MarkLogic UNIVERSITY

# Unit 14

## Introduction to Semantics and RDF

Configure the Triple Index

Load Triple Data

Explore Triple Data

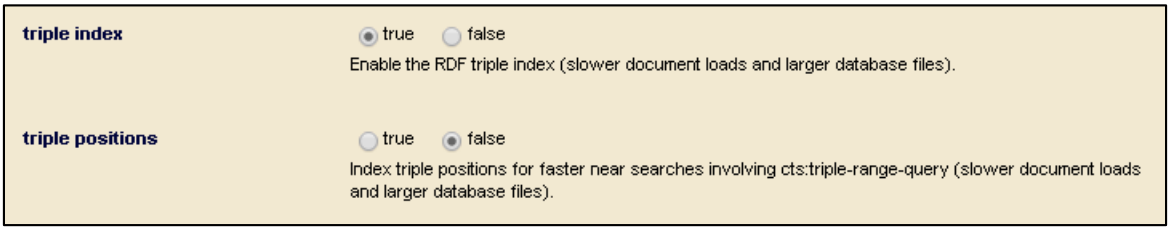
Query Triple Data with SPARQL

Query Triple Data from a Node.js App

## Exercise 1: Configure the Triple Index

In this exercise you will configure the triple index for your database.

1. In a browser tab, open the Admin interface at **http://localhost:8001**
2. Click Configure→Databases→top-songs-content
3. Scroll down through the database configuration settings and note the options for the triple indexes. The triple index was already turned on as a result of our automated index deployment in a prior lab. The triple positions index is needed for near queries on triple data, which we do not require for this exercise:



## Exercise 2: Load Triple Data

In this data we will load openly available RDF triple data using MarkLogic Content Pump. This data was made available from DBpedia (http://dbpedia.org/) and contains many facts about people and places modeled as RDF triples.

1. Navigate to **C:\mls-developer-node\Unit14\config** and open **14-2 (mlcp-options-win).txt**
2. Study the mlcp options:

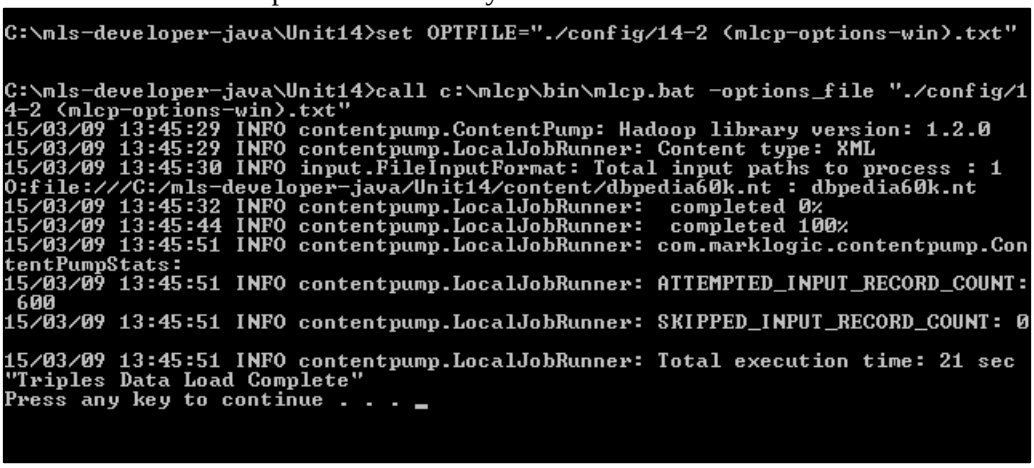
```
import
-mode
local
-host
localhost
-port
7011
-username
admin
-password
admin
-input_file_path
C:\\mls-developer-node\\Unit14\\content\\dbpedia60k.nt
-input_file_type
RDF
-output_uri_prefix
/triples/
```

3. Navigate to **C:\mls-developer-node\Unit14** and *edit* **14-2 (mlcp-load-triples).bat**:

```
set OPTFILE="./config/14-2 (mlcp-options-win).txt"
call c:\mlcp\bin\mlcp.bat -options_file %OPTFILE%
echo "Triples Data Load Complete"
pause
```

4. Note that it defines where to go and read the load options.
5. Note that it calls the mlcp.bat file from the location mlcp has been placed. This is the directory it has been placed on your VM; it is not required to be at this location.
6. Double click **14-2 (mlcp-load-triples).bat** to execute the batch file.

