FREE eBook

LEARNING sparql

Free unaffiliated eBook created from
Stack Overflow contributors.

#sparql

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sparql

It is an unofficial and free sparql ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sparql.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with sparql

Remarks

This section provides an overview of what sparql is, and why a developer might want to use it.

It should also mention any large subjects within sparql, and link out to the related topics. Since the Documentation for sparql is new, you may need to create initial versions of those related topics.

Versions

Version	Release Date
1.0	2008-01-15
1.1	2013-03-21

Examples

Getting Started with a SPARQL Endpoint

A query engine is required in order to execute SPARQL queries over a dataset. Query engines may be embedded in applications, or accessed remotely. Remote access may be through a vendor-specific API, or through the SPARQL protocol if an implementation supports it. This documentation will not cover how to submit queries through specific vendor APIs.

SPARQL Endpoint implementations typically provide a user-friendly web interface for submitting queries and viewing their results. Public SPARQL endpoints (such as DBPedia) can serve as useful datasets for non-mutating examples.

If you wish to configure a private SPARQL Endpoint for experimentation, Apache Fuseki provides a free and platform independent option.

Read Getting started with sparql online: https://riptutorial.com/sparql/topic/1303/getting-started-with-sparql

Chapter 2: Working with graphs

Introduction

What is a Named Graph? An internal database document identifier (name) used by a SQL-compliant RDBMS to partition storage of relations represented as RDF sentence/statement graphs. Why are Named Graphs Important? A Named Graph is like a page in a book (the database) that contains a collection of paragraphs (sentence collections). Thus, it provides a powerful mechanism for query scoping that negates the need to scope all database queries to the entire database.

Examples

Storing Data in a Named Graph

```
PREFIX    rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX    owl: <http://www.w3.org/2002/07/owl#>
PREFIX schema: <http://schema.org/>
PREFIX   foaf: <http://xmlns.com/foaf/0.1/>

WITH <urn:this:database:doc>
INSERT { <#relatedTo> a                     rdf:Property, owl:ObjectProperty ;
                      rdfs:label            "relatedTo"                      ;
                      rdfs:domain           rdfs:Resource                    ;
                      rdfs:range            xsd:anyURI                       .
         foaf:knows rdfs:subPropertyOf      <#relatedTo>                     ;
                    owl:equivalentProperty  schema:knows                     .
         <#this>    a                       foaf:Person                      ;
                    schema:name             "John Doe"                .
         <#that>    a                       foaf:Person                      ;
                    schema:name             "Jane Doe"                  .
       }
```

Read Working with graphs online: https://riptutorial.com/sparql/topic/8092/working-with-graphs

Chapter 3: Working with language tags

Examples

Getting a language tag from a literal

Literals in RDF may be language tagged strings. These literals have a text part as well as a language tag. For instance, the literal **"color"@en** has the text part **"color"** and the language tag **"en"**. In a SPARQL query, use the **lang** function to get the language of a literal with a language tag. If a literal does not have a language tag, then **lang** returns the empty string, **""**.

Comparing language tags

The SPARQL function **langMatches** can be used to compare the language tags of RDF literals in SPARQL queries. The simple equality operator, **=**, can be used to compare exact string matches, but will not correctly consider regional variants. For instance, the four possible values of **?str** in:

```
values ?str { "color"@en-US "color"@en "colour"@en "colour"@en-GB }
```

are all English language strings, but two of these have regional specifications. This means that

```
select ?str {
  values ?str { "color"@en-US "color"@en "colour"@en "colour"@en-GB }
  filter (lang(?str) = 'en')
}
```

will return only two results, since only two of the values of **?str** have **"en"** as a language tag. However,

```
select ?str {
  values ?str { "color"@en-US "color"@en "colour"@en "colour"@en-GB }
  filter langMatches(lang(?str), 'en')
}
```

will return all four values.

Read Working with language tags online: https://riptutorial.com/sparql/topic/5210/working-with-language-tags

Credits

S. No	Chapters	Contributors
1	Getting started with sparql	Community, Joshua Taylor, Rob Hall, unor
2	Working with graphs	i-blis, Kingsley Uyi Idehen
3	Working with language tags	Joshua Taylor