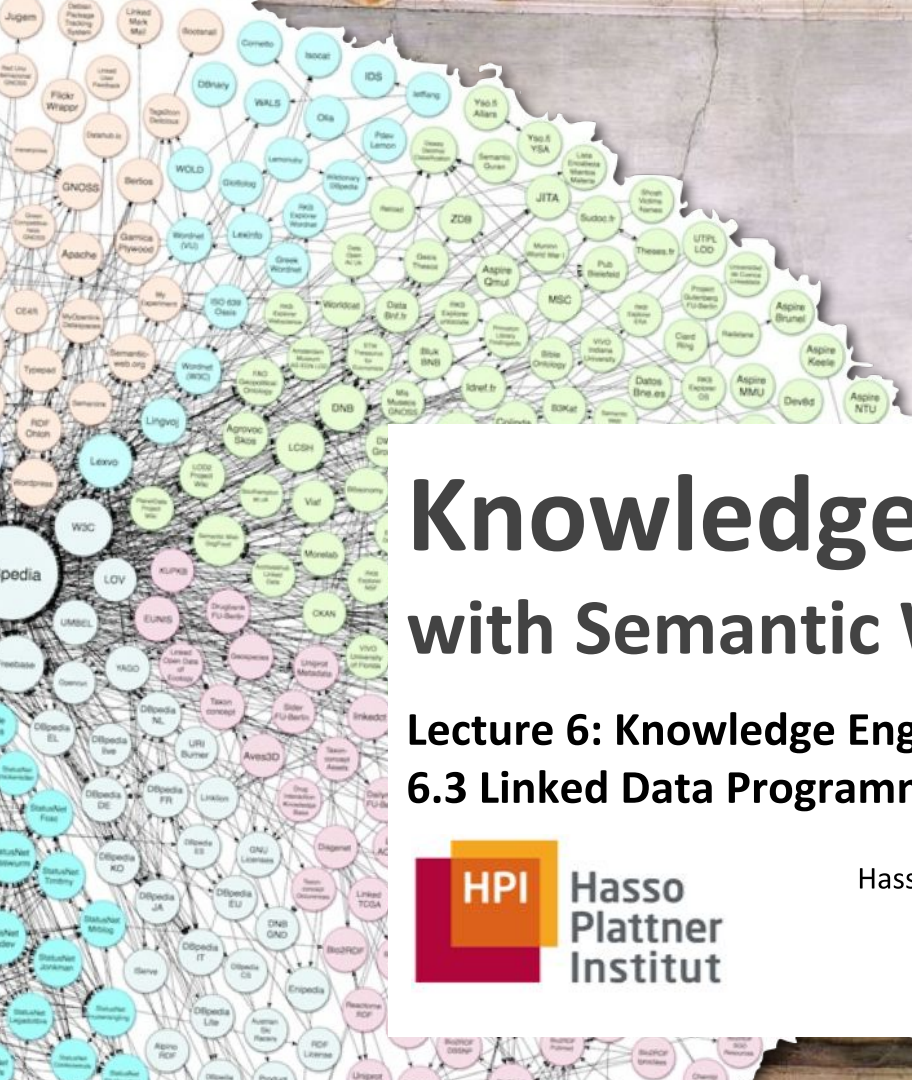




Knowledge Engineering with Semantic Web Technologies

Lecture 6: Knowledge Engineering 6.3 Linked Data Programming



Dr. Harald Sack
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Autumn 2015

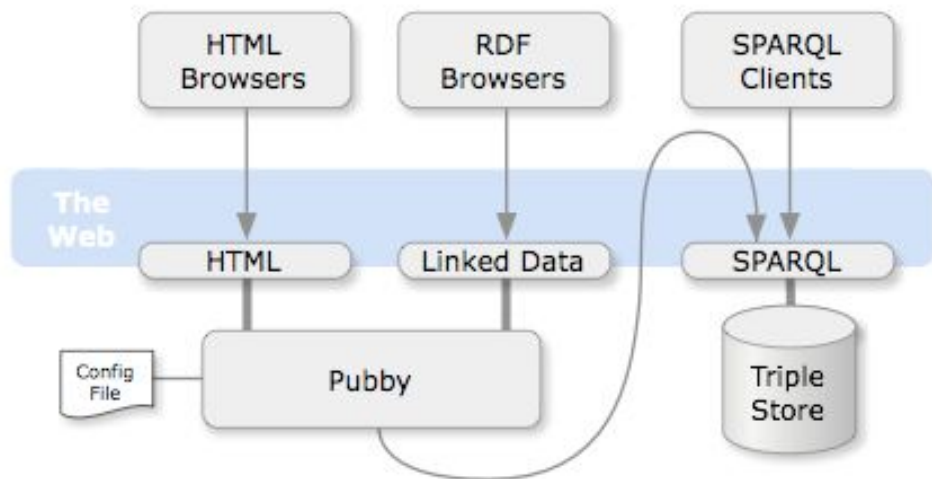
Linked Data Publication in the Web

- **Native publication**
 - Triple Stores (SPARQL Endpoints)
 - OpenLink Virtuoso, Sesame, Fuseki, ...
- SPARQL endpoints are a **RESTful Web Services**
 - HTTP GET Request with SPARQL query
 - Result available as
 - XML, JSON, plaintext (SPARQL Select/Ask)
 - RDF/XML, NTriples, Turtle, N3 (SPARQL Describe/Construct)
 - Data format can be determined via HTTP Accept Header
e.g. `Accept: application/sparql-results+json`
