Interactive Construction of Semantic Widgets for Visualizing Semantic Web Data

Timo Stegemann **Juergen Ziegler** **Tim Hussein** **Werner Gaulke**

University of Duisburg-Essen
Lotharstr. 65, 47057 Duisburg, Germany
{firstname.lastname}@uni-due.de

ABSTRACT

The rapidly growing amount of semantically represented data on the Web creates the need for more intuitive methods and tools to interact with these data and to use them in standard Web applications. We present a method how users can interactively define personalized views of large semantic data spaces. Specifically, we propose X3S as a technique and format for specifying 'semantic widgets' that integrate querying and filtering of semantic data with defining their layout and presentation style. In addition, an editor has been developed that allows to create X3S templates in a direct manipulation style. The editor and the underlying format are evaluated against existing approaches by comparing their functional capabilities as well as in an initial user study.

ACM Classification Keywords

H.3.3; H.5

Keywords

Semantic Web; Data Exploration; Data Visualization; Semantic Widgets; Semantic Stylesheets

INTRODUCTION

In the course of the decade which has passed since the vision of a Semantic Web was first articulated by Berners-Lee [3], the amount of data on the Web represented with semantic techniques has increased enormously. In contrast to the conventional Web, which links web pages as complete documents, the Semantic Web links data based on a uniform data model. This model consists of elementary statements in 'subject-predicate-object' form, expressed with RDF (Resource Description Framework), and uses shared vocabularies defined as ontologies by means of RDFS (RDF-Schema) and OWL (Web Ontology Language) [7]. Linking the individual statements results in a single, huge data graph. This is often referred to as the Web of Data.

Improved techniques for automatically translating existing datasets into semantic formats have brought about a significantly accelerated growth, leveraging large existing information pools such as encyclopedias (e.g. DBPedia which is extracted from Wikipedia [1]), geographical databases, bibliographies, product data (for an overview see, e.g. [6]) and many others. The LOD (Linking Open Data) initiative has invested considerable effort in interconnecting different datasets, resulting in the LOD Cloud with around 295 connected datasets and 31 billion triples (as of fall 2011).

While the original objectives of the Semantic Web were mainly directed at making Web information machine-processable, it is increasingly recognized that linked open data and other semantic data pools are valuable information sources that can be used interactively by end users. This creates a need for methods and tools that allow users to interact with Semantic Web data directly, rather than through a web application that integrates and delivers data in standard web pages.

In contrast to conventional Web front ends, user interfaces and search tools for Semantic Web data can exploit the semantics of the underlying data structure to let users explore the data under different perspectives and formulate more targeted and complex queries. The potential for making semantic data explorable has been demonstrated by a variety of tools that use the different semantic relations for creating search facets, allowing users to flexibly filter the data. Visual techniques for formulating complex queries have been developed, for instance, in prior work of ours [8]. In addition to providing improved search and exploration capabilities, however, semantic representations can also enable users to construct their own views of a semantic data pool, which can either be used for exploring the data in some user-defined visual configuration or for presenting (and potentially editing) a more or less complex cut-out of the large RDF based data graph. With the development described in this paper, we aim at providing a solution for the latter problem by offering users a technique for composing complex, reusable query and presentation patterns ('semantic widgets'), that show a coherent part of a data graph in a user-defined format.

In this paper, we propose a novel technique for interactively defining reusable templates for querying and presenting semantic data. We call this technique X3S which stands for 'XSL-transformed SPARQL results and Semantic Stylesheets'. In contrast to CSS (Cascading Style Sheets), known from web design, X3S includes directives for filtering data by using SPARQL (SPARQL Protocol And RDF Query Language) as well as for presenting and decorating them (by using XSL (Extensible Stylesheet Language) transformations and CSS). For creating X3S files, we developed an editor enabling the user to create complex templates for querying

http://doi.acm.org/10.1145/2305484.2305510

and presenting semantic data in an intuitive, direct manipulation-style manner.

X3S and the associated editor targets user groups who have some knowledge of Semantic Web concepts and data structures but who may not be familiar with specific formats or query languages. Apart from exploring semantic data directly, a main purpose of the technique is to define reusable views that can be published in standard Web pages. A potential user may thus be, for example, a shop developer or administrator who needs an easy to use method for publishing semantically represented product data.

The rest of this paper is organized as follows: First, we review existing work specifically related to the X3S technique, the semantic stylesheet format proposed, and identify areas for improvement. The X3S technique and format are presented in the next section, followed by a description of an editor implementing the technique. Finally, we analytically compare it against related developments with respect to a set of requirements.

