Unit 8

Indexing

Modify a Database Configuration

Build a Range Index

Automate Index Deployment with the Management REST API

Capture a Database Configuration

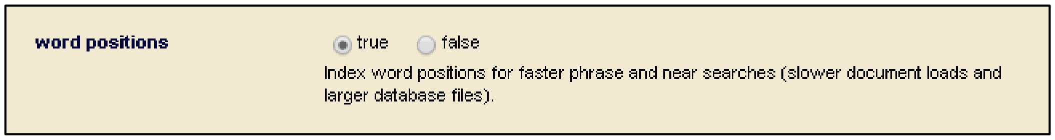
DIY: Setup Star Wars Indexes

Exercise 1: Modify a Database Configuration

In this exercise you will turn on some additional term list indexes in order to support accurate unfiltered search queries against your database.

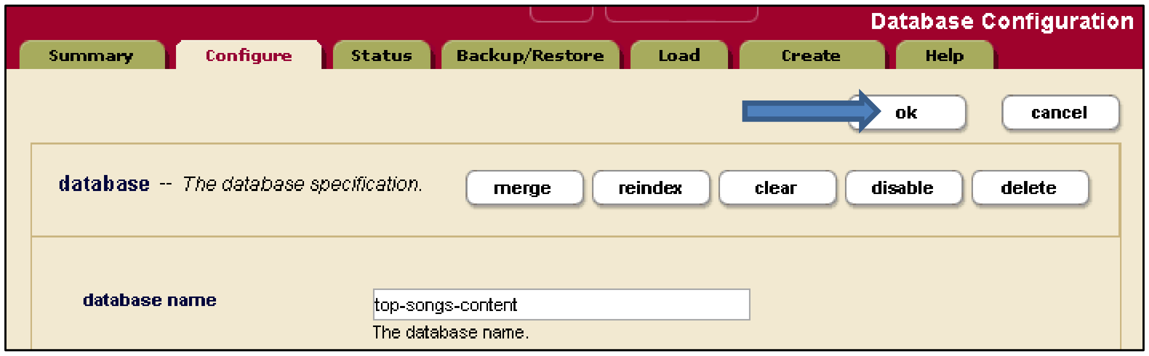
Specifically, you wish to be able to perform near queries and be able to return results accurately from the indexes (unfiltered search).

1. In a browser, open the Admin interface at http://localhost:8001.
2. From the left hand navigation column click **Configure→Databases→top-songs-content**.
3. Scroll down the configuration until you find the **word positions** index and set it to **true**:



word positions [✓ true] false
Index word positions for faster phrase and near searches (slower document loads and larger database files).

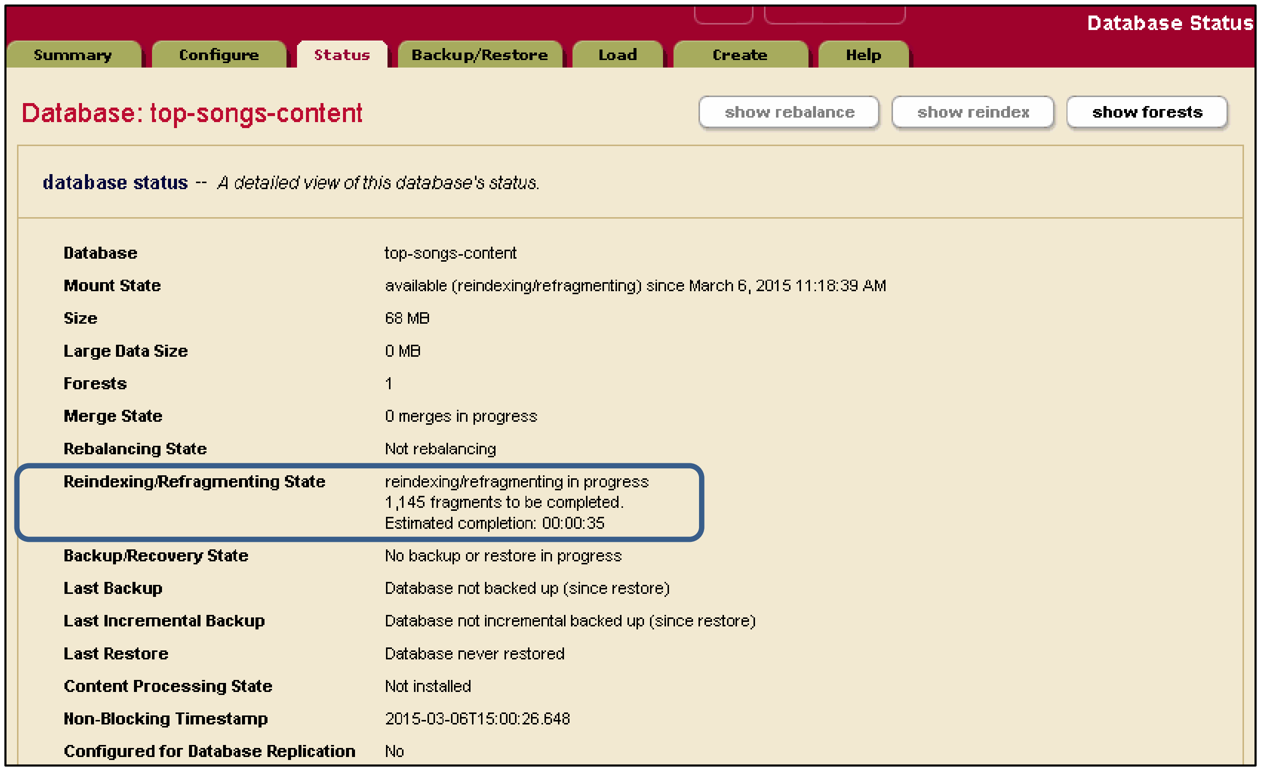
4. Scroll back to the top of the page and click **OK**:



database -- *The database specification.*

database name: top-songs-content
The database name.

