rdfedit: User supporting web application for creating and manipulating RDF instance data

Oliver Pohl

Berlin School of Library and Information Science, Humboldt-Universität zu Berlin
oliverpohl@ibi.hu-berlin.de

Abstract. rdfedit is a web application running on Django, rdflib and jQuery DataTables that supports novices in the field of Semantic Web technologies with the creation of RDF instance metadata. By utilizing the Semantic Web search engine Sindice, rdfedit can transform literals into URIs, fetch triples from external resources and import them into the user's local graph. Metadata experts can easily configure these features of rdfedit to fit their preferences regarding metadata schemata, so metadata creators with few knowledge about Semantic Web technologies can create RDF data in a fast and consistent manner while also following the Linked Data principles.

Keywords: RDF Editor, Metadata, Sindice

Published Version: Pohl, O.: rdfedit: User Supporting Web Application for Creating and Manipulating RDF Instance Data. In: Closs, S., Studer, R., Garoufallou, E., and Sicilia, M.-A. (eds.) Metadata and Semantics Research. pp. 54–59. Springer International Publishing (2014).

1 Introduction

More than a decade has passed since [3] has shared his ideas of the Semantic Web. Since then, the Semantic Web has grown and evolved [5, 12, 19]but it is still is not living up to its potential. When analyzing the application of RDFa throughout the Web in 2012, [13] only determined 4.7 per cent of all websites inside the Bing corpus made use of that technology. A year later, [4] conducted a similar study, this time analyzing websites within the Common Crawl Index[1], receiving only slightly higher percentage of 5.6.

The reasons on why Semantic Web technologies are not used more thoroughly are diverse. For once, businesses do not see a valid reason for adopting such technologies [21]. Moreover, these technologies require background knowledge regarding the Semantic Web, but underlying concepts are hard to explain and hard to understand for non-experts [1, 18]. [16] predicted that the adoption of the Semantic Web will slowly rise by 2014 and finally be accepted as a mainstream

[1] http://commoncrawl.org/common-crawl-url-index/

technology by 2019. Hence Semantic Web technologies seem to be on the verge of the innovators phase to the early adopters phase when grouping the Semantic Web user base into [17]'s diffusion model of innovations.

The opinions on how to get more people to use Semantic Web technologies vary. [11] suggests to make the generation of Semantic Web metadata completely invisible for users, since it should be "a by-product of everyday computer use". Following this approach, a toolkit that generates RDFa when users create new content was added to the content management software Drupal [6, 10]. Other opinions state that such technologies should provide an additional value to the user [22] and should be founded on features that users are already acquainted with, such as relational database tables [14].

With the intention of supporting the growth of the Semantic Web by helping novices in that field access related technologies, I programmed the web application *rdfedit*[2] to helpt users create and edit RDF data. The core idea behind *rdfedit* is to make the creation of RDF instance data easier for people who know little to nothing about Semantic Web and associated technologies. The responsibility of creating good quality RDF data is distributed among Semantic Web or metadata experts and the users who create the actual RDF data, thus only having a few expert users who maintain and manage the system and many novice users who feed it with data.

A target audience for this application are cultural heritage institutions who want to create or enrich RDF metadata. Only a few experts will suffice to configure *rdfedit* in a way so it can be used by non-experts in order to produce data in a schema of their institution's preference.

2 Impelementation & Features

rdfedit is running on Django while heavily making use of rdflib[3], a python library to process RDF data, and jQuery DataTables[4] for displaying the tabular interface. Users currently can upload already existing RDF/XML [9] files for further editing or simply start with a new, empty graph. When uploading a graph, Django/rdflib receives the file, extracts all triples and parses it to the RDF table so the users can start working with their graphs in their web browser. The table consists of three columns: subject – predicate – object, showing the simple triple like structure of RDF.

On upload of an already existing graph, all triples are analyzed and preprocessed for later auto-completion. Since many subject and predicate URIs will be used multiple times, full URIs are being suggested to the user when they start typing them during a triple addition. This way, users do not have to re-type or re-paste URIs, thus saving time and adding valid and consistent data.