(or via parameters of the SPARQL query)

Linked Data Publication in the Web

- **Native publication**

- Triple Stores
 - OpenLink Virtuoso, Sesame, Fuseki, ...
- Linked Data Endpoints
 - Pubby, Jetty,...

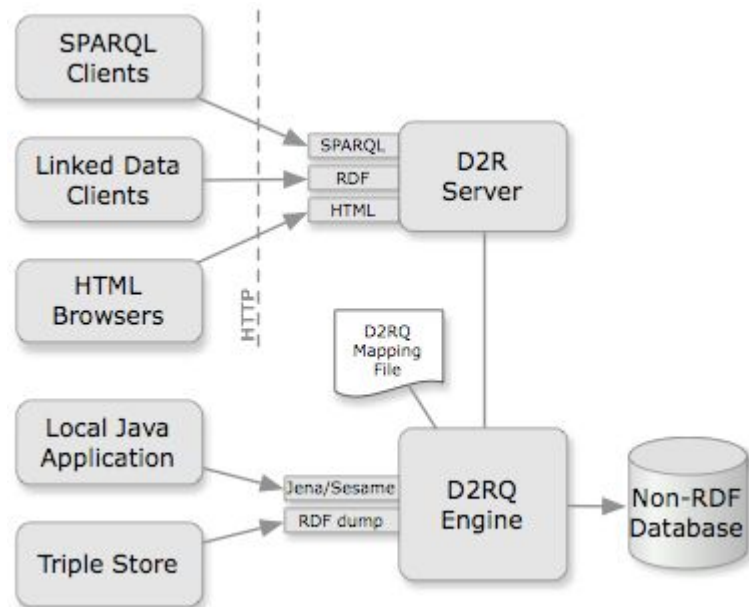


<http://wifo5-03.informatik.uni-mannheim.de/pubby/>

Linked Data Publication in the Web

- **Native publication**

- Triple Stores
 - OpenLink Virtuoso, Sesame, Fuseki, ...
- Linked Data Endpoints
 - Pubby, Jetty, ...
- D2R Servers
 - D2RQ, RDF2RDF, ...



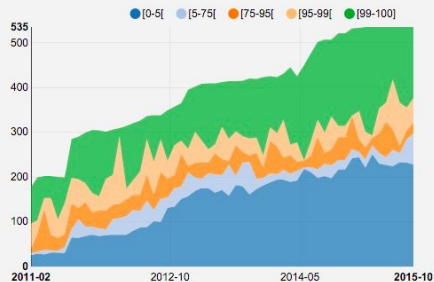
Linked Data Publication in the Web

SPARQL ENDPOINTS STATUS > DATAHUB.IO

Last update: Thu Nov 12 2015 10:32:27 GMT+0000 (GMT)

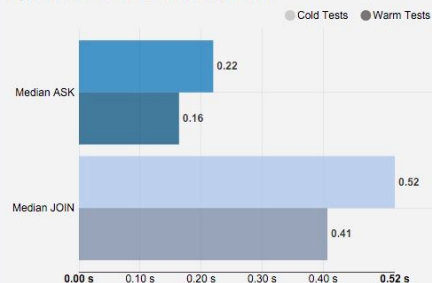
► Description:

AVAILABILITY

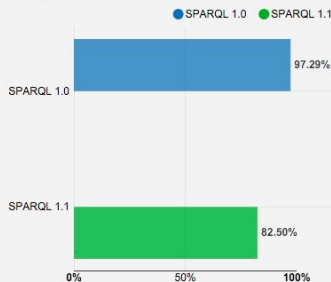


PERFORMANCE

10,000 is the most common result-size threshold



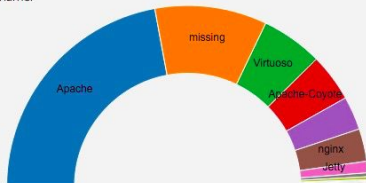
INTEROPERABILITY



DISCOVERABILITY

13.58% of the endpoints provide a VoID description
10.94% of the endpoints provide a SD description
79.06% of the endpoints have **no** description

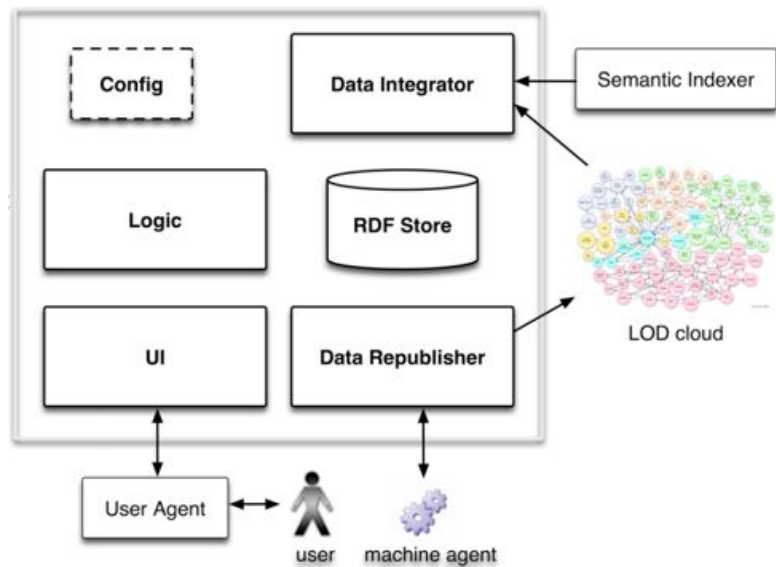
Server name:



Linked Data Driven Web Applications

- Required Components:

- **Local RDF Store**
 - caching of results
 - permanent storage
- **Logic** (Controller) and
- **User Interface** (=Business Logic)
 - (not LOD specific)
- **Data Integration component**
 - get data directly from LOD-Cloud or
 - via Semantic Indexer
- **Data Republishing component**
 - write back application dependent data into the Web of Data



Linked Data Driven Web Applications

- The easiest way is to make use of a suitable library:
 - SPARQL Javascript Library
http://www.thefigtrees.net/lee/blog/2006/04/sparql_calendar_demo_a_sparql.html
 - ARC for SPARQL (PHP)
<https://github.com/semsol/arc2/wiki>
 - dotNetRDF (C#)
<http://dotnetrdf.org/>
 - Jena/ARQ (Java)
<http://jena.sourceforge.net/>
 - Sesame (Java)
<http://rdf4j.org/>
 - **SPARQL Wrapper (Python)**
<http://sparql-wrapper.sourceforge.net/>

Linked Data Programming Example

- The easiest way is to make use of a suitable library:
 - **SPARQL Wrapper (Python)**
<http://sparql-wrapper.sourceforge.net/>
- Access to Linked Data via **SPARQL endpoints**
 - let's choose **DBpedia** (just for simplicity...)
<http://dbpedia.org/sparql>
- ...now we have to think of a simple example...

Linked Data Programming Example

- **Example Application:**

- Build a simple application that looks for **today's birthdays** of famous people, as e.g. **scientists**
- Create a **list of scientists**, whose birthday is today including some **additional information**, as e.g.
 - Year of Birth
 - Short description
- Let's create a simple (local) **web page** for the task, which can be displayed in the browser
- we use **Python** and the **SPARQL Wrapper library**

HPI Hasso Plattner Institut



SPARQL Queries

Virtuoso SPARQL Query Editor

[About](#) | [Namespace Prefixes](#) | [Inference rules](#) | [iSPARQL](#)

Default Data Set Name (Graph IRI)

Query Text

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
SELECT distinct ?birthdate ?thumbnail ?scientist ?name ?description WHERE {
?scientist rdf:type dbo:Scientist ;
           dbo:birthDate ?birthdate ;
           rdfs:label ?name ;
           dc:description ?description
FILTER ((lang(?name)="en")&&(lang(?description)="en")&&(STRLEN(STR(?birthdate))>6)&&(SUBSTR(STR(?birthdate),6)=SUBSTR(STR(bif:curdate('')),6))) .
OPTIONAL { ?scientist dbo:thumbnail ?thumbnail . }
} ORDER BY ?birthdate
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see [details](#).)