RELATED WORK

A number of tools for browsing the Semantic Web and for presenting RDF based data have been developed so far. Most of these developments either fall into the class of Semantic Web browsers or semantic data integration/mash-up tools [10]. While the latter aim at integrating different Semantic Web data sources, Semantic Web browsers serve a similar purpose for semantic data as conventional Web browsers do for HTML data.

These tools enable users to explore large corpora of data in an associative manner. However, the presentation of the data is mostly limited to tabular listings of all instances, properties, and property values. Examples of such tools are Tabulator[1] [4] and Disco[2]. While these tools are capable of providing overviews of data sets, they do not support filtering of instances or producing specific layouts and styles. These limitations prevent a more rich interaction with semantic data, which is our target. There are currently only few projects with a similar focus. We will describe them in the following paragraphs.

The Xenon project [12] has introduced an ontology described as 'RDF stylesheet language'. The goal of Xenon is to display semantic data in a human usable way and facilitate changing the representation according to the user's needs. Based on the concept of XSLT, the authors created a RDF-based stylesheet, which defines methods for transforming RDF-data. This includes concepts such as *lenses* and *views*. The purpose of lenses is to select items from instance sets, whereas views describe the visualization of data elements.

The RDF vocabulary Fresnel was developed as a successor of Xenon. Fresnel also uses lenses to select data but dropped the views concept and with it the possibility to embed visual markup such as HTML [11]. As a replacement, the *formats* concept is introduced. It is used for formatting data and to enrich the data with additional information for rendering. For example, a format defines whether the selected lens data should be handled as a link, an image, or as a text. This additional information is used by the browser for creating the final visualization. The data might be presented, for instance, as a table or graph, depending on the decision of the browser. To select data, lenses can use a special Fresnel Selector Language (FSL) or standard SPARQL queries.

With OWL-PL [6], a language for transforming RDF/OWL data into (X)HTML is introduced. The language is strongly inspired by XLST and has the main goal of providing a simple transformation language for semantic data. OWL-PL allows the combination of transformational and representational markup. Stylesheets are converted into HTML by means of a server-side Java application.

LESS [2] supports a complete workflow, ranging from creating and processing templates to sharing templates between users. The declarative template language LeTL (LESS template language) is based on the Smarty Template Engine[3], which simplifies the separation of data and presentation of PHP-projects. It can process and transform semantic data from RDF documents or data requested by SPARQL queries. LESS provides an editor for creating LeTL templates. The editor shows available properties of the RDF data or of the SPARQL result. The properties can be directly used in LeTL code and combined with HTML markup to generate the final HTML output. The created LESS template is saved on the LESS server.

In contrast to the other approaches introduced in this section, Dido (Data-Interactive DOcument) [9], does not process semantic data in RDF format, but provides means for direct manipulation and creation of data as well as for including presentation directives. The embedded data are saved as key-value pairs in JSON (JavaScript-Object Notation). Selection and visualization are defined with lenses and views similar to Xenon. In the editor, the user can directly choose how the selected data should be presented, for example, as a list or table.

All approaches reviewed so far have in common that they define their own templating languages for handling semantic data. Only Dido and LESS include editors for creating new stylesheets. These editors, however, still require a considerable level of knowledge in semantic technologies. Editors for the other formats have not been developed so far.

Having their benefits in the respective use cases, these approaches are hard to re-implement and the editors are not very user friendly. This created the motivation for our own development. X3S provides a standards-based open format for reusable semantic stylesheets and can be used with arbitrary RDF-based data sources. The reference implementation provides an easy to use editor enabling non-experts to intuitively explore semantic data visually and to define personalized templates.

[1] http:www.w3.org/2005/ajar/tab

[2] http://www.wiwiss.fu-berlin.de/bizer/ng4j/disco

[3] http://www.smarty.net/

Based on the analysis of the limitations of existing tools we derived a set of requirements for our own development. One important requirement is the possibility to select arbitrary properties from a specified resource, instead of receiving and displaying all available data. The technique should further allow filtering the data in a flexible manner. One of the strengths of semantic data is the possibility to define one resource as a property of another resource. The technique should therefore also provide means for nesting several properties.