[2] rdfedit can be accessed at: `http://141.20.126.167/rdfedit/index`
Source code repository available at: `https://github.com/suchmaske/rdfedit`

[3] `https://github.com/RDFLib`

[4] `http://www.datatables.net/`

In general, *rdfedit* offers basic functionalities like adding, editing and deleting single triples and more complex features like bulk editing, literal-to-URI conversion and aggregation of RDF data from external resources. To make these processes work, an RDF/JSON [7] is being kept in the background and invisible to the user. New jQuery functions were written in order to synchronize the changes made to the table by the user with the RDF/JSON object, hence intertwining the DataTables powered RDF table and the RDF/JSON object with Django and rdflib. When users are done editing their graph, *rdfedit* submits the altered RDF/JSON object back to Django, where rdflib transforms that object to a RDF/XML file and serves it as a download to the user.

Since *rdfedit* aims towards helping the creation of valid and consistent RDF data, users can apply changes made to a single triple to all other affected triples within the same graph. When applying a bulk edit, *rdfedit* looks for all triples containing the old, unaltered URIs in the subject and object column and substitutes all matching cells with the new URI.

rdfedit also utilizes the API of the Semantic Web search engine Sindice[5] to offer the user a literal-to-URI conversion and fetch data from external graphs. Administrators (Semantic Web experts) of *rdfedit* can determine centrally what kind of URIs are appropriate, what data to fetch from where and how imported data should be mapped into the local graph. Hence they ease the burden of good quality metadata creation off the metadata creators (Semantic Web novices), since the latter no longer need to think about what vocabularies to use or what knowledge base might be the most appropriate for their current task. All the experts have to do is to edit two *rdfedit* configuration files (see Table 1). These files influence the outcome of the literal-to-URI conversion and triple aggregation from external resources.

Query Configuration	**Mapping configuration**

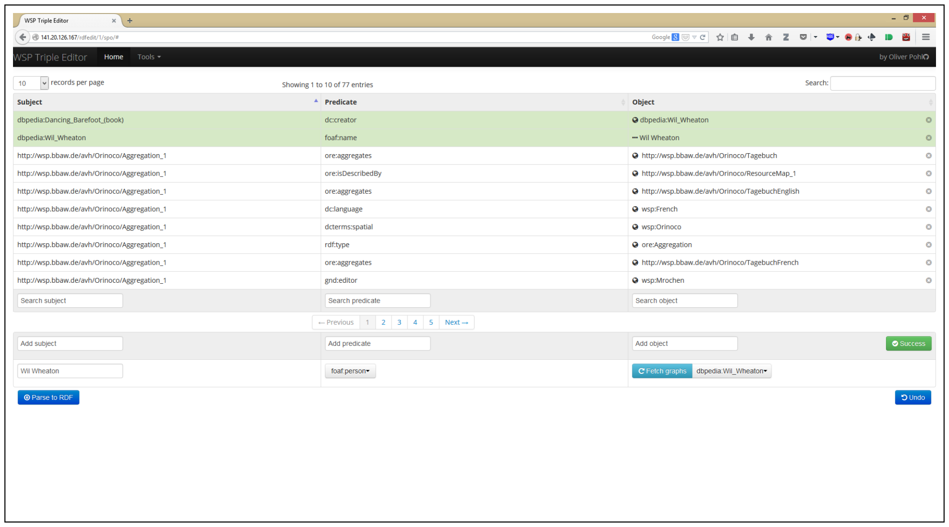
`{`
`  "foaf:person": {`
`    "fq=domain": "dbpedia",`
`    "fq=class": "foaf:person"`
`  },`
`  "dcterms:spatial": {`
`    "fq=domain": "geonames",`
`    "fq=format": "RDF"`
`  }`
`}`
	`{`
	`  "foaf:person": {`
	`    "dbpprop:author": "dc:creator",`
	`    "foaf:name": "foaf:name"`
	`  },`
	`  "dcterms:spatial": {`
	`    "rdfs:isDefinedBy": "dcterms:spatial"`
	`  }`
	`}`

Table 1. Example configurations for fetching graphs and triples from Sindice.com

When the user adds a single new triple, *rdfedit* checks the `query-config.json` file (see Table 1, left) and checks whether the predicate of that new triple exists as a JSON-key in the configuration file. If that is the case, Django composes

[5] `http://sindice.com/developers/searchapiv3`

a query for Sindice accordingly. For example, a user wants to add a new triple about the actor Wil Wheaton. Since he or she knows that Mr. Wheaton is a person, the user chooses to use `foaf:person` as the predicate. The new triple would consist of a literal object: `:subject foaf:person ''Wil Wheaton'' .`. *rdfedit* then queries Sindice with the parameters given by the configuration file: a) only include graphs from the DBPedia, b) only show graphs that are of the class `foaf:person`. The results of that query are being forwarded into the object cell, where the user then can choose one of the result URIs. In this case, he picks `dbpedia:Wil_Wheaton` and the literal `''Wil Wheaton''` is then replaced by the aforementioned URI, following the principle of using "Things, not Strings" [20].