5. This will cause a database reindex to begin. To track the progress, click on the **Status** tab.
6. If a reindex is currently ongoing, you will see a status similar to the following:

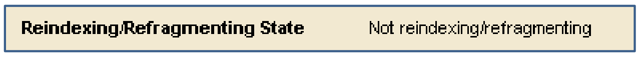


Database: top-songs-content

database status -- *A detailed view of this database's status.*

Database	top-songs-content
Mount State	available (reindexing/refragmenting) since March 6, 2015 11:18:39 AM
Size	68 MB
Large Data Size	0 MB
Forests	1
Merge State	0 merges in progress
Rebalancing State	Not rebalancing
Reindexing/Refragmenting State	reindexing/refragmenting in progress 1,145 fragments to be completed. Estimated completion: 00:00:35
Backup/Recovery State	No backup or restore in progress
Last Backup	Database not backed up (since restore)
Last Incremental Backup	Database not incremental backed up (since restore)
Last Restore	Database never restored
Content Processing State	Not installed
Non-Blocking Timestamp	2015-03-06T15:00:26.648
Configured for Database Replication	No

7. Wait a few seconds and then refresh your browser. When the reindex is complete, you will see the following status:



Reindexing/Refragmenting State	Not reindexing/refragmenting

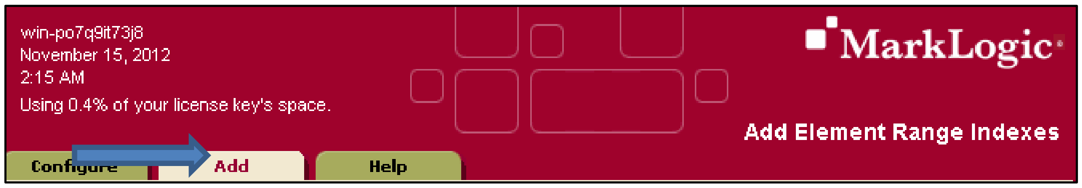
Exercise 2: Build a Range Index

In this exercise we will use the Admin interface to build a string type range index on the artist property.

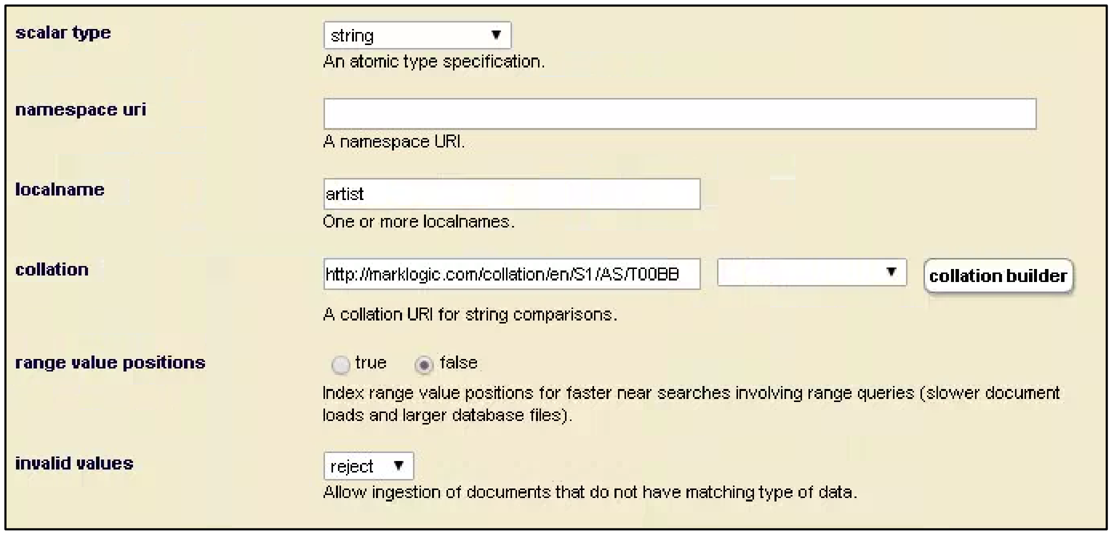
1. Open the Admin interface (http://localhost:8001)
2. Select **Configure→Databases→top-songs-content→Element Range Indexes**

Note:
Element range indexes apply for both XML elements and JSON properties. Because JSON documents don't have the concept of namespaces, when setting up an index on the property you will simply leave the namespace empty.

3. Click the **Add** tab to create a new range index as shown:


win-po7q9it73j8
November 15, 2012
2:15 AM
Using 0.4% of your license key's space.
MarkLogic
Add Element Range Indexes
Configure
Add
Help

4. Configure the range index as follows:
    a. Scalar type = string
    b. Namespace URI = leave this empty
    c. Localname = artist
    d. Collation = http://marklogic.com/collation/en/S1/AS/T00BB
    e. Range value positions = false
    f. Invalid values = reject
5. Your index should appear as follows:



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 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.

6. Click **OK** to create the index.

Question:
When are indexes created?
Answer:
Indexes are created in real time as documents are ingested into MarkLogic. When a document write (or update) commits, that document and its indexes are ready for action.

