Painless URI Dereferencing Using the DataTank

Pieter Colpaert(✉), Ruben Verborgh, Erik Mannens, and Rik Van de Walle

Department of Electronics and Information Systems, Multimedia Lab,
Ghent University - iMinds, Gaston Crommenlaan 8 Bus 201,
9050 Ledeberg, Ghent, Belgium
pieter@irail.be, {ruben.verborgh,erik.mannens}@ugent.be

Abstract. If we want a broad adoption of Linked Data, the barrier to conform to the Linked Data principles need to be as low as possible. One of the Linked Data principles is that URIs should be dereferenceable. This demonstrator shows how to set up The DataTank and configure a Linked Data repository, such as a turtle file or SPARQL endpoint, in it. Different content-types are acceptable and the response in the right format is generated.

Keywords: Linked data · Semantic web · Dereferencing · Linked data fragments

1 Introduction

Following the trend towards Open Data, Linked Data and the Semantic Web, more and more organizations are publishing their data to the Web. The academic world and standardization bodies have defined various frameworks and principles in order to do so: e.g., the Linked Data (LD) principles [2], hypermedia and REST principles [3], Linked Data Fragments (LDF) principles [6] or the RDF framework[1]. In our lab at Ghent University we are involved in projects on Linked Data which request often to do the same over and over again, adhering to the same set of principles. In this paper we are introducing *tdt/triples*, a project which takes away the pain of having to implement these principles for publishing data over and over again.

First we are going to describe how the perfect triple looks like according to principles. In the related work, we are discussing what projects already take care of these principles and why we believe *tdt/triples* will be a great benefit to a lot of tools. In the next section, we introduce the *tdt/triples* project. Next, the live demo is described and, finally, a conclusion is formulated.

2 The Perfect Triple

For each concept identifier within a triple – a fact described using RDF – a URI is used. In the design issues with Linked Data[2] Tim Berners-Lee, inventor

[1] A primer on RDF: http://www.w3.org/TR/rdf11-concepts/.
[2] http://www.w3.org/DesignIssues/LinkedData.html

DOI: 10.1007/978-3-319-11955-7_39

of the World-Wide Web, introduces dereferencing: "When someone looks up a URI, provide useful information". According to the W3C's note on dereferencing HTTP URIs[3], the act of retrieving a representation of a resource identified by a URI is known as dereferencing that URI. This way, user agents can retrieve facts about the concept the URI is identifying.

Different kinds of user agents are surfing the Web. For a URI about a certain government service, a user agent may be for example looking for news facts. The user agent is going to prefer an *application/rss+xml* representation over a *text/html* representation. The server which serves the representations of the URI may also have its preferences: it may prefer to send *text/html* over *application/rss+xml* as the latter has for instance a slow implementation on the server. The HTTP protocol supports this *content negotiation* using its *Accept* headers. RDF also has different serializations in which triples can be defined. Using content negotiation, different representations of the data should be provided.

These URIs also need to be discoverable. Using DCAT, a W3C vocabulary to describe data catalogs, and VoID, the W3C Vocabulary of Interlinked Datasets, dataset can be made discoverable.

LDF [6] makes the Web of Data Web-scale by allowing clients to query datasets. A basic LDF server has to provide 3 things: data that corresponds to a basic triple pattern, meta-data that consists of the (approximate) total triple count and controls that lead to all other fragments of the same dataset.

The perfect triple is a triple whom's URIs are dereferenced through the HTTP protocol. According to REST principles, the responses to requests towards these URIs have to be cacheable. Furthermore, different serializations and representations of the data have to be provided through content negotiation in order to make the data easy to consume for all machines. Hypermedia controls are needed when dereferencing the URI which provide affordances to the URIs in the triple. Finally, meta-data has to be provided in order to query the data according to the LDF principles and meta-data about the the dataset needs to be given using the DCAT and VoID ontologies.