10 records per page — Showing 1 to 10 of 77 entries — Search:

Subject	Predicate	Object
dbpedia:Dancing_Barefoot_(book)	dc:creator	dbpedia:Wil_Wheaton
dbpedia:Wil_Wheaton	foaf:name	Wil Wheaton
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	ore:aggregates	http://wsp.bbaw.de/avh/Orinoco/Tagebuch
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	ore:isDescribedBy	http://wsp.bbaw.de/avh/Orinoco/ResourceMap_1
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	ore:aggregates	http://wsp.bbaw.de/avh/Orinoco/TagebuchEnglish
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	dc:language	wsp:French
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	dcterms:spatial	wsp:Orinoco
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	rdf:type	ore:Aggregation
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	ore:aggregates	http://wsp.bbaw.de/avh/Orinoco/TagebuchFrench
http://wsp.bbaw.de/avh/Orinoco/Aggregation_1	gnd:editor	wsp:Mrochen
Search subject	Search predicate	Search object

← Previous 1 2 3 4 5 Next →

Add subject	Add predicate	Add object	Success
Wil Wheaton	foaf:person	Fetch graphs dbpedia:Wil_Wheaton	

Parse to RDF — Undo

Fig. 1. Screenshot of rdfedit showing the imported triples from `dbpedia:Wil_Wheaton`

Fetching triples from external resources happens in a similar fashion. Relying on the JSON-keys in the `query-config.json` file, *rdfedit* renders a dropdown menu in the browser, which the user can select predefined classes from. In combination with the chosen class, the user enters some keywords and *rdfedit* composes and submits a query to Sindice, where relevant graphs are being looked up. This time, the user wants to import a set of triples about Wil Wheaton rather than adding them one by one. He chooses the class `foaf:person`, enters the keywords "Wil Wheaton" and *rdfedit* interacts with Sindice in the same way as described in the literal-to-URIs conversion example. Again, Sindice returns a list of URIs from which the user can choose one that fits his or her intentions best.

When having decided for a URI, *rdfedit* loads the graph behind that URI. At this point, *rdfedit* compares the `mapping-config.json` file (see Table 1, right) with the fetched graph and extracts all triples that have the same value, as the

JSON-keys for the user picked class of the configuration file. Since the user in our example has chosen the class `foaf:person` the triples with the predicates `dbpprop:author` and `foaf:name` will be extracted from the graph. Additionally, *rdfedit* transforms the original predicates to the values of those JSON-keys. Here, `dbpprop:author` would be remapped to `dc:creator` (see Fig. 2). This feature encourages the reuse of already existing data and supports the Linked Data principles as described by [2].

3 Discussion & Outlook

It is unclear whether the features relying on Sindice will continue to work in the future, since the team behind the Semantic Web search engine announced the termination of support for their product [8]. Therefore alternate solutions for the tasks described have to be evaluated and implemented. It might be necessary to use SPARQL queries instead of utilizing the Sindice API to ensure a better longevity of *rdfedit*. For *rdfedit* the application of SPARQL is controversial, because its original intention was to not bother users with SPARQL's complexity, whether they are Semantic Web experts or newcomers. Still, users should be able to create RDF data in a semi-automatic way: while the application fetches and inserts the data automatically, the user should still maintain in control and check whether the imported data is actually useful.

At the moment, empirical usability tests are planned but have not been conducted yet. In the near future, *rdfedit* will be evaluated by researchers of the Berlin Brandenburg Academy of Science (BBAW), whose disciplinary background, technical affinity and Semantic Web expertise vary heavily. It is expected that most of the participating BBAW members will fall into [15]'s user cube category of novice users since they have a background in the (Digital) Humanities rather than Computer or Information Science related fields.

With the results of the evaluation, *rdfedit* will further try to make the creation of RDF instance data easier for people new to Semantic Web technologies. It will continue trying to lessen the responsibility for those newcomers for creating good quality data, for instance by implementing interoperability with metadata crosswalks for the expert side. Furthermore, the jQuery written for the manipulation of RDF/JSON are planned to be extracted and published separately under an Open Source license, in order to contribute the growth and application of Semantic Web technologies.