Results Format:

Execution timeout:

milliseconds (values less than 1000 are ignored)

Options:

☒ Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see [details](#))

Copyright © 2015 [OpenLink Software](#)

Virtuoso version 07.20.3214 on Linux (i686-generic-linux-glibc212-64), Single Server Edition

```

#!/usr/bin/python
# -*- coding: utf-8 -*-

import urllib2
from datetime import datetime
from SPARQLWrapper import SPARQLWrapper, JSON, XML, N3, RDF

sparql = SPARQLWrapper("http://dbpedia.org/sparql")

sparql.setQuery("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

Select distinct ?birthdate ?thumbnail ?scientist ?name ?description WHERE {
?scientist rdf:type dbo:Scientist ;
          dbo:birthdate ?birthdate ;
          rdfs:label ?name ;
          dc:description ?description
FILTER ((lang(?name)="en")&&(lang(?description)="en")&&(STRLEN(STR(?birthdate))>6)&&(SUBSTR(STR(?birthdate),6)=SUBSTR(STR(bif:curdate('')),6))) .
OPTIONAL { ?scientist dbo:thumbnail ?thumbnail . }
} ORDER BY ?birthdate
""")

sparql.setReturnFormat(JSON)
results = sparql.query().convert()

# Create HTML output
print '<html><head><title>Scientific Birthdays of Today</title></head>'

#extract Weekday %A / Month %B / Day of the Month %d by formatting today's date accordingly
datum = datetime.today().strftime("%A %B %d")
print '<body><h1>Scientific Birthdays of {}</h1>'.format(datum)

print '<ul>'

for result in results["results"]["bindings"]:
    if ("scientist" in result):
        #Create Wikipedia Link
        url = "http://en.wikipedia.org/wiki/" + result["scientist"]["value"].encode('ascii', 'ignore').split('/')[1]
    else:
        url = 'NONE'
    if ("name" in result):
        name = result["name"]["value"].encode('ascii', 'ignore')
    else:
        name = 'NONE'
    if ("birthdate" in result):
        date = result["birthdate"]["value"].encode('ascii', 'ignore')
    else:
        date = 'NONE'
    if ("description" in result):
        description = result["description"]["value"].encode('ascii', 'ignore')
    else:
        description = ' '
    if ("thumbnail" in result):
        pic = result["thumbnail"]["value"].encode('ascii', 'ignore')
    else:
        pic = 'http://upload.wikimedia.org/wikipedia/commons/7/7a/Question_Mark.gif'

    #parse date as datetime
    dt = datetime.strptime(date, '%Y-%m-%d')
    print '<li><b>{}</b> --  <a href="{}">{}</a>, {}</li>'.format(dt.year, pic.replace("300px", "60px"), url, name, description)

print '</ul>'
print '</body></html>'

```

Linked Data Programming Example

← → ↺ file:///Users/harald/Downloads/sparqlwrapper-1.7.3/scripts/scientists.html

Scientific Birthdays of Friday November 13

- 1715 --  [Dorothea Erxleben](#), Physician
- 1821 --  [William Hillebrand](#), German physician and botanist
- 1822 --  [Herman H. J. Lyngbe](#), Danish bookseller
- 1835 --  [Robert Kennicott](#), American zoologist
- 1836 --  [Alfred M. Mayer](#), American physicist
- 1849 --  [Hugo Weidel](#), Austrian chemist
- 1878 --  [Max Dehn](#), German-American mathematician

Linked Data Driven Web Applications

- **Alternative:** simple example with Jena ARQ and Java:

```
import com.hp.hpl.jena.query.*;

String service = "..."; // address of the SPARQL endpoint
String query = "SELECT ..."; // your SPARQL query
QueryExecution e = QueryExecutionFactory. sparqlService(service, query)

ResultSet results = e.execSelect();
while ( results.hasNext() ) {
    QuerySolution s = results.nextSolution();
    // ...
}

e.close();
```



04: Metadata and Semantic Annotation

OpenHPI - Course Knowledge Engineering with Semantic Web Technologies

Lecture 6: Knowledge Engineering