DEFINING SEMANTIC WIDGETS WITH X3S

In order to construct reusable views on some set of semantic data, several steps need to be performed in a workflow-like fashion. The running example used throughout this article is based on RDF data about universities in Germany and related information. Assume a user wants to explore this information space to learn more about some of these universities and the cities, in which they are located. The user first searches for 'university' as a concept to define an entry point into the RDF data graph, then adds various descriptive (data) properties and related concepts by following object properties such as the study programs offered, the city where the university is located and cultural attractions in that city. The user then applies a visual style to this data configuration and stores the complete configuration as a reusable semantic widget. This widget can then be applied to every instance of a German university in the graph and simplifies the comparison of returned information.

X3S is a technique for describing these different steps in a single template. It also provides a format for serializing and storing templates for future reuse.

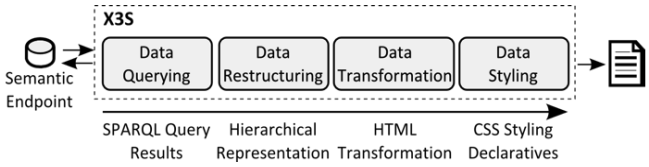

Figure 1: X3S workflow to transform semantic data into a HTML document.

Internally, X3S comprises a workflow consisting of four steps: querying, restructuring, transforming, and styling, as illustrated in Figure 1. While established formats and technologies can be used for most of these steps, a new technique for restructuring the query results was needed to allow a consistent, redundancy-free presentation. The complete workflow is finally stored as an XML document.

Querying

The first part of a X3S document contains a SPARQL encoded query. Any SPARQL compliant data source can be used. The SPARQL server handles the processing of different RDF formats and provides a standardized interface to the RDF data. With X3S, a single SPARQL query, which can request arbitrary properties, is defined. The result of this SPARQL query is a set of property-value pairs that are formatted in a W3C (World Wide Web Consortium) conform XML representation. Figure 2 gives an example for such a query and its result.

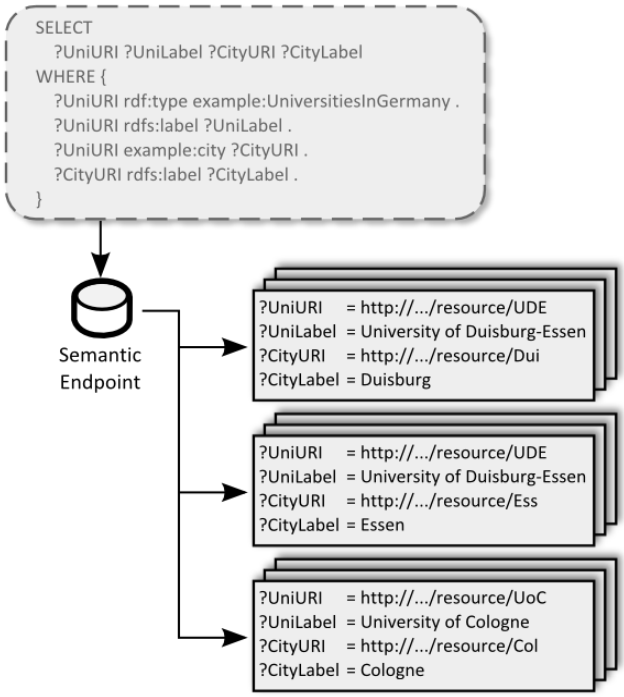

Figure 2: A SPARQL server returns several result sets for a single query.

In our example, we might request the resource URI (Uniform Resource Identifier) and label for each university and the respective city, where the university is located in. The SPARQL server returns a list of result sets, where every set contains values for one university URI, university label, city URI and city label. The URIs are particularly relevant as they represent the resource itself. Properties can be filtered, limited or ordered by SPARQL commands, which are similar to common SQL commands. By requesting the data from a SPARQL endpoint, the problem of different RDF formats and non-unique serializations can be solved.

Restructuring

The SPARQL results are returned as a list of flat result sets, without any hierarchical structure. To allow straightforward processing of the data, it should be re-structured into a hierarchical form. As the SPARQL results may contain redundancies, several result sets must also be merged. The university example provides a case, where two result sets have to be merged. Because the university of Duisburg-Essen is located in two cities (Duisburg and Essen), two result sets with partially redundant information are returned. Both result sets contain information about the university but one contains information about the city of Duisburg, the other about the city of Essen. This behavior occurs in every case where a single property provides more than one value.