References

[1] Benjamins, D.V.R., Radoff, M., Davis, M., Greaves, M., Lockwood, R., Contreras, D.J.: Semantic Technology Adoption: A Business Perspective. In: Domingue, J., Fensel, D., Hendler, J.A. (eds.) Handbook of Semantic Web Technologies, pp. 619–657. Springer Berlin Heidelberg (Jan 2011), `http://link.springer.com/referenceworkentry/10.1007/978-3-540-92913-0_15`

[2] Berners-Lee, T.: Linked Data - Design Issues (2006), `http://www.w3.org/DesignIssues/LinkedData.html`

[3] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. The Scientific American (May) (2001), http://www.scientificamerican.com/article/the-semantic-web/

[4] Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., Völker, J.: Deployment of RDFa, Microdata, and Microformats on the Web – A Quantitative Analysis. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) The Semantic Web – ISWC 2013, pp. 17–32. No. 8219 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2013), http://link.springer.com/chapter/10.1007/978-3-642-41338-4_2

[5] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far:. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009), http://www.igi-global.com/article/linked-data-story-far/37496

[6] Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and Consume Linked Data with Drupal! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009, pp. 763–778. No. 5823 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2009), http://link.springer.com/chapter/10.1007/978-3-642-04930-9_48

[7] Davis, I., Steiner, T., Le Hors, A.J.: RDF 1.1 JSON Alternate Serialization (RDF/JSON): W3C Working Group Note 07 November 2013. Tech. rep., W3 (Nov 2013), http://www.w3.org/TR/rdf-json/

[8] Franzon, E.: End of Support for the Sindice.com search engine: history, lessons learned, and legacy (May 2014), http://semanticweb.com/end-support-sindice-com-search-engine-history-lessons-learned-legacy-guest-post_b42797

[9] Gandon, F., Schreiber, G.: RDF 1.1 XML Syntax: W3C Recommendation 25 February 2014. Tech. rep., W3 (2014), http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

[10] Havlik, D.: Building Environmental Semantic Web Applications with Drupal. In: Hřebíček, J., Schimak, G., Denzer, R. (eds.) Environmental Software Systems. Frameworks of eEnvironment, pp. 385–397. No. 359 in IFIP Advances in Information and Communication Technology, Springer Berlin Heidelberg (Jan 2011), http://link.springer.com/chapter/10.1007/978-3-642-22285-6_42

[11] Hendler, J.: Agents and the semantic web. IEEE Intelligent systems 16(2), 30–37 (2001)

[12] Jentzsch, A., Cyganiak, R., Bizer, C.: State of the LOD Cloud. Tech. rep. (2011), http://lod-cloud.net/state/

[13] Mika, P., Potter, T.: Metadata Statistics for a Large Web Corpus. LDOW 937 (2012)

[14] Newman, A.: A Relational View of the Semantic Web (Mar 2007), http://www.xml.com/pub/a/2007/03/14/a-relational-view-of-the-semantic-web.html

[15] Nielsen, J.: Interactive Technologies : Usability Engineering. Morgan Kaufmann, Saint Louis, MO, USA (1994), http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10712933

[16] Nixon, L., Volz, D.R., Ciravegna, F., Studer, R.: Future Trends. In: Domingue, J., Fensel, D., Hendler, J.A. (eds.) Handbook of Semantic Web Technologies, pp. 581–618. Springer Berlin Heidelberg (Jan 2011), http://link.springer.com/referenceworkentry/10.1007/978-3-540-92913-0_14

[17] Rogers, E.M.: Innovativeness and Adopter Categories. In: Diffusion of innovations, pp. 267–299. Free Press, New York, 5. edn. (2003)

[18] Salo, D.: Soylent Semantic Web Is People! In: SWIB 2013. Hamburg, Germany (Nov 2013), http://www.slideshare.net/cavlec/soylent-semantic-web-is-people-with-notes?utm_source=slideshow&utm_medium=ssemail&utm_campaign=upload_digest

[19] Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the Linked Data Best Practices in Different Topical Domains. In: The Semantic Web - ISWC 2014. Springer, Trentino, Italy (2014), http://dws.informatik.uni-mannheim.de/fileadmin/lehrstuehle/ki/pub/SchmachtenbergBizerPaulheim-AdoptionOfLinkedDataBestPractices.pdf

[20] Singhal, A.: Introducing the Knowledge Graph: things, not strings (Feb 2012), http://googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html

[21] Sletten, B.: Keep On Keeping On (Jan 2014), http://semanticweb.com/keep-on-keepin-on_b41339

[22] Stuart, D.: Facilitating access to the web of data : a guide for librarians. London : Facet Publ., London, 1. publ. edn. (2011)