7. Validate the load completes successfully:




8. Exit out of the command prompt.

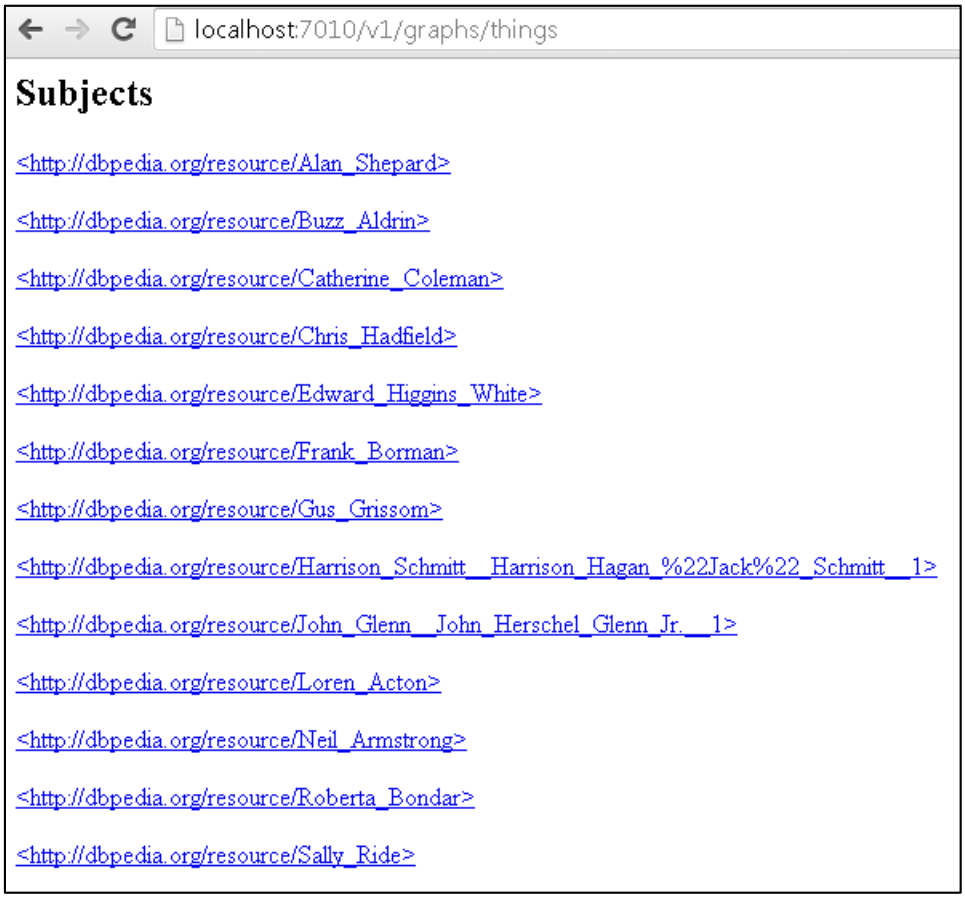
## Exercise 3: Explore Triple Data

In this exercise you will use a REST API endpoint specifically designed to provide you some general visibility into your triple data so that you can explore the data, its model, and relationships.

1. Open a new browser tab and navigate to **http://localhost:7010/v1/graphs/things**

> *Note:*
> *Remember that port 7010 is where we created our REST API instance earlier. While we could access this endpoint from the command line using a cURL command as well, using the browser will provide the data to us in a more user friendly presentation.*

2. You are presented with a list of subjects from your triple data. The endpoint provides a response containing the first set of subjects. This is not all the triple data that you have in the database.




3. Take a minute to explore the data, for example clicking on Neil Armstrong:

4 triples

<http://dbpedia.org/resource/Neil_Armstrong> <http://dbpedia.org/ontology/Astronaut/timeInSpace> "0.5166666666666667"^^<http://dbpedia.org/datatype/minute> .
<http://dbpedia.org/resource/Neil_Armstrong> <http://dbpedia.org/ontology/Astronaut/timeInSpace> "12372.0"^^<http://dbpedia.org/datatype/minute> .
<http://dbpedia.org/resource/Neil_Armstrong> <http://www.w3.org/2000/01/rdf-schema#comment> "Neil Alden Armstrong (born August 5, 1930) is an American former astronaut, test pilot, aerospace engineer, university professor, United States Naval Aviator, and the first person to set foot upon the Moon. Before becoming an astronaut, Armstrong was in the United States Navy and served in the Korean War."^^<> .
<http://dbpedia.org/resource/Apollo_11> <http://dbpedia.org/ontology/crewMember> <http://dbpedia.org/resource/Neil_Armstrong> .

