Schema Facet

Overview

SchemaFacetRequest objects provide the ability to determine which fields are populated for the documents matching a query. For example, using the Quick Start Tutorial Factbook example, a SchemaFacetRequest could be used to determine which schema fields are present in country documents.

View incoming links. >>

Facets , HTTP REST APIs

- Overview
- Schema Facet Example
- Requesting Schema Facets
    - Java Client API
        - Examples (Java)
            - Basic Example
            - Only request fields with facet = true
            - Request counts for specified fields
- JSON REST API
    - XML REST Syntax
        - Syntax
        - Parameters
    - REST API Examples
        - SortFields (and similar)
        - SchemaField
        - CountValues

Schema Facet Example

This is an example of a schema facet as displayed in SAIL. Use the following search URL:

```
http://localhost:17000/sail/search?q.type=simple&q=*:*&facet=.fields(maxBuckets=100)
```

The result is a facet that lists the stored fields in the matching result set, and indicates how many documents have a value for each field. This example was run against the Factbook query (from the Quick Start Tutorial) using only the data for the 2587 cities.

The facet shows that nearly every field is populated in nearly every city record, but down at the bottom we can see that a few cities do not have **location** values, and that **keyphrases**, **company** and **people** values are quite rare in this dataset.

There are additional REST API examples at the bottom of this page.

Requesting Schema Facets

The following sections describe how to request a schema facet using the Java and REST Query APIs.

> ⚠ Multiple SchemaFacetRequests can be requested, however each one must be given a different name in order to be retrievable in the QueryResponse.

Java Client API

Schema Facets can be requested via the Java Client API using the SchemaFacetRequest .

Examples (Java)

Basic Example

```
SearchClient client = ...;
QueryRequest query = new QueryRequest("table:documents");

// Add the SchemaFacetRequest to the query
query.addFacet( new SchemaFacetRequest("schema") );

// Get the response
QueryResponse results = client.search(query);

// Display the fields
System.out.println("Fields:");
for (FacetBucket bucket : results.getFacet("schema")) {
  System.out.printf("%s - %d\n", bucket.getLabel(), bucket.getCount());
}
```

Only request fields with facet = true

```
SearchClient client = ...;
QueryRequest query = new QueryRequest("table:documents");

// Add the SchemaFacetRequest to the query
SchemaFacetRequest facet = new SchemaFacetRequest("facets");
facet.setFacetFields(true);
facet.setSortFields(false);
facet.setIndexedFields(false);
facet.setStoredFields(false);
query.addFacet(facet);

// Get the response
QueryResponse results = client.search(query);

// Display the fields
System.out.println("Facet Fields:");
for (FacetBucket bucket : results.getFacet("facets")) {
  System.out.printf("%s - %d\n", bucket.getLabel(), bucket.getCount());
}
```

Request counts for specified fields

```
SearchClient client = ...;
QueryRequest query = new QueryRequest("table:documents");

// Add the SchemaFacetRequest to the query
SchemaFacetRequest facet = new SchemaFacetRequest("fields");
facet.addSchemaField("title");
facet.addSchemaField("text");
query.addFacet(facet);

// Get the response
QueryResponse results = client.search(query);

// Display the fields
System.out.println("Fields:");
for (FacetBucket bucket : results.getFacet("facets")) {
  System.out.printf("%s - %d\n", bucket.getLabel(), bucket.getCount());
}
```

JSON REST API

See the REST API Tutorial for examples showing how to issue a JSON REST query.

See the REST Query API page for details of composing JSON REST queries.

The schema-facet functionality is available in the **.fields** virtual field, as shown in the example below.

```
{
     "workflow": "search",
     "query": "*:*",
     "queryLanguage": "advanced",
     "locale": "en",
     "rows": 10,
     "facets" : ["populatedFields(field=.fields, sortBy=VALUE, primarySort=ASC, minBucketCount=1)"],
     "fields": [ ".id", "title", "text", "date" ]
}
```

XML REST Syntax

A schema facet can be requested via the XML REST API using the **.fields** virtual field.

