# Unit 9: Faceted Search

# Learning Objectives

- Describe faceted search and its prerequisites.
- Calculate facet values and return them in the search response.
- Work with facet values in your application code.
- Create a bucketed constraint.

MarkLogic

# Examples of Facets

MarkLogic

Samplestack

Log In

Ask a Question

Search the Stack!...

Search

Search Tips

Filter Results

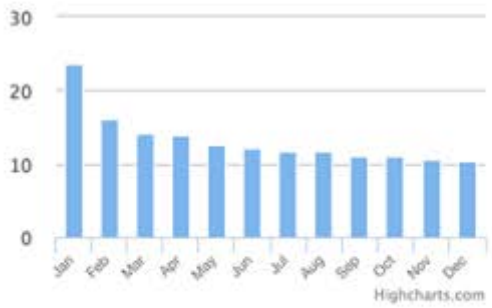
Clear All

Show my contributions only

Show resolved only

Filter By Date

Clear Dates




to

Tags

Clear Tags

Search Tags

angularjs (1092)

jquery (1229)

Browsing 27,348 Questions

Relevance | Newest | Votes

VOTES 1,267 | 2 ANSWERS

What does the yield keyword do in Python?

Python | Javascript | Node.js

Asked 21s ago by Sowmyashree Vasu

VOTES 1,267 | 2 ANSWERS

node.js + nginx - And now?

Javascript | Node.js | Nginx

Asked 27s ago by Squat Chimo

VOTES 1,267 | 2 ANSWERS

What is the purpose of Node.js module.exports and how do you use it?

Javascript | Node.js

Asked 54s ago by Vortico

VOTES 1,267 | 2 ANSWERS

How do I get PyCharm to show method signatures and documentation in the Python/IPython console and the editor?

Javascript | Node.js | Python | IPython | PyCharm

Asked 34s ago by habib_101

VOTES 1,267 | 2 ANSWERS

Node.js on multi-core machines?

Javascript | Node.js

Asked 54s ago by Vortico

VOTES 1,267

How do you extract POST data in node.js?

# Facet Prerequisites

- Create a Range Index

**range element index** -- *An index for fast element inequality comparisons.* delete

**scalar type**
string
An atomic type specification.

**namespace uri**
http://marklogic.com/MLU/top-songs
A namespace URI.

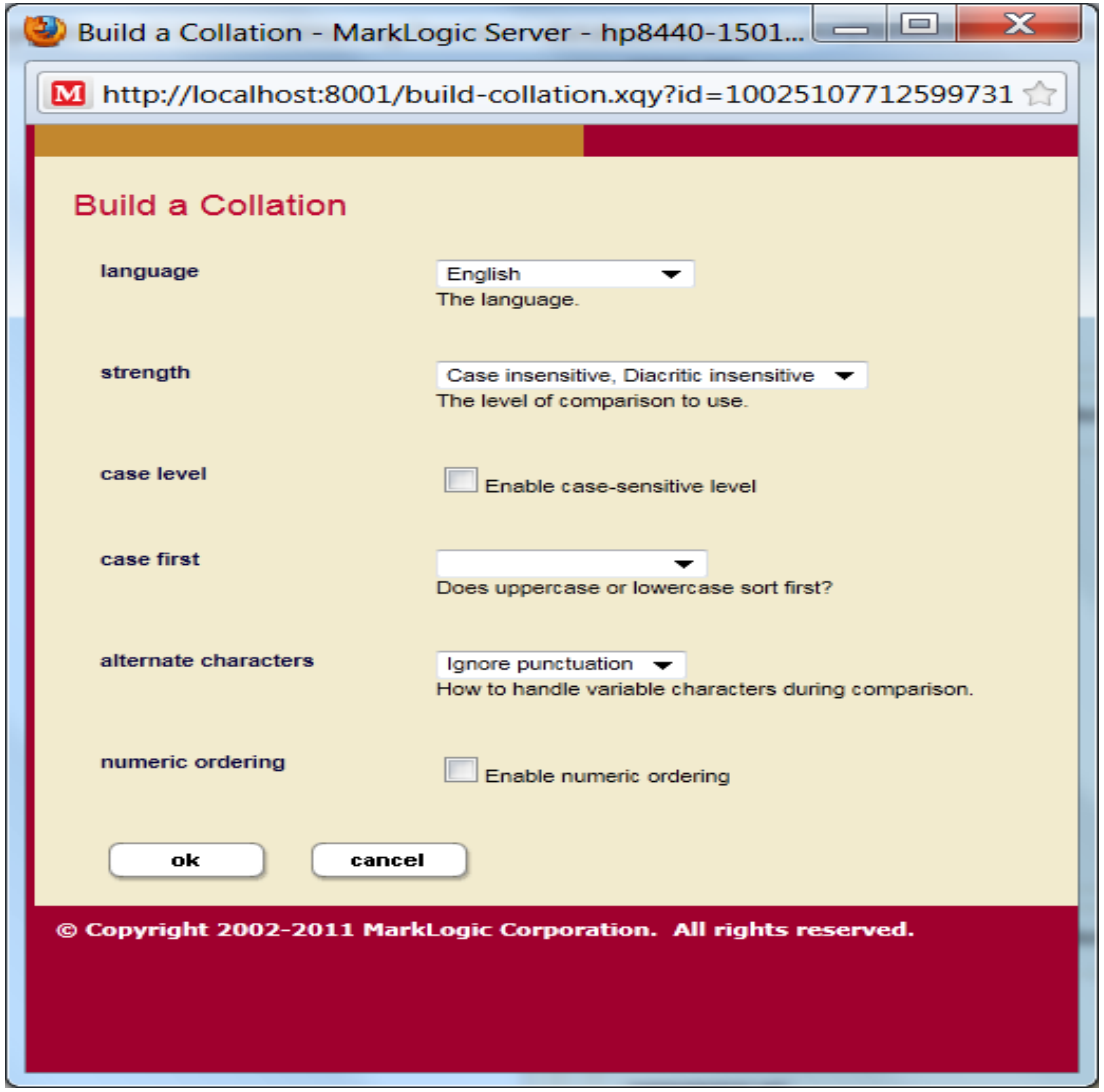
**localname**
artist
One or more localnames.