4. You are presented with all the triples that have Neil Armstrong as a **subject** or **object**.

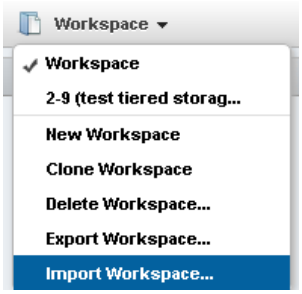
| SUBJECT (IRI) | PREDICATE (IRI) | OBJECT (typed value) |
|---|---|---|
| Neil Armstrong | timeInSpace | 0.516666 (minute) |
| Neil Armstrong | timeInSpace | 12372.0 (minute) |
| Neil Armstrong | Comment | "Neil Alden Armstrong…" (no data type explicitly defined = string) |
| Apollo 11 | crewmember | Neil Armstrong |

5. Take a few minutes to explore your data.

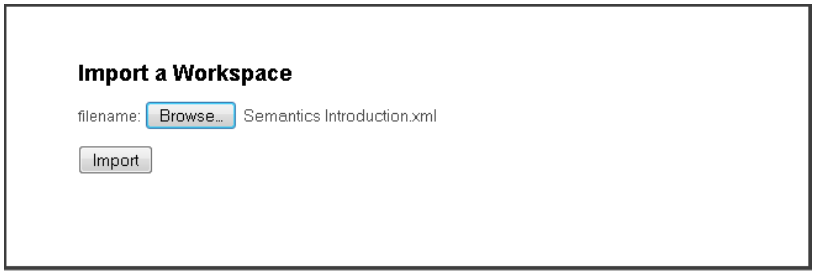
## Exercise 4: Query Triple Data with SPARQL

In this exercise you will learn to write SPARQL queries against the triple data loaded in MarkLogic.

1. Open Query console (http://localhost:8000/qconsole)
2. From the upper right pane in Query Console, import a workspace:

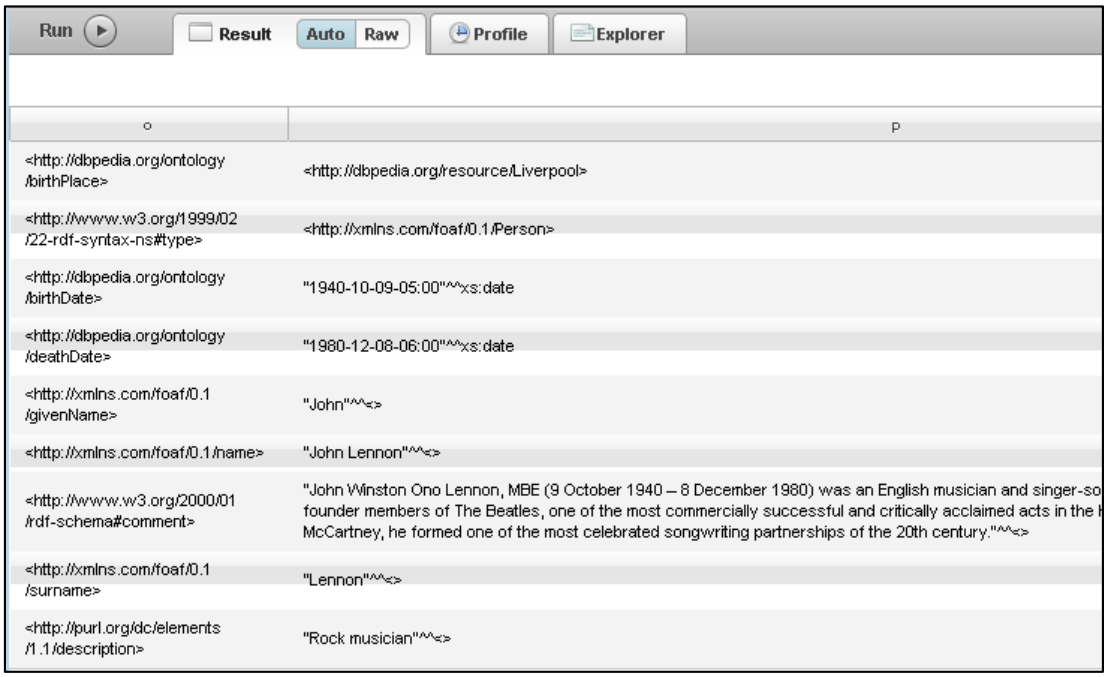



3. Find the workspace at: **c:\mls-developer-node\Unit14\Semantics Introduction.xml**




4. Click the **Import** button.
5. Click the tab labeled **SPARQL 1**.
6. Take a minute to study the comments describing the query and the SPARQL query itself.
7. Select your **top-songs-content** database in the content source and click Run.
8. See the result that shows you predicate values where the subject is equal to Apollo 11 and the object is equal to crewMember:

9. Click the tab labeled **SPARQL 2**.
10. Study the comments outlining your objective. Take a few minutes to experiment writing a query to return the desired results.
11. Try to write this on your own, but if you get stuck, you may view the solution in the tab labeled **SPARQL 3**.
12. Look at the results to familiarize yourself with what type of facts our data tells us about a particular subject, as well as how the IRIs are structured. This type of information might be valuable to users of the Top Songs application who are looking for more information regarding a particular artist. As we go forward, you will build this capability into the Top Songs application:




13. Click the tab labeled **SPARQL 4**.
14. Study the comments outlining your objective. Take a few minutes to experiment writing a query to return the desired results.
15. Try to write this on your own, but if you get stuck, you may view the solution in the tab labeled **SPARQL 5**.
16. Notice the WHERE constraint in the SPARQL query:

```
where { db:John_Lennon onto:birthPlace ?birthTown .
        ?person onto:birthPlace ?birthTown
      }
```

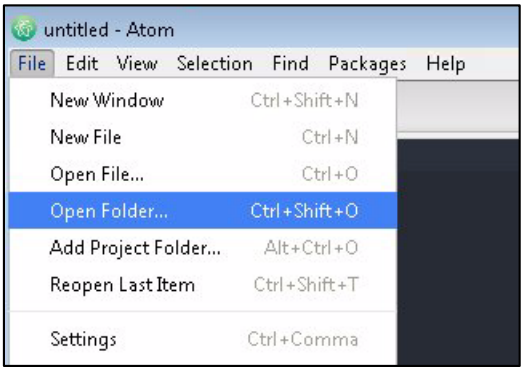
17. This where clause can be interpreted as saying: "Figure out the birth place where the subject John Lennon was born. Then based on that subjects (John Lennon) birthplace (Liverpool), figure out other subjects who have the same birthplace as John Lennon."
18. Click the tab labeled **SPARQL 6**.
19. Take a minute to study the comments describing the query and the SPARQL query itself.
20. Click Run, and see the result of the DISTINCT and LIMIT constructs. Study the results to become more familiar with your triple data.
21. Click the tab labeled **SPARQL 7**.
22. Take a minute to study the comments describing the query and the SPARQL query itself.
23. Click Run, and see the result of the ASK construct.
24. Click the tab labeled **SPARQL 8**.
25. Take a minute to study the comments describing the query and the SPARQL query itself.
26. Click Run, and see the result of the OPTIONAL, FILTER, and ORDER BY constructs.
27. Click the tab labeled **SPARQL 9**.
28. Take a minute to study the comments describing the query and the SPARQL query itself.
29. Click Run, and see the result of the CONSTRUCT and FILTER constructs.

## Exercise 5: Query Triple Data from a Node.js App

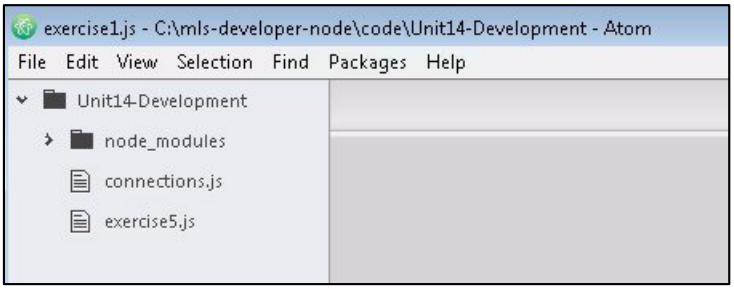
In this exercise you will run a SPARQL query from a Node.js app.

1. From the command line, navigate to the **c:\mls-developer-node\code\Unit14-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 **Unit14-Development** folder from c**:\mls-developer-node\code**
5. You should see the following structure in place in your editor:




6. Select **connections.js**.
7. Note the definition of a MarkLogic Administrator user. You don't need to be an admin to run SPARQL queries, but you must have a role with the **sem:sparql** execute privilege:

```
mlAdmin: {
  host: "localhost",
  port: 7010,
  user: "admin",
  password: "admin"
}
```

8. Select **exercise5.js**.
9. Note the definition of the SPARQL query and the execution using the **graphs.sparql** method:

```
var marklogic = require("marklogic");
var dbConn = require("./connections.js");
var mlAdmin = marklogic.createDatabaseClient(dbConn.mlAdmin);

var query = [
  "PREFIX db: <http://dbpedia.org/resource/>",
  "SELECT * ",
  "WHERE { db:John_Lennon ?p ?o }"
];

mlAdmin.graphs.sparql("application/sparql-results+json", query.join("\n")
).result(function (response) {
  console.log(JSON.stringify(response, null, 2));
}, function(error) {
  console.log(JSON.stringify(error, null, 2));
});
```

10. Now let's test the code.
11. At the command line, go to the **c:\mls-developer-node\Unit14-Development** directory.
12. Enter **node exercise5.js** and press enter.
13. You should see triples about John Lennon in the response:

```
c:\mls-developer-node\code\Unit14-Development>node exercise5.js
{
  "head": {
    "vars": [
      "p",
      "o"
    ]
  },
  "results": {
    "bindings": [
      {
        "p": {
          "type": "uri",
          "value": "http://dbpedia.org/ontology/birthPlace"
        },
        "o": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/Liverpool"
        }
      },
```