3 Related Work

Pubby[4] is a Java project written by Richard Cyganiak and Chris Bizer in order to dereference URIs in a triple store with a SPARQL endpoint that supports *DESCRIBE* queries. It supports content negotiation[5] and it follows the REST and hypermedia principles. Pubby does not provide meta-data through the hypermedia interface using DCAT or VoID. Known limitations are that it only supports SPARQL endpoints which support *DESCRIBE*, multiple datasets may not work as expected and hash URIs are not supported. Furthermore, the visualization of the triples in the HTML representation are not extensible[6].

[3] http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14
[4] http://wifo5-03.informatik.uni-mannheim.de/pubby/
[5] https://github.com/cygri/pubby/blob/master/src/main/java/de/fuberlin/wiwiss/pubby/negotiation/ContentTypeNegotiator.java
[6] https://github.com/cygri/pubby/issues/20

Amongst all limitations, only supporting SPARQL endpoints is a problem as for instance ontologies are commonly written by hand in a turtle file and uploaded on a server without triple store, or some data are not stored in a triple store at all and needs to be accessed in another way.

Triplify[7] [1] is a light-weight Data Publication platform. It is a small plugin for Web applications, which reveals the semantic structures encoded in relational databases. Triplify is focused on converting unstructured data from Web applications, stored in a relational database, to RDF. While the project also lowers the barrier for adoption of the Linked Data technology, it does not provide any solution to dereferncing new URIs.

Virtuoso[8], a triple store built by OpenLink, provides optional URI dereferencing on top of their triple store software. In a document[9], they describe their implementation details exposing RDF data and bridging the "Linked Data Web" and the traditional "Document Web". While it is an interesting project, it only works on top of Virtuoso.

Another project is the DBpedia viewer [4] and is used by the, at the time of writing, beta version of DBpedia live. The main drawback for reusing this project, is that the code[10] is written specifically for DBpedia's Virtuoso instance.

4 Demo

For this demonstrator, we have chosen a URI namespace, http://triples.demo.thedatatank.com/, further referred to with the prefix *triples:*. On this namespace, it is our goal to provide a hypermedia interface which dereferences URIs defined in an RDF file. For public transit stop points, we want two LDP[11] containers: one describing the stoppoints of the Belgian railway company NMBS, *triples:NMBS* and one describing the stoppoints of the Dutch railway company NS, *triples:NS*. In these containers, the locations and names of these stop points are given.

The *tdt/triples*[12] project can be installed over *tdt/core*[13] instance. After installation using the composer[14] command on a (apache, mysql, PHP5.4+) stack, the instance can be configured using the API at the relative path */api/triples*. POSTing a new resource to this collection can be done by providing the type of the reader and its parameters. Self-documentation on the sources and their parameters can be found in the discovery document, which in this case can be found at *triples:discovery*.

In the beginning of our example project, our turtle file is small and only contains two transit agencies. In order to dereference the URIs in it, we configure The

[7] http://triplify.org/
[8] http://virtuoso.openlinksw.com/
[9] http://virtuoso.openlinksw.com/whitepapers/deploying%20linked%20data.html
[10] https://github.com/lukovnikov/dbpedia-vad-i18n
[11] The Linked Data Platform (LDP) is a vocabulary to describe read-write Linked Data resources based on HTTP access: http://www.w3.org/TR/ldp/.
[12] http://github.com/tdt/triples
[13] http://github.com/tdt/core
[14] A dependency manager for PHP: http://getcomposer.org.

DataTank to dereference our file by POSTing its location to *triples:api/triples*, thus, no need to set up a triple store and SPARQL endpoint. The turtle file used in this demo can be found at *triples:demo.ttl*. When going to the specific URIs, only the triples mentioning this URI as a subject are returned and visualized. The visualization within the HTML uses a javascript library called rdf2html[15]. It is an extensible library which creates a certain type of visualization for an, or a couple of, ontologies. Through content-negotiation, the same data can be requested through different formats: *text/html*, *application/json*, *application/ld+json*, *application/rdf+xml*, *text/turtle*...

A basic LDF server has to be able to solve triple patterns. The DataTank solves this using this template: {*URI*}*?subject=*{*subject*}*&predicate=*{*predicate*} *&object=*{*object*}. This way, we have also introduced URIs for triples and triple patterns. By dereferencing these, we can provide more information: the answer to the triple pattern can be given, but also the (approximation of the) count, provenance and extra hypermedia handles can be given. The basic LDF server is advertised in the data catalog feed on the server accessible at *triples:api/dcat*.

