Making the Semantic Data Web Easily Writeable with RDFauthor

Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org

Abstract. In this demo we present RDFauthor, an approach for authoring information that adheres to the RDF data model. RDFauthor completely hides syntax as well as RDF and ontology data model difficulties from end users and allows to edit information on arbitrary RDFa-annotated web pages. RDFauthor is based on extracting RDF triples from RDFa-annoted Web pages and transforming the RDFa-annotated HTML view into an editable form by using a set of authoring widgets. As a result, every RDFa-annotated web page can be made writeable, even if information originates from different sources.

1 Introduction

To a large extent the overwhelming success of the World Wide Web was based on the ability of ordinary users to author content easily. In order to publish content on the WWW, users had to do little more than to annotate text files with few, easy-to-learn HTML tags. Unfortunately, on the semantic data web the situation is slightly more complicated. Users do not only have to learn a new syntax (such as N3, RDF/XML or RDFa), but also have to get acquainted with the RDF data model, ontology languages (such as RDF-S, OWL) and a growing collection of connected RDF vocabularies for different use cases (such as FOAF, SKOS and SIOC).

Previously, many applications were developed to ease the syntax side of semantic authoring [4,2]. The *RDFauthor* approach[1] is based on the idea of making arbitrary XHTML views with integrated RDFa annotations editable. *RDFa* [1] is the W3C Recommendation, which allows to combine human and machine-readable representations within a single XHTML document. RDFauthor builds on RDFa by preserving provenance information in RDFa representations following the named-graph paradigm and by establishing a mapping from RDFa view representations to authoring widgets. On configurable events (such as the clicking of a button or moving over a certain information fragment with the mouse) the widgets will be activated and allow the editing of all RDFa-annotated information on the Web page. While editing, the widgets can access background information sources on the Data Web in order to facilitate the reuse of identifiers or to encourage the interlinking of

[1] http://aksw.org/Projects/RDFauthor

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 436–440, 2010.

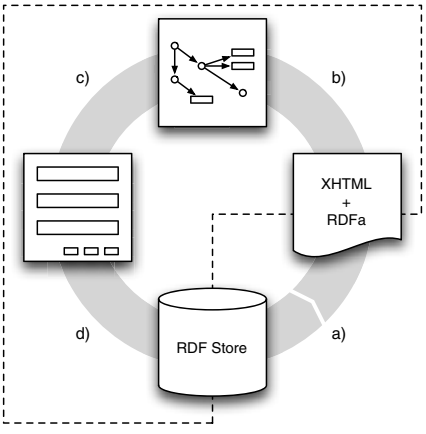

Fig. 1. Editing cycle for an RDFa-enhanced web page. The processes involved are a) page creation and delivery, b) client-side page processing, c) form creation and d) update propagation via SPARQL/Update. The dashed line encloses the processes carried out by RDFauthor.

resources. Our resource editing widget, for example, suggests suitable, previously defined resources derived from calls to the Sindice Semantic Web index [5]. Once editing is completed, the changes are propagated to the underlying triple stores by means of the SPARQL/Update language.

2 System Architecture Overview

The basic cycle of how web pages are edited with RDFauthor is depicted in figure 1. It is composed of four distinct processes, of which (b) client-side page processing, (c) widget selection and (d) form creation and update propagation via SPARQL/Update [3] are handled by RDFauthor.

In order to link RDFa annotations on the page to the respective querying/update services (i.e. SPARQL/Update endpoints), we propose the use of the `link` HTML tag with an `about`-attribute to identify the named graph, a `rel`-attribute with the value `update:updateEndpoint` and a `href`-attribute with the URL of the respective SPARQL/Update endpoint. Another option to declare graph metadata is the use of empty `span`- or `div`-elements together with the RDFa attributes inside the body of the page.

The initiation of the RDFauthor editing processes happens through element-based or page-wide trigger events, which can be e.g. the clicking of an edit button, hovering over an RDFa element or a bookmarklet. When the user finishes the editing process, all widgets involved are asked to update the page graph with their changes. The difference between the original and modified page graphs are calculated (i. e. added statements, removed statements), yielding a diff graph. The associated store to each graph is then updated with the respective diff graph by means of SPARQL/Update operations.

3 Use Cases

We demonstrate the benefits of RDFauthor, by showcasing the integration of the approach into two Semantic Web applications and a standalone RDFauthor bookmarklet facilitating the collection of RDF data from arbitrary RDFa-annotated websites.

3.1 OntoWiki

OntoWiki [2][2] is a tool for browsing and collaboratively editing RDF knowledge bases. It differs from conventional Semantic Wikis in that OntoWiki uses RDF as its natural data model instead of Wiki texts. Information in OntoWiki is always represented according to the RDF statement paradigm and can be browsed and edited by means of views, which are generated automatically by employing the ontology features, such as class hierarchies or domain and range restrictions. OntoWiki adheres to the Wiki principles by striving to make the editing of information as simple as possible and by maintaining a comprehensive revision history. It has recently been extended to incorporate a number of Linked Data features, such as exposing all information stored in OntoWiki as Linked Data as well as retrieving background information from the Linked Data Web. Apart from providing a comprehensive user interface, OntoWiki also contains a number of components for the rapid development of Semantic Web applications, such as the RDF API Erfurt, methods for authentication, access control, caching and various visualization components.

RDFauthor is used in OntoWiki both in the generic resource property view as well as in extensions which render resources in a domain-specific way (e. g. specific visualizations for SKOS concepts or FOAF persons). In order to perform the integration, we have extended OntoWiki in two ways:

1. We extended the default properties view for resources and all other views with RDFa attributes to annotate which data is presented as well as to link the graph to the internal update service. Since OntoWiki is entirely based on an RDF store, this extension was easy to implement. Likewise, all extension developers had to extend their views, e. g. for SKOS concepts.
2. We included RDFauthor by referencing it in the head of every OntoWiki page and adding JavaScript edit buttons on every page where data should be editable.