Question:
Since our documents have already been loaded, what happens when we create new indexes?
Answer:
This will cause the database to reindex if the database has reindexing turned on. For the scope of our sample application, this is not a large cost. But at scale, the cost could be significant. You can see the status of a reindex operation using the Admin tool on the database status page.

Exercise 3: Automate Index Deployment with the Management REST API

In the prior exercise, we created a single range index by hand using the Admin interface. This is a valid way to work, and in fact may be the way you choose to configure a brand new project that you are building on your development environment.

However, a fully functioning application may require many database configurations to be made, such as: multiple range indexes, term list index settings, fields and word query setup. It would be time consuming and error prone to do this manually each time you migrated the project to a new environment.

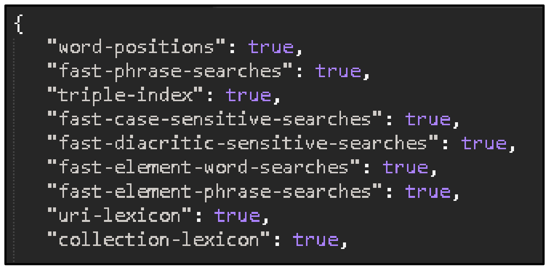
MarkLogic provides ways to help automate this process:

One option is to model your database configuration as a JSON (or XML) document and then use the Management REST API to deploy that configuration the appropriate hosts.

Another option involves using a GUI tool called configuration manager.

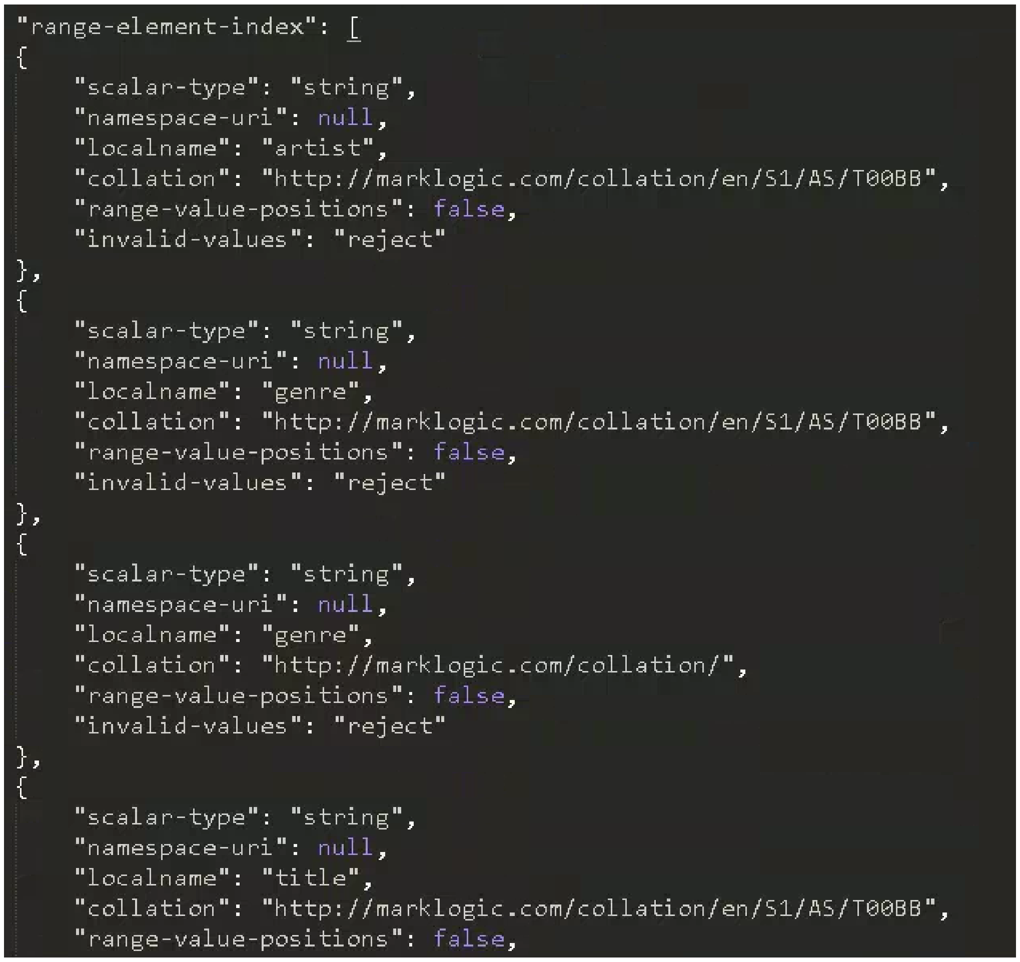
In this exercise we will look at using the Management REST API to deploy all the indexes that are needed for our project.