To handle this behavior and to simplify the retrieval of the SPARQL query's hierarchy, an additional XML structure is used. All SPARQL results are mapped onto this structure. It is defined by the hierarchy of the previously created

SPARQL query. For the XML structure, every property from the query is represented by an XML element. The root-element as well as all object properties contains their properties as child nodes. The SPARQL query and the XML structure are associated through the variable names defined in the query. These variable names are used in the XML structure as attribute values.

In our example, the XML element, representing the university URI is the root element. The elements that represent the university label and the city URI are child elements. The city URI element is a parent element of the city label element. Finally, all result sets are aggregated into a single XML structure that contains all values as attributes of XML nodes. Figure 3 shows this restructuring process.

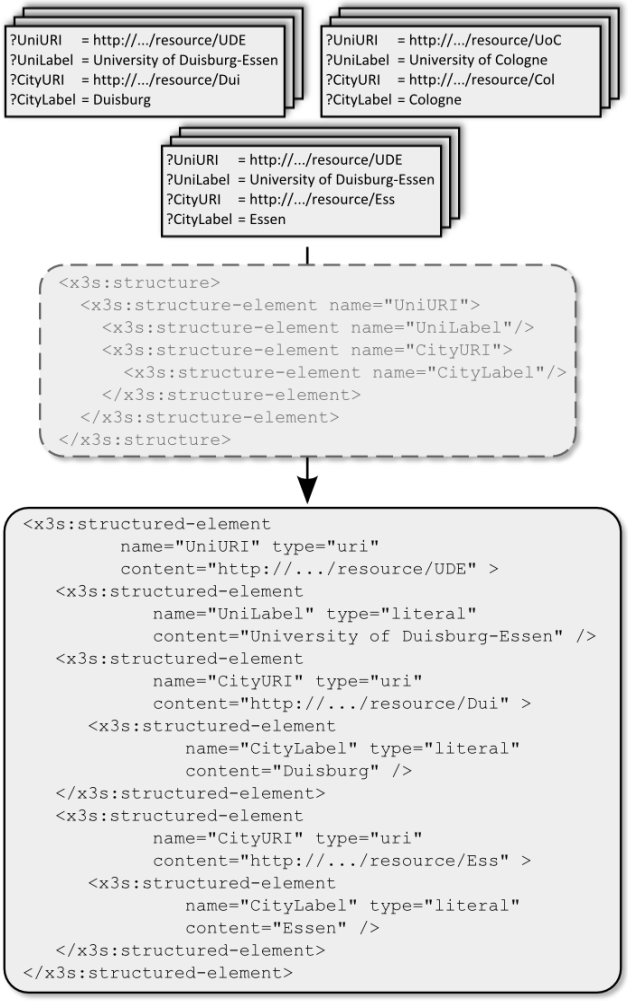

Figure 3: SPARQL results with redundancies are mapped onto a predefined hierarchical XML structure that meets the hierarchy of the primary SPARQL request. All values are saved in the node's attributes.

Direct processing of partially redundant data via XSLT, which is used in the next process step, would require grouping mechanisms that are possible but become very complex when nesting more than one object into another. Therefore all result sets are mapped onto a predefined hierarchical XML structure that meets the hierarchy of the SPARQL request without redundancies.

The restructuring process is the only step in the X3S workflow that is not based on common techniques.

Transformation

We use XSLT 1.0 for transforming the resulting XML trees or fragments into HTML, including CSS classes and IDs that can later be used to decorate the HTML (see Figure 4).

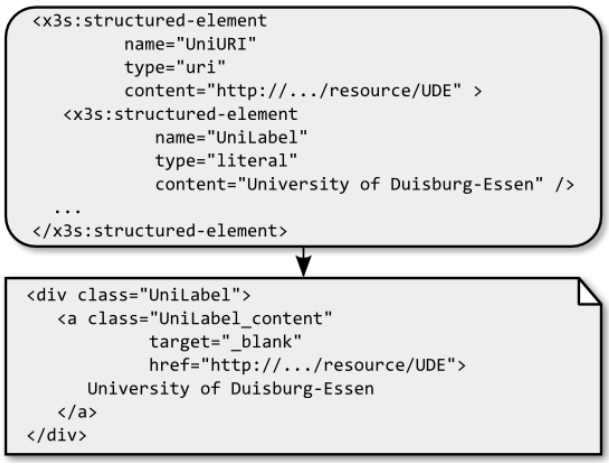

Figure 4: The restructured Data is transformed via XSLT into an HTML document.