3.2 vCard and Publication Mashup

In order to showcase the simultaneous authoring of information from multiple sources, we integrated RDFauthor into the text-based wiki application WackoWiki[3]. WackoWiki is often used in small and medium-sized companies as well as in small organizations such as research groups.

[2] Online at: `http://ontowiki.net`

[3] `http://wackowiki.org`

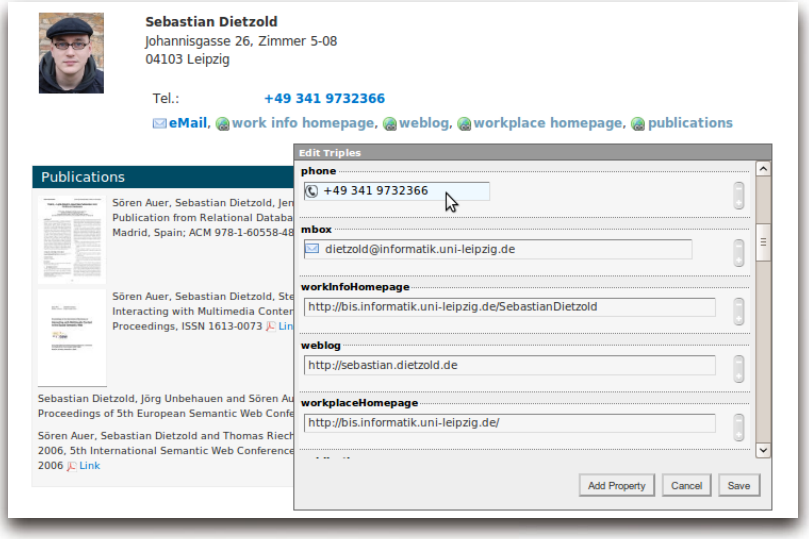

Fig. 2. RDFa-enhanced FOAF vCard and publications mashup with statements from different named graphs. In addition to generic literal and resource widgets, authoring widgets for the URI schemes `tel:` and `mailto:` hide the URI syntax.

The AKSW research group uses a WackoWiki installation for its entire web page (`http://aksw.org`) and integrates external data sources by means of so-called WackoWiki actions. Actions are small scripts that prepare some content and output it at the given position in the wiki page. In addition, actions are able able to fetch data from external resources, thus allowing us to use structured information on different places in the wiki, e. g. by presenting the last publications selected by author, project or topic.

While integrating and presenting this information is easy and covered by many applications and techniques, full read/write integration of such external resources is tackled by RDFauthor. By employing RDFauthor, users of our wiki are able to edit both the wiki page and the structured information in one place and avoid using different web applications for one edit task and with different data. The RDFauthor GUI on top of a Wacko Wiki page is depicted in figure 2.

3.3 Data Collection from RDFa Websites

Another interesting usage scenario, which is more concerned with collecting data instead of editing, is described in this section. Most of the RDFa-enabled pages on the web do not yet contain provenance and update information. However, RDFauthor also allows to use an arbitrary update endpoint, which does not necessarily have to match the originating endpoint.

Since a SPARQL/Update-capable RDF store and a target graph is all the information required for using RDFauthor, it is easy to embed these into a

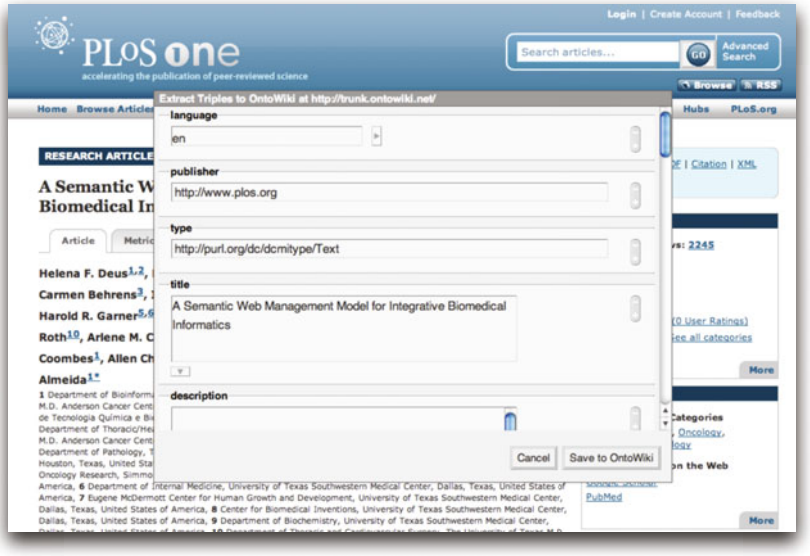

Fig. 3. RDFauthor overlay with widgets for triples extracted from a PLoS web page

bookmarklet used to initialize the editing process. In this case, the number of possible SPARQL/Update endpoints is limited to those under one's control. RDFauthor extracts the data from any page visited and displays the edit form. The data can be revised and unwanted statements can be removed from the view. Saving works, however, differently: instead of propagating the changed data back to the original source, it is sent to one's own RDF store and saved into the previously set-up graph.

References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and Processing. In: Recommendation, World Wide Web Consortium (W3C) (October 2008), `http://www.w3.org/TR/rdfa-syntax/`
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
3. Seaborne, A., Manjunath, G.: SPARQL/Update: A language for updating RDF graphs. Technical Report Version 5: 2008-04-29, Hewlett-Packard (2008)
4. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting Collaborative Ontology Development in Protégé. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 17–32. Springer, Heidelberg (2008)
5. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the Open Linked Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)