GraphDB SPARQL Query Example

Query an Ontotext GraphDB repository with SPARQL from a FrameworX solution and show the result in a DataGrid that follows a tag.

How-to Guides **Solution Examples** **Feature Examples** **Data Foundation Examples** GraphDB SPARQL Query

---

Download the solution: StarWars.dbsln

Companion to the Reports WebData - SPARQL and Graph Databases reference page.

Prerequisites:

- FrameworX 10.1.6a or later.
- Ontotext GraphDB running on `localhost:7200`. With Docker: `docker run -d --name graphdb -p 7200:7200 ontotext/graphdb:10.8.5`
- A GraphDB repository named `starwars` with the Star Wars sample data. See Create the GraphDB repository below.

Summary

The solution sends a SPARQL query to a GraphDB repository, receives the answer as a DataTable and shows it in a DataGrid. The query asks for every relationship of one character and the name of the object at the other end, for example `starship X-wing`. A Text tag holds the character name, so changing the tag changes the table on screen.

The same pattern answers plant questions: replace the character with an equipment name and the relationships with `feedsInto`, and the query returns what is downstream of that equipment.

Technical Information

Create the GraphDB repository

The data comes from the Star Wars guide built into the GraphDB Workbench. The guide creates the repository and imports `starwars.ttl`, 3,966 statements about the films, characters, planets, starships and vehicles.

1. Open the GraphDB Workbench at `http://localhost:7200`.
2. Go to **Help Interactive guides** and start **The Star Wars guide**.
3. Follow the guide through the repository creation and the import of `starwars.ttl`. Keep the repository ID `starwars`. With another ID, change the end of the WebData row's DefaultURL to match.
4. The rest of the guide (visual graph, SPARQL queries) is optional for this example.

To check the repository, open **SPARQL** in the Workbench, select `starwars` and run the query below with `Luke Skywalker` in place of `{Tag.Character}`. It returns 11 rows.

WebData row

In **Reports WebData**, the row **StarWars**:

Column	Value
API	`REST`
Encoding	`Text`
DefaultURL	`http://localhost:7200/repositories/starwars`
Authorization	`NoAuth`

The row's Editor holds the query. `{Tag.Character}` is filled in with the tag value each time the query runs.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX voc: <https://swapi.co/vocabulary/>
SELECT ?relation ?item
WHERE {
  ?person rdfs:label "{Tag.Character}" ;
          ?rel ?obj .
  ?obj rdfs:label ?item .
  FILTER(STRSTARTS(STR(?rel), STR(voc:)))
  BIND(STRAFTER(STR(?rel), STR(voc:)) AS ?relation)
}
ORDER BY ?relation ?item
```

Tags

Tag	Type	Use
`Character`	Text	The name the query looks up. Start value `Luke Skywalker`.
`Result`	DataTable	The query result, shown by the DataGrid.
`Rows`	Integer	Number of rows returned, or `-1` when the WebData row is disabled.
`Err`	Text	The error message when the call fails, empty otherwise.

Script Task

The Server task **QueryGraph** runs every 10 seconds:

```
try
{
    @Tag.Err = "";
    DataTable t = await @Report.WebData.StarWars.SparqlQueryTableAsync();
    @Tag.Result = t;
    @Tag.Rows = t == null ? -1 : t.Rows.Count;
}
catch (Exception ex)
{
    @Tag.Err = ex.GetType().Name + ": " + ex.Message;
    @Info.Trace("SPARQL query failed: " + ex.Message);
}
```

Display

The main Display has a DataGrid with **Data Source** set to `Tag.Result`. It shows two columns, `relation` and `item`.

Expected result

Character	Rows
Luke Skywalker	11: five films, a friend, a homeworld, two starships and two vehicles
Leia Organa	7
Han Solo	7
Darth Vader	6
Yoda	6

Change `Character` while the solution runs, and the DataGrid shows the new character's relationships within 10 seconds.

Caveats

- The name must match the label in the graph exactly, including capital letters.
- `{Tag.Character}` is filled in only because the query is in the row's Editor. Query text passed as an argument to `SparqlQueryTableAsync(query)` is sent as written.

- The 10-second period suits a demonstration. In a solution, run the query on a tag change or from a button.

In this section...
