Using JavaScript RDFa Widgets for Model/View Separation inside Read/Write Websites

Sebastian Dietzold, Sebastian Hellmann and Martin Peklo

Universität Leipzig, Department of Computer Science
Johannisgasse 26, D-04103 Leipzig, Germany,
{dietzold|hellmann}@informatik.uni-leipzig.de

Abstract. As more and more websites start to embed RDFa content in their web application view, the need arises to provide a more extensive way for viewing and editing this semantic content independently from the remainder of the application. We present a JavaScript API that allows the independent creation of editing widgets for embedded RDFa, which adds a new edge to Web development in the context of the Semantic Web. As an addition, the API also provides sound update methods that allow on-the-spot model synchronization between client and server.

1 Introduction

Future web applications will use more and more client side application logic to obtain fast and responsive user interfaces. While it is currently possible to use semantic technologies at the backend, web application developers can benefit greatly from a way to preserve semantics between the server and the client. Especially functionalities such as extensive editing, including type verification, can be implemented more efficiently (i.e faster and without additional requests) on the client side. By using RDFa [1] or Microformats [3], semantically enriched data can be transferred back and forth and e.g. JavaScript enables the modification on the client side. To return semantic data from the client side back to the server, developers need APIs to modify both the semantic model and the corresponding view from the RDF model in a consistent and independent way.

We present an RDF-API for JavaScript (rdfapi-js[1]), which uses an RDFa parser to create a JavaScript object as an in-memory representation of the RDF model from the website. The API gives developers the methods to modify the in-memory model, roll out the changes to the (X)HTML view and report them back to the server. Furthermore, RDFa widgets are used to create custom editing functions based on the local model in the (X)HTML representation. These widgets give a degree of independence to developers, since it is now possible to add appropriate editing widgets based on the semantic content of the RDFa enhanced website without interfering with the development of the view of the application. This greatly improves maintainability and customization. A useful

[1] *rdfapi-js* is available at `http://powl.svn.sf.net/viewvc/powl/trunk/rdfapi-js/`

implication we would also like to mention is the redundancy of creating separate views for editing, because updating can often be solved with the proposed widgets.

2 API Overview

The RDF-API for JavaScript implements the RDF JSON specification proposal from the Talis n2 Wiki[2] for data representation and allows for RDF manipulation methods. The RDF model is extracted by a modified version of the parser from the RDFa Javascript implementation[3] provided by the W3C.

The added methods can be grouped as follows:

Statement modification: These methods are used to modify the content of the model. We have implemented methods for adding one or more statements to a model and for deleting one or more statements from the model.

Namespace management: Since RDFa incorporates the use of namespaces, methods on in-memory models for adding, deleting and applying namespaces are needed. The Talis RDF JSON specification proposal does not include namespace declarations, so we have added this to our object notification. We have implemented methods for adding and deleting namespaces as well as for converting models from qualified names to full URIs (and back) according to the current namespace declarations.

Model comparison and check: These methods are used to check the consistency of a given model and to compare two models and produce a statement diff for the update service. At this time, we do not care about bnodes in the diff, e.g. by using blank node enrichment [6] or minimum self-contained graphs [5]. This could be added in the future.

We complete the rdfapi-js by adding a small set of administration methods for counting, (X)HTML output and debugging.

3 Usage in a Semantic Wiki

We use rdfapi-js in our semantic wiki application OntoWiki [2]. OntoWiki is able to handle plugins for the visualization of resources from specific RDF schema (e.g. a FOAF person). These plugins are simple templates which have access to the attributes of the resource that is to be rendered. Without rdfapi-js, plugin developers have the choice to develop either a special edit template or let users edit the attributes with the generic table-edit view. With rdfapi-js, a new option is to enable view templates for inline editing just by adding RDFa markup.

---

[2] `http://n2.talis.com/wiki/RDF_JSON_Specification`

[3] `http://www.w3.org/2006/07/SWD/RDFa/impl/js/`

The following steps describe an examplary editing process:

1. The complete page with RDFa markup and the link to the rdfapi-js and a control script are created on the server and transferred to the client.
2. The client receives the page and loads the rdfapi-js as well as the application control script through a `script`-link.
3. rdfapi-js parses the page and creates an in-memory model from the RDFa markup.
4. An onclick handler is added next to every RDFa Literal found in the page.
5. By clicking on the event handler, an editing widget is displayed as an overlay to the existing page (see Fig.1).

Fig. 1. An RDFa enhanced vCard: plain (left), with highlighted statements and widget (right)

6. The user submits the new value to the widget. The widget modifies the in-memory model by using statement modification methods of rdfapi-js. These API methods apply the changes to the in-memory model as well as modify the (X)HTML view.
7. The modified in-memory model triggers a new submit button which enables the user to submit the change request to the server. If pressed by the user, a statement difference is calculated and transferred to the server via an asynchronous HTTP request to an update service. The current implementation sends two JSON models to the server, one with the statements which have to be deleted, another with the statements which have to be created on the server. The execution of these atomic add / delete actions is left to the server update interface.

Widgets are not limited to the editing of plain literals. According to the in-memory model, different widgets can be loaded (e.g. for dates or geo locations). The idea of such widgets is to achieve independence from the page template so that a growing widget library can enhance every application which uses rdfapi-js.

4 Conclusion and Future Work

The presented API includes methods for customized client-side edit widgets to modify embedded RDFa. It tackles the synchronization of a locally changed

RDF model with the model on the server. The main advantage is the separation of edit functions from the data view. Widgets can be created for different datatypes and properties. These widgets add new possibilities for flexible user interaction. The API is included in OntoWiki, but can be used in other Semantic Web applications as well. Without making a promising statement, we have already considered to propose the API as a full alternative to an editing view. However, the problems that have to be overcome are numerous such as multi-user synchronization, security and user validation. Although solutions for these problems have to be provided, they are clearly out of the scope of this API, since these problems have to be solved on the server side.

Future work includes the improvement of the diff computation algorithm to support bnodes and the enhancement of the widget library with a set of widgets for common attributes. The (X)HTML rollout of in-memory changes is currently limited to plain literals and resource relations. This limitation has to be resolved in order to support more complex RDFa markup. To generalize the editing process, future work should include support for SPARUL [4] queries in addition to transferring JSON model objects for change requests.

References

1. Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and Processing. W3C Working Draft, W3C, 2008. `http://www.w3.org/TR/rdfa-syntax/`.
2. Sören Auer, Sebastian Dietzold, and Thomas Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer, 2006.
3. Rohit Khare and Tantek Çelik. Microformats: a pragmatic path to the semantic web. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 865–866. ACM, 2006.
4. Andy Seaborne and Geetha Manjunath. SPARQL/Update – A language for updating RDF graphs. Technical report, Hewlett-Packard, January 2008. V4.
5. Giovanni Tummarello, Christian Morbidoni, Reto Bachmann-Gmür, and Orri Erling. RDFSync: Efficient Remote Synchronization of RDF Models. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 537–551. Springer, 2007.
6. Max Völkel, Carlos F. Enguix, Sebastian Ryszard Kruk, Anna V. Zhdanova, Robert Stevens, and York Sure. SemVersion - Versioning RDF and Ontologies. Technical report, AIFB, University of Karlsruhe, 2005.