**collation**
arklogic.com/collation/en/S1/AS/T00BB collation builder
A collation URI for string comparisons.

**range value positions**
true false
Index range value positions for faster near searches involving range queries (slower document loads and larger database files).

# Facet Prerequisites

- Create a Collation
- For string range indexes
- Governs string comparisons
  - Doc 1:
    - <artist>The Beatles</artist>
  - Doc 2:
    - <artist>the beatles</artist>
  - What data will the range index on <artist> contain?



# Get Facet Data

- You'll need a query builder instance:

```
var qb = marklogic.queryBuilder;
```

- You'll need your database clients:

```
var dbRead = marklogic.createDatabaseClient(dbConn.restReader);
var dbWrite = marklogic.createDatabaseClient(dbConn.restWriter);
var dbAdmin = marklogic.createDatabaseClient(dbConn.restAdmin);
```

- Assume these things are in place for the next few code examples.

# Get Facet Data: The Full Response

- Query:

```
var qText = "coldplay";

dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    )
    .calculate(
      qb.facet("artist")
    )
    .slice(1, 2, qb.extract({
        paths: ["//artist", "//title"]
      })
    )
).result( function(responseData) {
  console.log("----FULL RESPONSE DATA----");
  console.log(responseData);
});
```

- Response:

```
----FULL RESPONSE DATA----
[ { 'snippet-format': 'empty-snippet',
    total: 9,
    start: 1,
    'page-length': 2,
    results: [],
    facets: { artist: [Object] } },
  { uri: '/songs/Coldplay+Viva-la-Vida.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '122',
    content:
     { context: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")',
       extracted: [Object] } },
  { uri: '/songs/Katy-Perry+I-Kissed-a-Girl.json',
    category: 'content',
    format: 'json',
    contentType: 'application/json',
    contentLength: '132',
    content:
     { context: 'fn:doc("/songs/Katy-Perry+I-Kissed-a-Girl.json")',
       extracted: [Object] } } ]
```

- The full response is an array.
  - Item[0] contains the result summary and facet data.

# Get Facet Data: Digging Deeper

- Query:

```
console.log(responseData[0].facets);
```

- Response:

```
{ artist:
   { type: 'xs:string',
     facetValues:
      [ [Object],
        [Object],
        [Object],
        [Object],
        [Object],
        [Object],
        [Object],
        [Object],
        [Object] ] } }
```

```
console.log(responseData[0].facets.artist.facetValues);
```

```
[ { name: 'Beyoncé featuring Jay-Z',
    count: 1,
    value: 'Beyoncé featuring Jay-Z' },
  { name: 'Coldplay', count: 1, value: 'Coldplay' },
  { name: 'Jay Sean featuring Lil Wayne',
    count: 1,
    value: 'Jay Sean featuring Lil Wayne' },
  { name: 'Katy Perry', count: 1, value: 'Katy Perry' },
  { name: 'Michael Jackson', count: 1, value: 'Michael Jackson' },
  { name: 'Nelly', count: 1, value: 'Nelly' },
  { name: 'Nelly Furtado', count: 1, value: 'Nelly Furtado' },
  { name: 'OutKast', count: 1, value: 'OutKast' },
  { name: 'Ray Charles', count: 1, value: 'Ray Charles' } ]
```