1. Navigate to **c:\mls-developer-node\Unit08** and open **database-config.json** in a text editor.
2. Take a few minutes to study the configuration details.
3. Note that several additional term list indexes are being enabled:



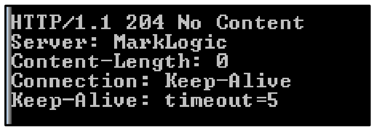
```
{
    "word-positions": true,
    "fast-phrase-searches": true,
    "triple-index": true,
    "fast-case-sensitive-searches": true,
    "fast-diacritic-sensitive-searches": true,
    "fast-element-word-searches": true,
    "fast-element-phrase-searches": true,
    "uri-lexicon": true,
    "collection-lexicon": true,
```

4. Note that several element range indexes are configured:



```
"range-element-index": [
{
    "scalar-type": "string",
    "namespace-uri": null,
    "localname": "artist",
    "collation": "http://marklogic.com/collation/en/S1/AS/T00BB",
    "range-value-positions": false,
    "invalid-values": "reject"
},
{
    "scalar-type": "string",
    "namespace-uri": null,
    "localname": "genre",
    "collation": "http://marklogic.com/collation/en/S1/AS/T00BB",
    "range-value-positions": false,
    "invalid-values": "reject"
},
{
    "scalar-type": "string",
    "namespace-uri": null,
    "localname": "genre",
    "collation": "http://marklogic.com/collation/",
    "range-value-positions": false,
    "invalid-values": "reject"
},
{
    "scalar-type": "string",
    "namespace-uri": null,
    "localname": "title",
    "collation": "http://marklogic.com/collation/en/S1/AS/T00BB",
    "range-value-positions": false,
```

5. Next, let's deploy this configuration to our database using the Management REST API.
6. Navigate to **c:\mls-developer-node\Unit08** and open **deploy-database-config.txt**.
7. Study the cURL command and then copy it.
8. Open a command line window and navigate to **c:\curl**.
9. Paste the curl command into the command line window and execute it in order to deploy the database configuration.
10. When successful, you should receive a 204 response:



```
HTTP/1.1 204 No Content
Server: MarkLogic
Content-Length: 0
Connection: Keep-Alive
Keep-Alive: timeout=5
```

11. Review your new database configuration using the Admin interface (http://localhost:8001).

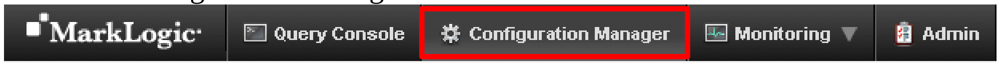
Exercise 4: Capture a Database Configuration

Now that you have your database configured, you may wish to capture a snapshot of your environment for documentation purposes or to share with other developers on your team. In order to do this, we will use a MarkLogic tool called Configuration Manager.

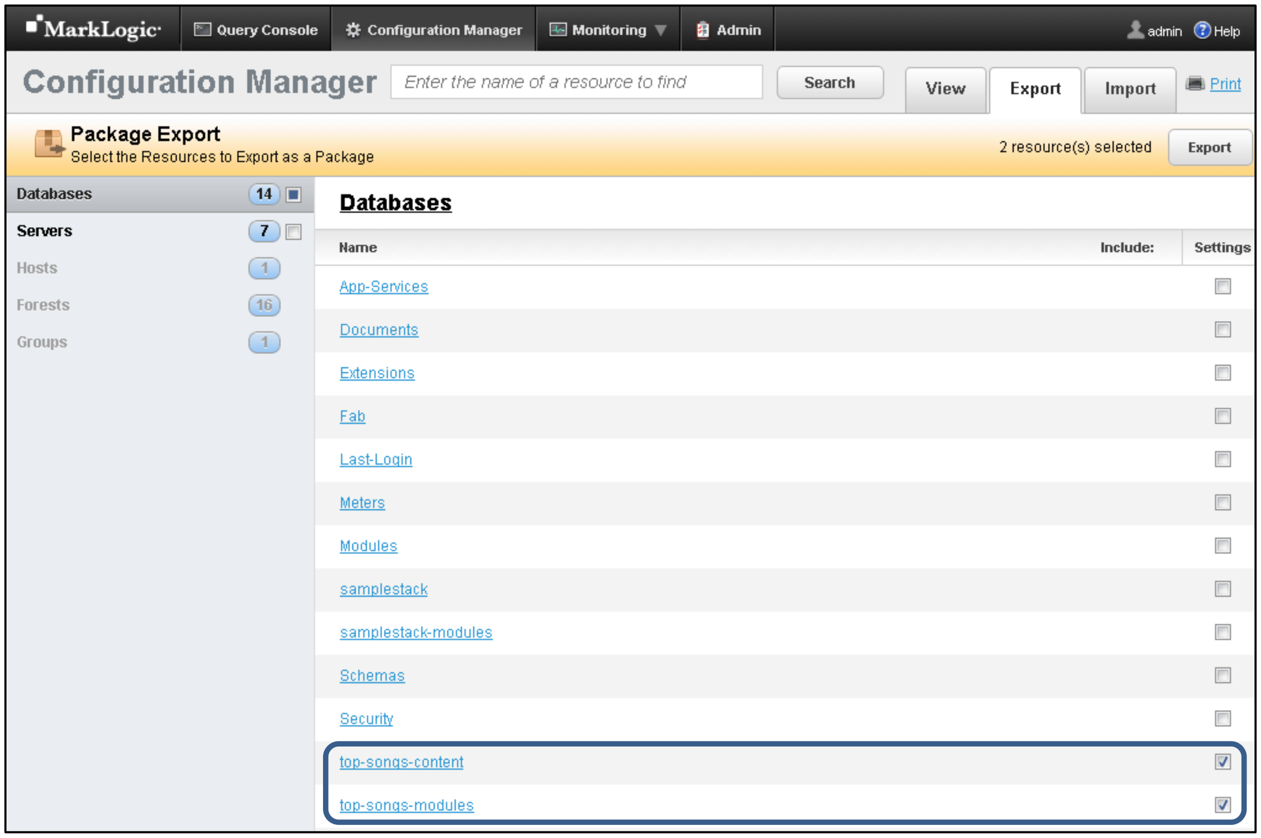
Configuration Manager performs two important functions. The first function is to provide a read only GUI interface for viewing your MarkLogic resource configurations. Because information is read only, it makes sense to provide access to this tool for those people on your team who need to view configuration data from time to time, but are not MarkLogic Administrators and therefore should not be able to use the Admin interface and make any configuration changes.

