Package: SPARQL (via r-universe)

January 9, 2025

Type Package

Title SPARQL client

Version 1.16

Date 2013-10-23

Author Willem Robert van Hage <willem.van.hage@synerscope.com>, with contributions from: Tomi Kauppinen, Benedikt Graeler, Christopher Davis, Jesper Hoeksema, Alan Ruttenberg, and Daniel Bahls.

Maintainer Willem Robert van Hage <willem.van.hage@synerscope.com>

Description Use SPARQL to pose SELECT or UPDATE queries to an end-point.

License GPL-3

Depends XML, RCurl

LazyLoad yes

NeedsCompilation no

Date/Publication 2013-10-25 17:39:14

Config/pak/sysreqs make libxml2-dev

Repository https://pik-piam.r-universe.dev

RemoteUrl https://github.com/cran/SPARQL

RemoteRef HEAD

RemoteSha 51b993f16cb924f2ccc325f08f42da3420ea08bc

Contents

SPARQL-package *SPARQL client*

Description

Load SPARQL SELECT query result tables as a data frame, or UPDATE the triple store by connecting to an end-point over HTTP.

The development of this library has been developed in part within the COMBINE project supported by the ONR Global NICOP grant N62909-11-1-7060.

Details

Package:	SPARQL
Type:	Package
Version:	1.15
Date:	2013-10-23
License:	GPL-3
Depends:	XML
LazyLoad:	yes

Author(s)

Willem Robert van Hage <willem.van.hage@synerscope.com>, with contributions from: Tomi Kauppinen, Benedikt Graeler, Christopher Davis, Jesper Hoeksema, Alan Ruttenberg, and Daniel Bahls. Maintainer: Willem Robert van Hage <willem.van.hage@synerscope.com>

References

SPARQL specification, http://www.w3.org/TR/rdf-sparql-query/.
Examples of SPARQL end-points, http://www.w3.org/wiki/SparqlEndpoints.

Examples

```
## Not run:
d <- SPARQL(url="http://services.data.gov.uk/reference/sparql",
            query="SELECT * WHERE { ?s ?p ?o . } LIMIT 10",
            ns=c('time','<http://www.w3.org/2006/time#>'))

is.data.frame(d$results)

# draw a pie chart from data from the Linked Open Piracy data set
endpoint <- "http://semanticweb.cs.vu.nl/lop/sparql/"
q <-
  "SELECT *
```

```
   WHERE {
     ?event sem:hasPlace ?place .
     ?place eez:inPiracyRegion ?region .
   } LIMIT 20"
prefix <- c("lop","http://semanticweb.cs.vu.nl/poseidon/ns/instances/",
            "eez","http://semanticweb.cs.vu.nl/poseidon/ns/eez/")
res <- SPARQL(endpoint,q,prefix)$results
pie(sort(table(res$region)),col=rainbow(12))

## End(Not run)
```

---

commonns *commonns*

Description

A vector of common namespaces and their prefixes.

Author(s)

Willem Robert van Hage

SPARQL *SPARQL client*

Description

This function connects to a SPARQL end-point over HTTP or HTTPs, poses a SELECT query or an update query (LOAD, INSERT, DELETE). If given a SELECT query it returns the results as a data frame with a named column for each variable from the SELECT query, a list of prefixes and namespaces that were shortened to qnames is also returned. If given an update query nothing is returned. If the parameter "query" is given, it is assumed the given query is a SELECT query and a GET request will be done to get the results from the URL of the end point. Otherwise, if the parameter "update" is given, it is assumed the given query is an update query and a POST request will be done to send the request to the URL of the end point.

Usage

```
SPARQL(url = "http://localhost/", query = "", update="", ns = NULL, param = "",
       extra = NULL, format="xml", curl_args=NULL, parser_args=NULL)
```

Arguments

url	The URL of the SPARQL end-point.
query	A SPARQL SELECT query to fire at the end-point.
update	A SPARQL update query (LOAD, INSERT, DELETE)) to fire at the end-point.
ns	Prefixes to shorten IRIs returned by the SPARQL end-point. For example, `ns=c('dc','<http://purl.org/dc/elements/1.1/>', 'rdfs','<http://www.w3.org/2000/01/rdf-schema#>')` will shorten the IRIs 'http://purl.org/dc/elements/1.1/title' to 'dc:title' and 'http://www.w3.org/2000/01/rdf-schema#label' to 'rdfs:label'.
param	By default a SPARQL end-point accepts queries in the "query" HTTP parameter and updates in the "update" parameter. If the end-point uses a different parameter you can specify this here.
extra	Extra parameters and their values that will be added to the HTTP request. Some SPARQL end-points require extra parameters to work. These can be supplied, in URL encoded form, as a character vector with this parameter. This field can be used to specify the various ways in which different end-points can be told to return a certain format. For example, extra=list(resultFormat="xml") or extra=list(output="xml",queryLn="SPARQL")
format	Can be used to explicitly state what kind of format is returned by the output. This version supports "xml", "csv" and "tsv".
curl_args	A list of arguments that will be passed to RCurl when fetching the SPARQL results over HTTP. This can be used, for example, to pass authentication arguments, or to change the mime type of a post request from multipart/form-data to application/x-www-form-urlencoded, by passing curl_args=list(style="post").
parser_args	A list of arguments that will be passed to the XML, CSV, TSV, etc. parser that processed the returned SPARQL result table.

Value

The returned data frame contains a column for each variable in the SELECT query. For example, the query "SELECT * WHERE { ?s ?p ?o . } LIMIT 10" will yield three columns named "s", "p", and "o". The query "SELECT ?s WHERE { ?s ?p ?o . } LIMIT 10" will yield only one column named "s".

Author(s)

Willem Robert van Hage and Tomi Kauppinen

References

SPARQL specification, http://www.w3.org/TR/rdf-sparql-query/.
SPARQL Update specification, http://www.w3.org/TR/sparql11-update/.
Examples of SPARQL end-points, http://www.w3.org/wiki/SparqlEndpoints.

Examples

```
## Not run:
d <- SPARQL(url="http://services.data.gov.uk/reference/sparql",
            query="SELECT * WHERE { ?s ?p ?o . } LIMIT 10",
            ns=c('time','<http://www.w3.org/2006/time#>'))

is.data.frame(d$results)

# draw a pie chart from data from the Linked Open Piracy data set
endpoint <- "http://semanticweb.cs.vu.nl/lop/sparql/"
q <-
  "SELECT *
   WHERE {
     ?event sem:hasPlace ?place .
     ?place eez:inPiracyRegion ?region .
   }"
prefix <- c("lop","http://semanticweb.cs.vu.nl/poseidon/ns/instances/",
            "eez","http://semanticweb.cs.vu.nl/poseidon/ns/eez/")
res <- SPARQL(endpoint,q,prefix)$results
pie(sort(table(res$region)),col=rainbow(12))

# draw a stacked bar chart from data from the Linked Open Piracy data set
q <-
  "SELECT *
   WHERE {
     ?event sem:eventType ?event_type .
     ?event sem:hasPlace ?place .
     ?place eez:inPiracyRegion ?region .
   }"
res <- SPARQL(endpoint,q,ns=prefix)$results
restable <- table(res$event_type,res$region)
par(mar=c(4,10,1,1))
barplot(restable,col=rainbow(10),horiz=TRUE,las=1,cex.names=0.8)
legend("topright",rownames(restable),
       cex=0.8,bty="n",fill=rainbow(10))

## End(Not run)
```

Index