MarkLogic

# Get Facet Counts

- Query:

```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    )
    .calculate(
      qb.facet("artist")
    )
    .slice(1, 2, qb.extract({
        paths: ["//artist", "//title"]
      })
    )
).result( function(responseData) {

    responseData[0].facets.artist.facetValues.forEach(function(facet) {
    console.log(facet.value + " (" + facet.count + ")");

  });
});
```

- Response:

```
Beyoncé featuring Jay-Z (1)
Coldplay (1)
Jay Sean featuring Lil Wayne (1)
Katy Perry (1)
Michael Jackson (1)
Nelly (1)
Nelly Furtado (1)
OutKast (1)
Ray Charles (1)
```

MarkLogic

# Facet Options

- Customize the response data:
  - `marklogic.queryBuilder.facetOptions`

```
var qText = "the beatles";

dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    )
    .calculate(
      qb.facet("artist", qb.facetOptions("frequency-order", "descending", "limit=10"))
    )
    .slice(1, 2, qb.extract({
        paths: ["//artist", "//title"]
      })
    )
).result( function(responseData) {
  responseData[0].facets.artist.facetValues.forEach(function(facet) {
    console.log(facet.value + " (" + facet.count + ")");
  });
});
```

```
The Beatles (19)
George Harrison (3)
Michael Jackson (2)
Whitney Houston (2)
"Honeycomb" Jimmie Rodgers (1)
"Learning the Blues" Frank Sinatra (1)
"Round and Round" Perry Como (1)
#cite_note-34[35] (1)
Bobby Goldsboro (1)
Bobby Vee (1)
```

# Bucketed Constraints

- Range queries with lower and upper bounds.
- Counts of documents that fall within defined ranges.

```
.calculate(
  qb.facet(
    "price",
    qb.bucket("0 to 10", "0", "<", "10"),
    qb.bucket("10 to 20", "10", "<", "20"),
    qb.bucket("20 to 30", "20", "<", "30"),
    qb.bucket("30 and up", "30", "<")
  )
)
```

# Debugging Facets

- First thing to always check:
  - Do I have a range index backing this up?
- Second thing to check:
  - Do I have the range index **exactly** as I'm specifying in my facet definition?
    - Correct data type?
    - If string range index, is collation correct?
- Example of trying to get facet results without a backing range index:

```
Error: query documents: response with invalid 400 status
```

# Labs: Unit 9

Exercise 1: Return and Process Facet Data
Exercise 2: Customize Facet Response Data
Exercise 3: Create a Second Facet
DIY: Create a Bucketed Constraint

# Unit Review Question 1:

These range indexes, defined on the same database, are the same and therefore interchangeable:

1. True
2. False

**scalar type** string
An atomic type specification.

**namespace uri**
A namespace URI.

**localname** artist
One or more localnames.

**collation** http://marklogic.com/collation/en/S1/AS/T00BB
collation builder
A collation URI for string comparisons.

**range value positions** true false
Index range value positions for faster near searches involving range queries (slower document loads and larger database files).

**invalid values** reject
Allow ingestion of documents that do not have matching type of data.

**scalar type** string
An atomic type specification.

**namespace uri**
A namespace URI.

**localname** artist
One or more localnames.

**collation** http://marklogic.com/collation/ Root Collation
collation builder
A collation URI for string comparisons.

**range value positions** true false
Index range value positions for faster near searches involving range queries (slower document loads and larger database files).

**invalid values** reject
Allow ingestion of documents that do not have matching type of data.

# Unit Review Question 1:

These range indexes, defined on the same database, are the same and therefore interchangeable:

1. True
2. **False**

**scalar type** string ▼
An atomic type specification.

**namespace uri**
A namespace URI.

**localname** artist
One or more localnames.

**collation** http://marklogic.com/collation/en/S1/AS/T00BB
collation builder
A collation URI for string comparisons.

**range value positions** true false
Index range value positions for faster near searches involving range queries (slower document loads and larger database files).

**invalid values** reject ▼
Allow ingestion of documents that do not have matching type of data.

**scalar type** string ▼
An atomic type specification.

**namespace uri**
A namespace URI.

**localname** artist
One or more localnames.

**collation** http://marklogic.com/collation/ Root Collation ▼
collation builder
A collation URI for string comparisons.

**range value positions** true false
Index range value positions for faster near searches involving range queries (slower document loads and larger database files).

**invalid values** reject ▼
Allow ingestion of documents that do not have matching type of data.

# Unit Review Question 2:

Facet values are returned from memory and are very fast:

1. True
2. False

# Unit Review Question 2:

Facet values are returned from memory and are very fast:

1. **True – remember, range indexes live in memory.**
2. False