When in the course of the project, a triple store is set up, an extra source can be configured on */api/triples*. This new SPARQL endpoint source can work in parallel to the turtle file, until the turtle file is not needed any longer.

5 Getting Started

tdt/triples is an extension of *tdt/core*. Both repositories are available at *github.com*. The only requirement is a standard Apache – MySQL – PHP stack. Composer[16], a dependency manager for PHP, can be used in order to set up The DataTank on your namespace. First, fill out *app/config/database.php* after downloading *tdt/core*. Next, run *composer require tdt/triples* followed by a *composer install*. If all goes well, your namespace should now return an empty The DataTank.

By using the interface at *api/admin* you can configure tdt/core resources. At the time of writing, supported sources are: a CSV file, a SHP file, an XML file, a JSON file, an XLS file, a JSON-LD file, an N3 file, a SPARQL endpoint and custom written PHP code to fetch and transform data. With the added *tdt/triples* functionalities, you can now configure extra sources at *api/triples*. Posting Listing 1 to *api/triples* will result in all the configured file's triples to be added to your namespace. The result, with this particular file, can be seen at for example *triples:NMBS*, which will now return the relevant triples, including VoID, Hydra[17] and DCAT metadata.

[15] http://github.com/tdt/rdf2html

[16] http://getcomposer.org

[17] A vocabulary to describe hypermedia APIs used by LDF: http://www.hydra-cg.com/.

```
{
  type:"Turtle",
  uri: "http://triples.demo.thedatatank.com/demo.ttl"
}
```

Listing 1. An example of a tdt/triples source configuration.

Up to date documentation can be found at http://docs.thedatatank.com.

6 Conclusion

This demonstrator introduces *tdt/triples*, a new project which lowers the bar to start dereferencing URIs on a certain namespace. Several principles to publish data on the Web were discussed and applied to the project, amongst others: the Linked Data principles, the hypermedia and REST principles and the Linked Data Fragments principles. The readers or visitors of the demonstrator are able to set up *tdt/triples* and can add their RDF file or SPARQL endpoint to the configuration. They are able to see an automatically generated representation requested through content-negotiation, see the rdf2html view and follow links. All processing is done on the fly: when the file or store changes, the URI representation does too.

We hope that *tdt/triples* and *rdf2html* will get a broad uptake amongst ontology maintainers, dataset owners and other people that have to dereference URIs as part of their Linked Data project. Future work lies in enabling the read/write Web within *tdt/triples* using distributed versioning techniques. This work has already started in our lab with R&Wbase [5]: distributed version control for triples.

Acknowledgments. We would like to thank Jan Vansteenlandt and Michiel Vancoillie of the Open Knowledge Foundation Belgium for the close collaboration on the implementation of tdt/triples. The research activities described in this paper were funded by Ghent University, iMinds, the Flemish department of Economics, Science and Innovation (EWI), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT) and the European Union.

References

1. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: lightweight linked data publication from relational databases. In: Proceedings of the 18th international conference on World Wide Web, pp. 621–630. ACM (2009)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. Int. J. Semant. Web Inf. Syst. **5**(3), 1–22 (2009)
3. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. (TOIT) **2**(2), 115–150 (2002)
4. Lukovnikov, D., Kontokostas, D., Stadler, C., Hellmann, S., Lehmann, J.: DBpedia viewer - An integrative interface for DBpedia leveraging the DBpedia service eco system. In: Proceedings of the 7th Workshop on Linked Data on the Web (2014)

5. Vander Sande, M., Colpaert, P., Verborgh, R., Coppens, S., Mannens, E., Van de Walle, R.: R&Wbase: Git for triples. In: Proceedings of the 6th Workshop on Linked Data on the Web (2013)
6. Verborgh, R., Vander Sande, M., Colpaert, P., Mannens, E., Van de Walle, R.: Web-scale querying through linked data fragments. In: Proceedings of the 7th Workshop on Linked Data on the Web (2014)