

MultitermQueryAnalysis

<!-- Solr3.6 (!) Solr4.0

One of the surprises for most Solr users is that wildcard queries haven't gone through any analysis. Practically, this means that wildcard (and prefix and range) queries are case sensitive, which is at odds with expectations. As of this [SOLR-2438](#), [SOLR-2918](#), and perhaps [SOLR-2921](#), this behavior is changed.

What's a multiterm you ask? Essentially it's any term that may "point to" more than one real term. For instance, `run*` could expand to `runs`, `runner`, `running`, `run`, etc. Likewise, a range query is really a "multiterm" query as well. Before Solr 3.6, these were completely unprocessed, the application layer usually had to apply any transformations required, for instance lower-casing the input. Running these types of terms through a "normal" query analysis chain leads to all sorts of *interesting* behavior so was avoided.

- [New analyzer chain](#)
 - [Defining the chain](#)
 - [Why is this "arguably expert"?](#)
- [Auto-detection: A.K.A. "doing the right thing"](#)
- [Legacy behavior](#)
- [Current components that implement MultiTermAwareComponent](#)

New analyzer chain

The [schema.xml](#) file controls how fields are analyzed at index and query time. Now, you can optionally add a `<analyzer type="multiterm">` analysis chain. Don't worry! If you don't, Solr tries to "do the right thing" (see below). This arguably an expert-level option, and ***in most cases the new default behavior should be fine***, so don't define your own "multiterm" analysis chain unless you really have a need. You can string together any of the available or custom elements of a Solr analysis chain when defining this new chain.

Defining the chain

Just like there are two current analyzer chains, "index" and "query" (e.g.. `<analyzer type="index">`) there is now a third, optional analyzer `<analyzer type="multiterm">`. Just put it in the the `<fieldType>` as you would "index" or "query" types.

Why is this "arguably expert"?

Well, the assumption is that 95% of the time, all queries really need is lowercasing and, perhaps, accent folding. Those cases and the common [CharFilters](#) are already automatically applied, it seems likely that defining your own multiterm analysis chain will not be necessary. If this assumption is incorrect, feel free to raise a JIRA! All the assumptions in the world aren't worth a few real-world tests.

If you define your own chain, be aware that Solr will throw an exception if the chain you've defined evaluates to more than one term. Odd choice of phrase when it's called "multiterm", but it *operates* on multi-term queries, it does not *produce* multiple terms. So, for instance, if you define your own chain and put [WordDelimiterFilterFactory](#) in it and then send camel-case terms with wildcards, Solr will throw an exception when it analyzes the query.

And note that when defining your own multiterm analysis chain in the `schema.xml` file, none of the comments below about implementing [MultiTermAwareComponent](#) apply. Any component you place in the multiterm analyzer will be used as-is.

Auto-detection: A.K.A. "doing the right thing"

Solr tries to take the pain out of creating another analysis chain when the behavior is predictable. Solr does this by looking at what has been defined and picking things out of that chain that "make sense". These are used to build the "multiterm" analyzer without anything needing to be specified. So you don't have to do *anything* to get this capability.

During schema file analysis, if there is no `<analyzer type="multiterm">` defined, Solr will construct it from the first of `<analyzer type="query">` or `<analyzer type="index">` or `<analyzer>`, in that order. The new analyzer will consist of any component of the analysis chain that implements [MultiTermAwareComponent](#). This new analyzer is then applied to all multiterm terms. Internally to Solr, any component that implements [MultiTermAwareComponent](#) are required to "do the right thing" to play nicely as part of the multiterm analysis chain.

Legacy behavior


If you require the old behavior, you can specify a multiterm analysis chain that consists of only a [KeywordTokenizerFactory](#) as part of your `<fieldType>` definition like this:


```
<fieldType ....>
  <index and query analyzers here>
  <analyzer type="multiterm">
    <tokenizer class="solr.KeywordTokenizerFactory" />
  </analyzer>
</fieldType>
```

Current components that implement [MultiTermAwareComponent](#)

Here's a list as of 27-Nov-2011. You can always determine what the current list is by finding classes that implement [MultiTermAwareComponent](#) (as it's currently named).

- [ASCIIFoldingFilterFactory](#)
- [LowerCaseFilterFactory](#)
- [LowerCaseTokenizerFactory](#)
- [MappingCharFilterFactory](#)
- [PersianCharFilterFactory](#)

 [Solr3.6](#) Note that [ISOLatin1AccentFilterFactory](#) is NOT [MultiTermAware](#). You should be using [ASCIIFoldingFilterFactory](#) instead as [ISOLatin1AccentFilterFactory](#) is deprecated.

 [Solr4.0](#) Here are some additional classes that implement [MultiTermAwareComponent](#) ==

- [ASCIIFoldingFilterFactory](#)
- [ArabicNormalizationFilterFactory](#)
- [CJKWidthFilterFactory](#)
- [CollationKeyFilterFactory](#)
- [ElisionFilterFactory](#)
- [GermanNormalizationFilterFactory](#)
- [GreekLowerCaseFilterFactory](#)
- [HindiNormalizationFilterFactory](#)
- [ICUCollationKeyFilterFactory](#)
- [ICUFoldingFilterFactory](#)
- [ICUNormalizer2FilterFactory](#)
- [ICUTransformFilterFactory](#)
- [IndicNormalizationFilterFactory](#)
- [IrishLowerCaseFilterFactory](#)
- [JapaneseIterationMarkCharFilterFactory](#)
- [LowerCaseFilterFactory](#)
- [MappingCharFilterFactory](#)
- [PersianCharFilterFactory](#)
- [PersianNormalizationFilterFactory](#)
- [TurkishLowerCaseFilterFactory](#)

Here's a longer blog on this subject [Blog Post](#)

 :TODO: 

There are tantalizing possibilities for other factories that could be made multiterm-aware, but they are beyond the scope of this initial implementation. See SOLR-2921.