Syntax

```
&facet=.fields[(parameter1=value1[, parameter2=value2[, ...]])]
```

Parameters

Parameter	Type	Default Value	Description
realtimeFields	boolean	false	If true, realtime schema fields (defined using the `realtimeField` element instead of the `field` element) will be returned
storedFields	boolean	true	If true, fields marked as `stored=true` will be returned
indexedFields	boolean	true	If true, fields marked as `index=true` will be returned
facetFields	boolean	true	If true, fields marked as `facet=true` will be returned
sortFields	boolean	true	If true, fields marked as `sort=true` will be returned
countBuckets	boolean	true	If document frequencies are not needed, setting this to `false` will result in faster computation.
countValues	boolean	true	If true buckets will contain value counts in addition to document counts. NOTE: Setting this to true will result in slower facet computation.
schemaField	String	*optional*	Request specific fields to calculate buckets for. (can be specified multiple times)
sortBy	enum (COUNT, VALUE)	COUNT	See Facet Bucket Sorting
primarySort	enum (ASC, DESC)	DESC	See Facet Bucket Sorting
secondarySort	enum (ASC, DESC)	ASC	See Facet Bucket Sorting
maxBuckets	integer	10	See Bucket Filtering
minBucketCount	integer	1	See Bucket Filtering. Count must be > 0.
distributedMaxBuckets	integer	-1 (unlimited)	See Bucket Filtering
distributedMinBucketCount	integer	1	See Bucket Filtering
statistics	boolean	false	See Facet Statistics

REST API Examples

This section demonstrates a few REST API examples and the facet lists they produce when run against country data from the Factbook exercise.

SortFields (and similar)

This example creates a facet of sortable fields (defined in the AIE Schema as **sort="true"**). The example also works for realtimeFIelds, facetFields, indexedFields, and storedFields. Just set the true/false values appropriately.

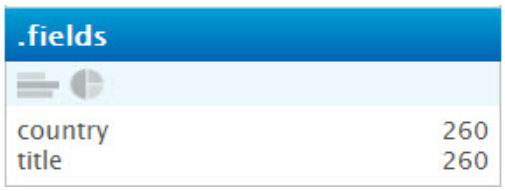
```
http://host:17000/sail/search?q.type=simple&q=*:*&facet=myFacetLabel(field=.fields,realtimeFields=false,
sortFields=true,facetFields=false,
indexedFields=false,storedFields=false)
```

The facet is a list of the sortable fields in a Factbook country document. Note that **myFacetLabel** will be the title of the displayed facet. Since the field types all default to true, it is essential to set the field types you *don't* want to false.

SchemaField

The schemaField parameter lets us specify exactly which fields should be returned in the facet. This is a request for a facet that reports about the title and country fields only.

```
http://host:17000/sail/search?q.type=simple&q=*:*&facet=.fields(schemaField=title,schemafield=country)
```



CountValues

If you set countValues=true, the raw facet description will contain counts of values in addition to counts of documents.

```
http://host:17000/sail/search?q.type=simple&q=*:*&facet=.fields(countValues=true)
```

You can see the value counts in the Legacy XML display of the Debug Search UI.

```
<facet field=".fields" count="9026" time="9">
   <schemaBucket count="260" values="680">climate</schemaBucket>
   <schemaBucket count="260" values="10954">content</schemaBucket>
   <schemaBucket count="260" values="260">conversionErrorCode</schemaBucket>
   <schemaBucket count="260" values="260">country</schemaBucket>
   <schemaBucket count="260" values="260">date</schemaBucket>
   <schemaBucket count="260" values="260">economy</schemaBucket>
   <schemaBucket count="260" values="260">filename</schemaBucket>
   <schemaBucket count="260" values="260">language</schemaBucket>
   <schemaBucket count="260" values="260">languages</schemaBucket>
   <schemaBucket count="260" values="1298">location</schemaBucket></facet>
```