A second use for Configuration Manager is to take an export of your MarkLogic resource configuration data at a given point in time. This could be used for documentation purposes or to deploy those resources on another host.

In this exercise we will use Configuration Manager to view our app server and database configuration in MarkLogic and take an export of that data for future reference.

1. In a browser navigate to **http://localhost:8000**
2. Click the Configuration Manager icon:


3. From the databases list, select your **top-songs-content** database to view its configuration.
4. Click the **Export** tab.
5. On the databases page, select the **settings** checkbox for your **top-songs-content** and **top-songs-modules** databases as shown:



Configuration Manager

Enter the name of a resource to find

Package Export

Select the Resources to Export as a Package

2 resource(s) selected

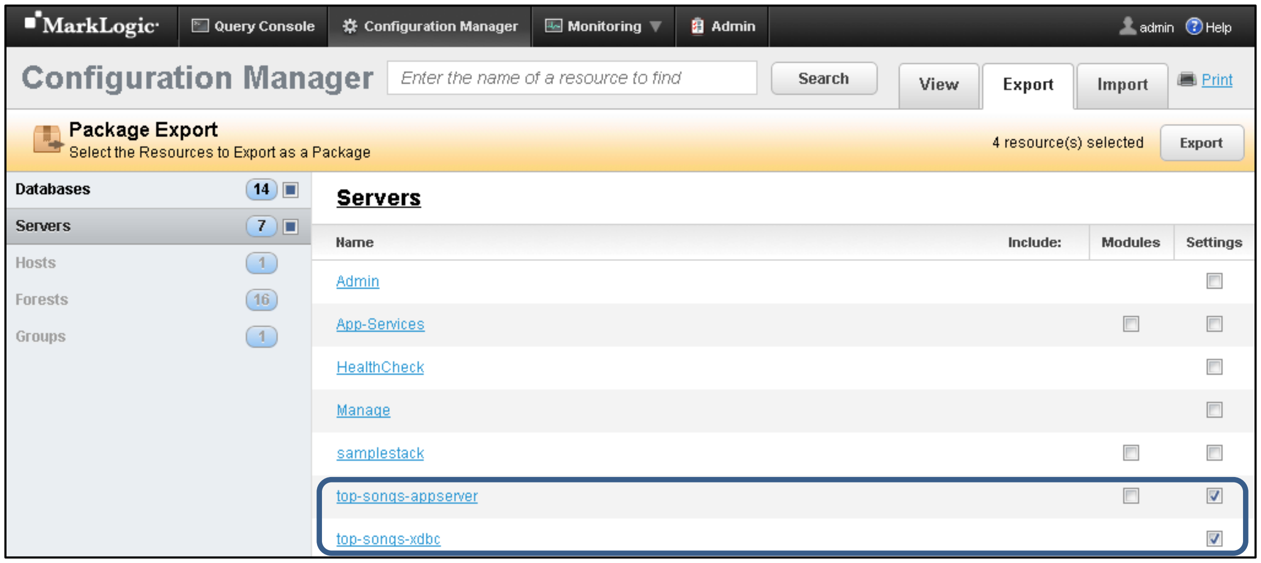
Export

Databases	14
Servers	7
Hosts	1
Forests	16
Groups	1

Databases

Name	Include: Settings
App-Services	☐
Documents	☐
Extensions	☐
Fab	☐
Last-Login	☐
Meters	☐
Modules	☐
samplestack	☐
samplestack-modules	☐
Schemas	☐
Security	☐
top-songs-content	✓
top-songs-modules	✓

6. Select **Servers** from the left hand column navigation pane.
7. Select the settings checkbox for the **top-songs-appserver** and **top-songs-xdbc** servers as shown:



Configuration Manager

Enter the name of a resource to find

Package Export

Select the Resources to Export as a Package

4 resource(s) selected

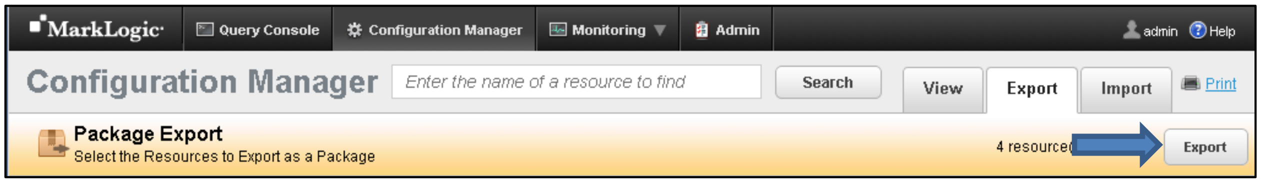
Export

Databases	14
Servers	7
Hosts	1
Forests	16
Groups	1

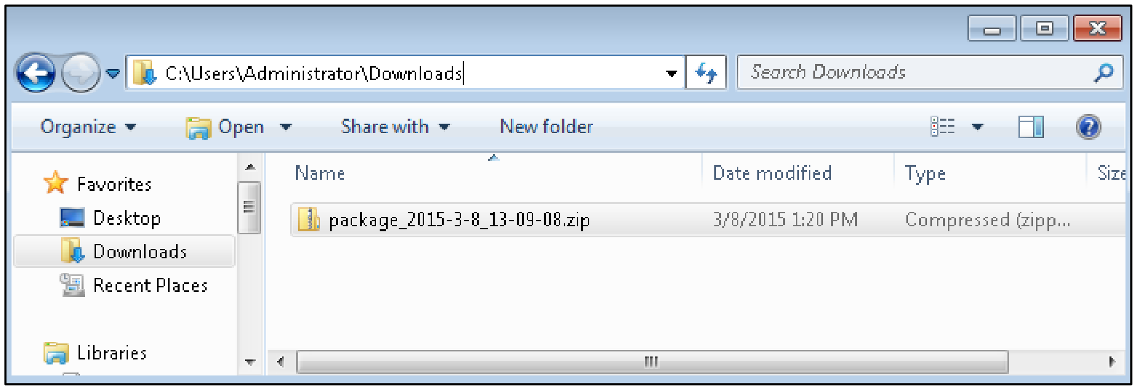
Servers

Name	Include: Modules	Settings
Admin		☐
App-Services	☐	☐
HealthCheck		☐
Manage		☐
samplestack	☐	☐
top-songs-appserver	☐	✓
top-songs-xdbc		✓

8. Click the **Export** button:



9. A zip file containing your configuration data will be created and placed in the Downloads folder on your machine:



Name	Date modified	Type	Size
package_2015-3-8_13-09-08.zip	3/8/2015 1:20 PM	Compressed (zipp...	

DIY: Setup Star Wars Indexes

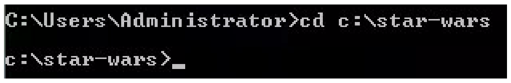
In this exercise you will configure some of the indexes that the final Star Wars application will need in order to function correctly. Once complete, you will test the final application.

Use the Admin interface to setup indexes on the **star-wars-content** database as follows:

1. Create a range index of type **double** on the **height** property.
2. Create a range index of type **string** on the **name** property.

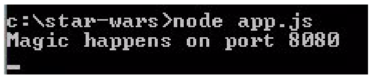
Hint:
Element range indexes apply for both XML elements and JSON properties.

3. Next, let's test the application.
4. From the command line navigate to **c:\star-wars**:



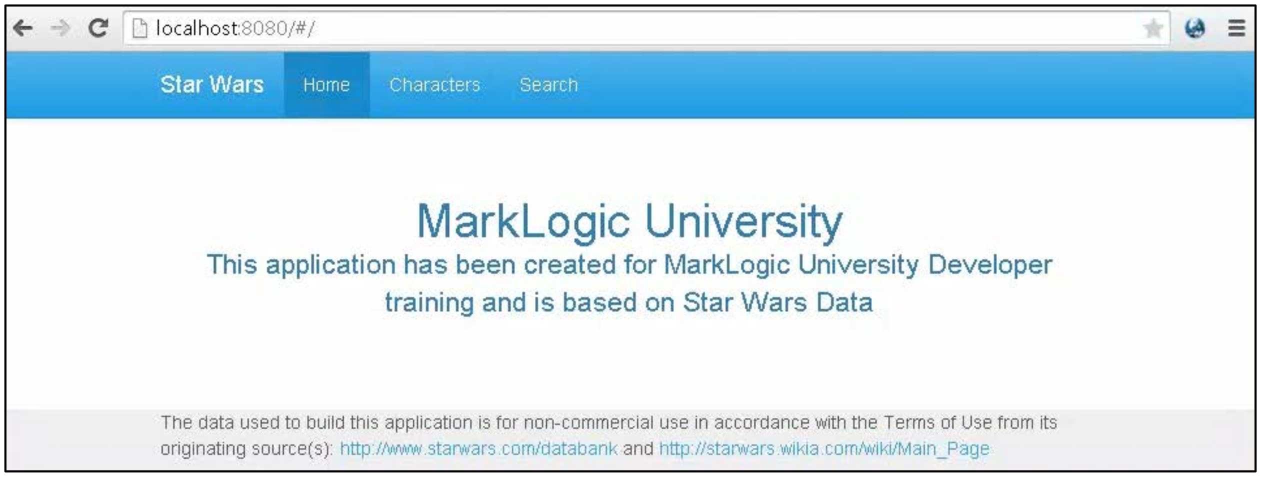
```
C:\Users\Administrator>cd c:\star-wars

c:\star-wars>_
```

5. Enter the command **node app.js** and press enter. You should see the following response:



```
c:\star-wars>node app.js
Magic happens on port 8080
_
```

