    'haus':{
      'frequency':482,
      'suggestions':['hats',{'frequency':6794},'hans',
{'frequency':5986},'haul',{'frequency':3152},'haas',
{'frequency':1054},'hays',{'frequency':533}]
    },
    'endication':{
      'frequency':0,
      'suggestions':['indication',{'frequency':9634},'syndication',
{'frequency':17777},'dedication',{'frequency':4470},'medication',
{'frequency':3746},'indications',{'frequency':2783}]
    }
  }
}
```

CategorySolrRequestHandler