MarkLogic UNIVERSITY

Unit 10

Search Applications: Snippets, Highlights, Sorting and Pagination

Implement Search Snippets

Customize Search Snippets

Implement Sorting

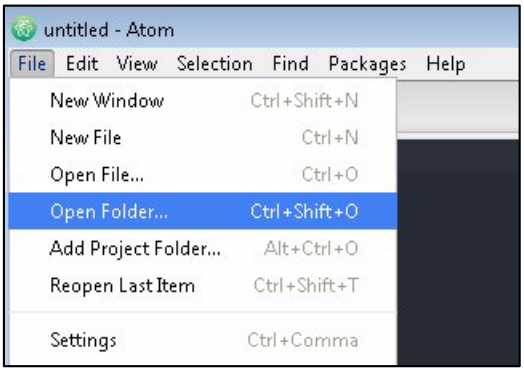
The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.

Exercise 1: Implement Search Snippets

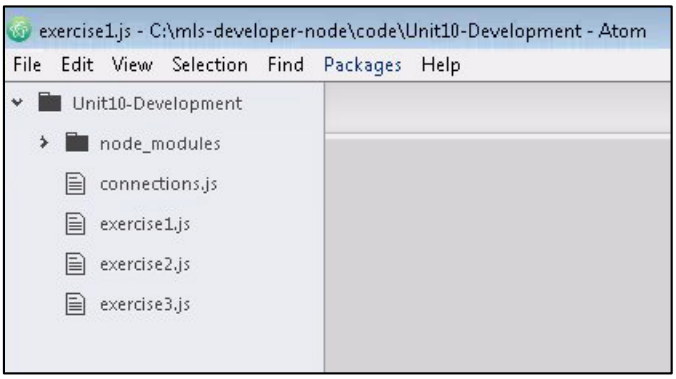
In this exercise you will implement a search that will return the default snippet data and you will work with that data in the response.

1. From the command line, navigate to the **c:\mls-developer-node\code\Unit10-Development** directory and run **npm install marklogic**.
2. Open the Atom editor from your Desktop:

3. In the Atom editor select File→Open Folder…:



4. Open the **Unit10-Development** folder from c**:\mls-developer-node\code**
5. You should see the following structure in place in your editor:



6. Select **exercise1.js**.

The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.

7. Take a moment to study the comments and code.
8. Note that in the query you are using the query builder to indicate that you desire default snippet data to be returned:

```
var qText = "coldplay";

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

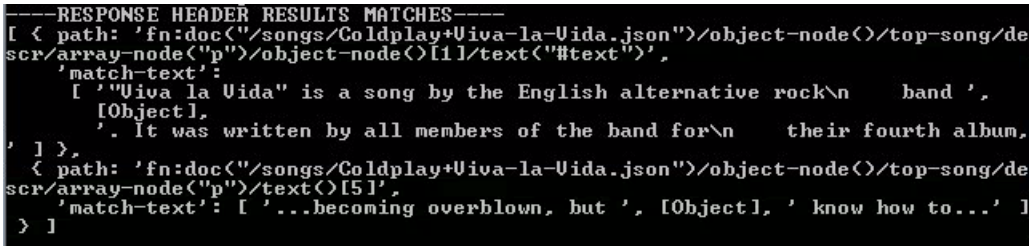
9. Note that you will then analyze the response data bit by bit to understand what it contains:

```
).result( function(responseData) {
  // output the full response data
  console.log("----FULL RESPONSE----")
  console.log(responseData);

  // output the response header
  console.log("----RESPONSE HEADER----")
  console.log(responseData[0])

  // output the results from the header
  console.log("----RESPONSE HEADER RESULTS----")
  console.log(responseData[0].results)

  // output the results from the header
  console.log("----RESPONSE HEADER RESULTS MATCHES----")
  console.log(responseData[0].results[0].matches)

});
```

10. Now let's test the code.
11. At the command line, go to the **c:\mls-developer-node\Unit10-Development** directory.
12. Enter **node exercise1.js** and press enter.
13. Take some time to studythe response data.

The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.

14. Note that the snippet data is in the first item in the array of response data that is returned. Drilling into that response data, we start to see the actual snippets here:



The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.

Exercise 2: Customize Search Snippets

In this exercise you will implement code that gives you control over how the snippet data is returned in the response.

1. In your editor, within the **Unit10-Development** project, select **exercise2.js**.
2. Study the comments and code.
3. Note that you are passing in JSON data that tells the query builder exactly how you want the snippet. For example, this code says to return snippets with at most 200 characters or 30 words, to return a maximum of 3 snippets, and to return snippets from the "descr" property:

```
qb.snippet(
  { "max-snippet-chars": 200,
    "per-match-tokens": 30,
    "max-matches": 3,
    "preferred-matches": "descr"
  })
```

4. Now let's test the code.
5. At the command line, go to the **c:\mls-developer-node\Unit10-Development** directory.
6. Enter **node exercise2.js** and press enter.
7. Take some time to study the response data:

```
c:\mls-developer-node\code\Unit10-Development>node exercise2.js
----RESPONSE HEADER RESULTS MATCHES----
[ { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/object-node()[1]/text("#text")',
    'match-text':
     [ '"Viva la Vida" is a song by the English alternative rock\n    band ',
       [Object],
       '. It was written by all members of the band for\n    their fourth album,
' ] },
  { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/text()[5]',
    'match-text': [ '...becoming overblown, but ', [Object], ' know how to...' ]
 } ]
```

8. Now let's change a snippet parameter and see the effect. Change "max-matches" to equal 1:

```
"max-matches": 1,
```

9. Save your code and test. Notice you now receive 1 "match-text" property in the response:

```
----RESPONSE HEADER RESULTS MATCHES----
[ { path: 'fn:doc("/songs/Coldplay+Viva-la-Vida.json")/object-node()/top-song/de
scr/array-node("p")/object-node()[1]/text("#text")',
    'match-text':
     [ '"Viva la Vida" is a song by the English alternative rock\n    band ',
       [Object],
       '. It was written by all members of the band for\n    their fourth album,
' ] } ]
```


The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.

Exercise 3: Implement Sorting

In this exercise you will implement code that enables you to sort the result set.

1. In your editor, within the **Unit10-Development** project, select **exercise3.js**.
2. Study the comments and code.
3. Note the **.orderBy** and the defining of the sort order using **qb.sort**:

```
dbRead.documents.query(
    qb.where(
      qb.parsedFrom(qText)
    ).orderBy(qb.sort("artist", "ascending"))
    .slice(1, 10, qb.extract({
        paths: ["//artist", "//title"]
      }))
```

4. Now let's test the code.
5. At the command line, go to the **c:\mls-developer-node\Unit10-Development** directory.
6. Enter **node exercise3.js** and press enter.
7. Take some time to study the response data:

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

8. Now let's change the order to descending:

```
qb.sort("artist", "descending")
```

9. Save your code and test, noticing the impact to the response data:

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


The information contained in this Course is copyrighted and all rights are reserved by MarkLogic Corporation. Copying, duplicating, selling, or otherwise distributing any part of the Course without prior written consent of an authorized representative of MarkLogic Corporation are prohibited.