6. Open a new browser tab and navigate to **http://localhost:8080**.
7. You should see the following page:



MarkLogic University

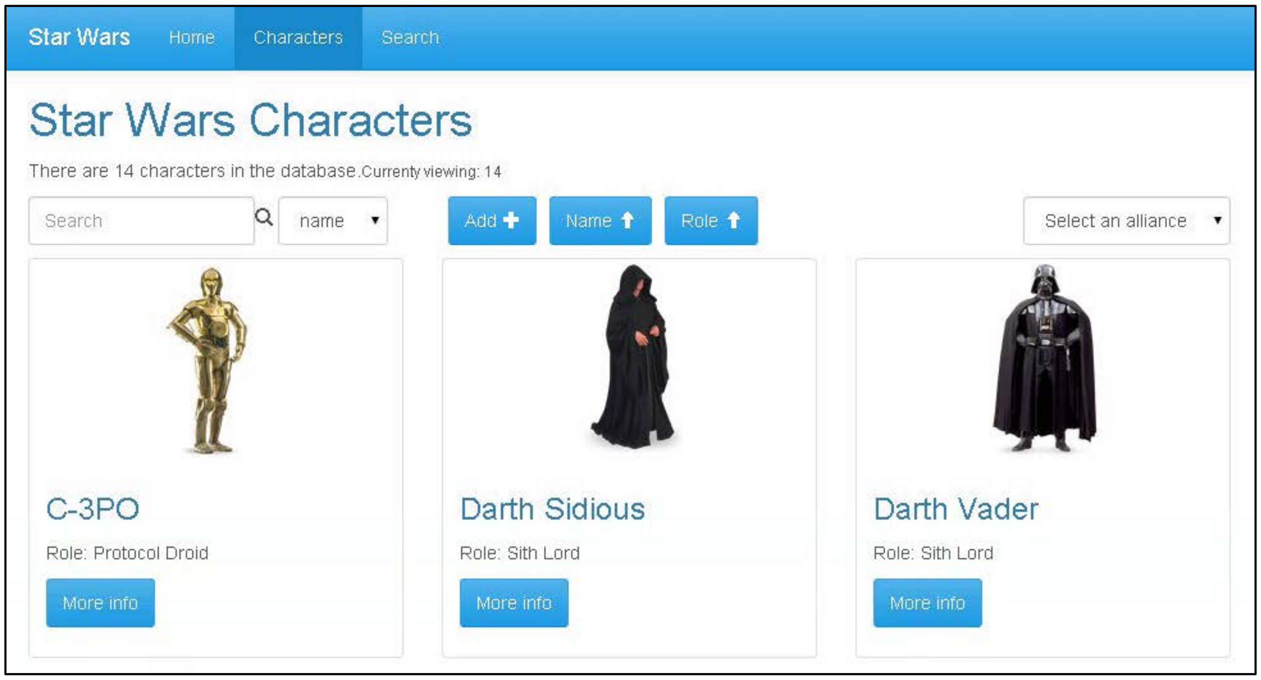
This application has been created for MarkLogic University Developer training and is based on Star Wars Data

The data used to build this application is for non-commercial use in accordance with the Terms of Use from its originating source(s): http://www.starwars.com/databank and http://starwars.wikia.com/wiki/Main_Page

8. Click the Characters tab and view some of the data that you loaded:

Note:
If you didn't meet all the defined requirements in the earlier DIY labs, your application may not function.

Consult your instructor for assistance if needed.



Star Wars Characters

There are 14 characters in the database.Currenty viewing: 14

C-3PO

Role: Protocol Droid

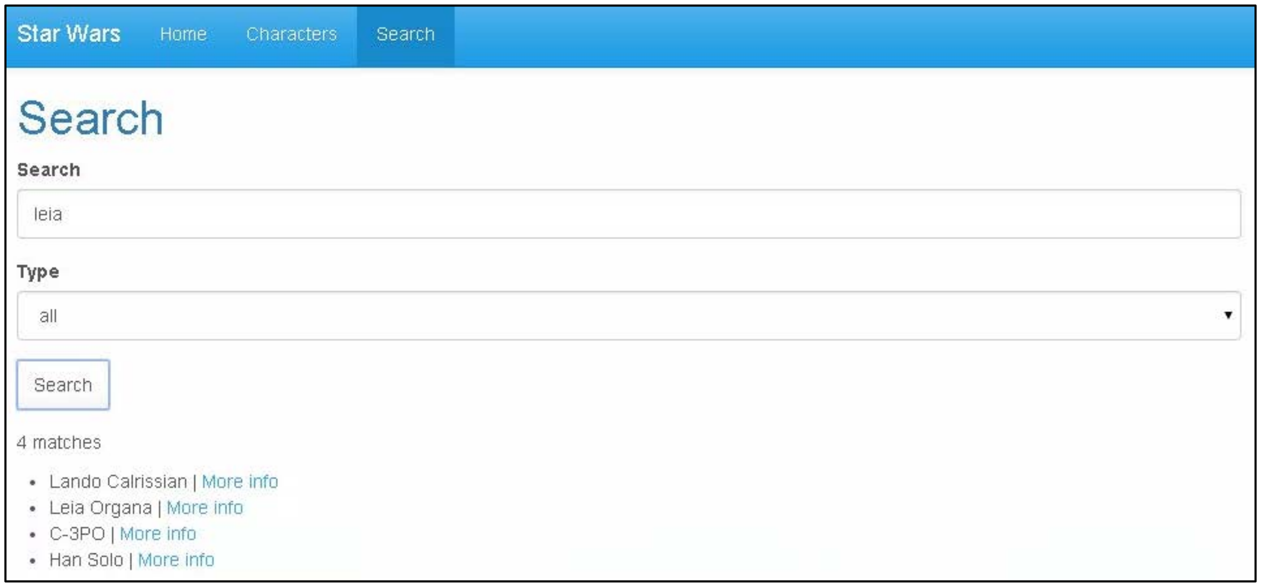
Darth Sidious

Role: Sith Lord

Darth Vader

Role: Sith Lord

9. Click on the search tab and experiment with different queries and results:



Search

Search

leia

Type

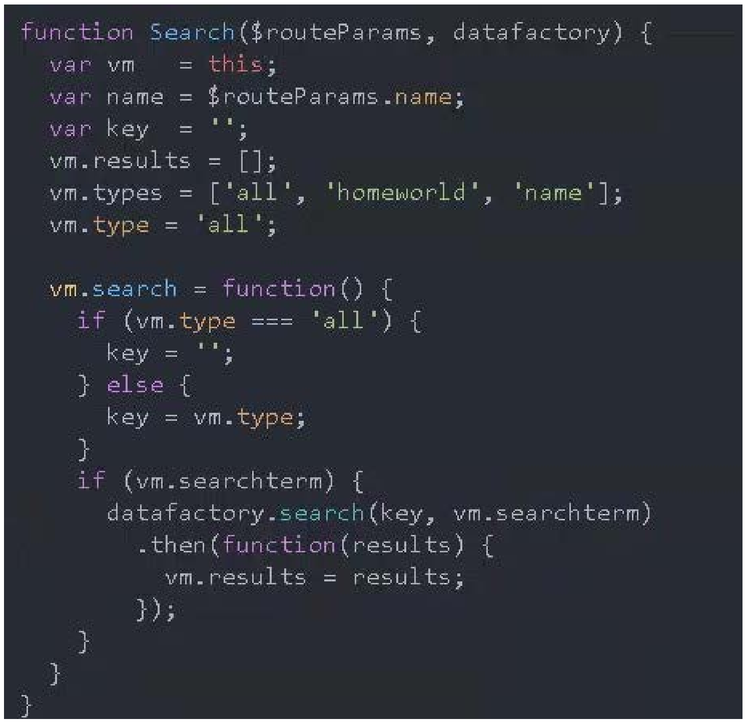
all

Search

4 matches

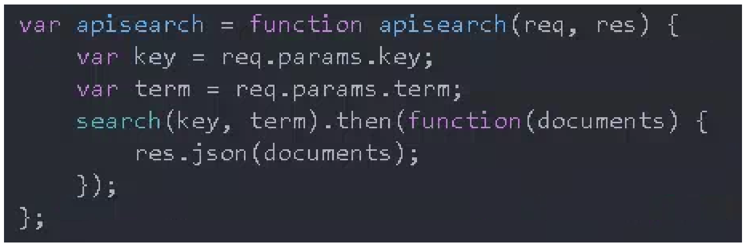
- Lando Calrissian | More info
- Leia Organa | More info
- C-3PO | More info
- Han Solo | More info

10. Explore the code in your editor at **c:\star-wars**
11. Take a look at **/public/js/characters/character-search/starwars.search.controller.js**:
12. Notice how the keys are being set based on the user selection in the “type” drop down box.



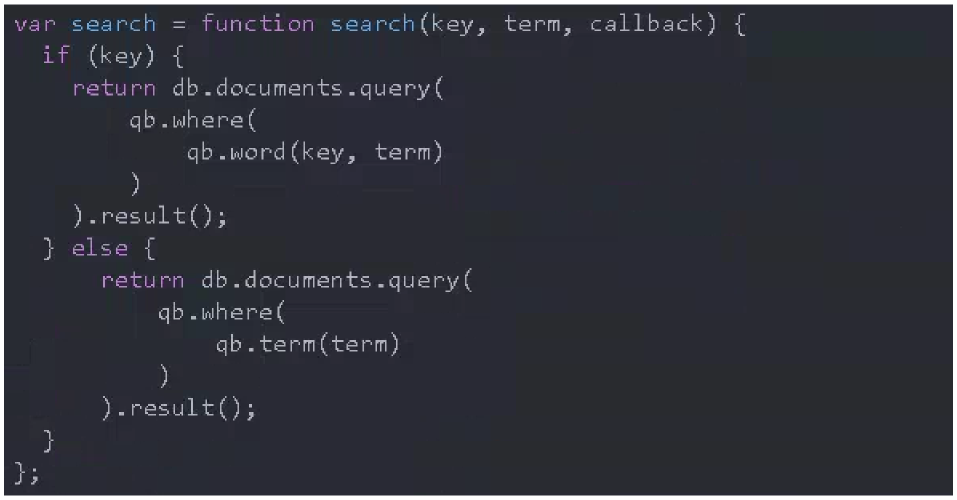
```
function Search($routeParams, datafactory) {
  var vm   = this;
  var name = $routeParams.name;
  var key  = '';
  vm.results = [];
  vm.types = ['all', 'homeworld', 'name'];
  vm.type = 'all';

  vm.search = function() {
    if (vm.type === 'all') {
      key = '';
    } else {
      key = vm.type;
    }
    if (vm.searchterm) {
      datafactory.search(key, vm.searchterm)
        .then(function(results) {
          vm.results = results;
        });
    }
  }
}
```

13. Take a look at c:\star-wars\routes.js
14. This is where the MarkLogic Node.js client API code is built.
15. Look at the function controlling the search.
16. Note that it gathers the parameters from the user input and passes the key and the term to a function called **search**:



```
var apisearch = function apisearch(req, res) {
    var key = req.params.key;
    var term = req.params.term;
    search(key, term).then(function(documents) {
        res.json(documents);
    });
};
```

17. Finally, let's take a look at the **search** function (also in routes.js):



```
var search = function search(key, term, callback) {
  if (key) {
    return db.documents.query(
        qb.where(
            qb.word(key, term)
        )
    ).result();
  } else {
      return db.documents.query(
          qb.where(
              qb.term(term)
          )
      ).result();
  }
};
```

18. Note that it is performing a word query against a specific property or against the document as a whole, depending on what was selected by the user on the front end and then passed into the function.