The restriction to XSLT 1.0 instead of higher version was made to facilitate the implementation of an X3S viewer with nearly every system or programming language. The support for more recent XSLT standards is still not satisfying.

The function of the XSL transformation is to transform the restructured results into HTML code that can be put into the body-tag of a HTML document. One possibility is to keep the hierarchical structure and transform the restructured query results into nested div-tags. This is the option, we have chosen for our reference implementation. Another possibility would be to display the data as a table. Furthermore in this step, our implementation converts URLs into clickable links or displayable images. Also proper CSS class names are inserted into HTML elements for a latter styling via CSS.

Styling

In order to control the appearance of the documents, CSS can be specified. All CSS declaratives will be embedded into to the final HTML documents header. There is no further linking or other complex processing of information in this last step. The previous transformation step is responsible to ensure that the HTML and CSS code are fitting together.

AN EDITOR FOR SEMANTIC STYLESHEETS

To support users who are not experts in semantic technologies or in the template creation process we developed an editor as a rich internet application based on Adobe Flex. The editor runs in any web browser with Flash capabilities and can make use of existing semantic data repositories that provide a SPARQL interface. The editor enables users to explore the content of the repository in a visual manner and persistently store the properties of interest

in X3S format as *semantic widgets* or *stylesheets*. These widgets can later be used as templates for exploring similar data. The editor is available online[4].

Figure 5 shows the main window of the editor. We use point-and-click interactions instead of complex text-based directives: Drag-and-drop is used for selecting properties from a list of all possible attributes for the particular RDF class (Figure 5 A). The properties can be filtered and sorted by type, relevance[5], or in alphabetic order. In order to provide visual cues, datatype and object properties are displayed with different icons. We chose a visualization based on a file editor metaphor, where object properties, which contain again other properties as values, are displayed as folders and datatype properties, which contain only literals, are displayed as files. From this information the user can directly derive how properties can be nested.

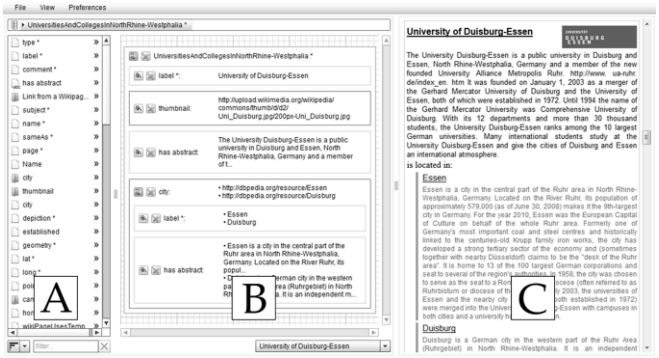

Figure 5: X3S editor while creating a semantic stylesheet, working with data from DBpedia.

The fictitious user in our example wants to select the set of all German universities as a starting point for her or his constructive exploration. The respective SPARQL query returns a list of values for each university. The semantic data corpus holds a large amount of information related to these universities. The user can drag elements from the list into the template located in the central workspace (Figure 5 B). A live preview of the resulting HTML document is displayed on the right side (Figure 5 C).

When adding datatype properties to the template, an appropriate display format can be selected (for instance showing the value of an URI as an image instead of the URI itself). The editor also tries to detect several properties automatically. As an additional feature, dynamic filters can be created based on datatype properties. This may be useful, if only a subset of all possible entities should be displayed (for instance, universities located in cities with a population between 500 000 and 1 000 000). Depending on the type of datatype property (String, Boolean, integer, etc.), widgets such as sliders for those filters are created dynamically.

[4] http://x3s.demos.interactivesystems.info/

[5] Relevance is measured by the frequency of occurrence of this property among all instances of the class.

The templates can finally be decorated by using standardized CSS. The editor supports the user by directly offering graphical shortcuts to common CSS declaratives such as font size. Elements can have defined borders or margins, images can be adjusted in width and height, etc.

FUNCTIONAL ASSESSMENT AND INITIAL EVALUATION

	X3S	Xenon	Fresnel	OWL-PL	LESS	Dido
Separation of Concerns	●	●	●	●	●	○
SPARQL-Endpoint Support	●				●	
Property Selection	●	●	●	●	●	●
Filtering	●	●	●	●	●	●
Styling	●	●		●	●	○
Nested Object-Properties	●	○	○	●	○	
Templating	●	●	●	●	●	●

Table 1: Comparison of the approaches: ● support, ○ partial support, no entry: no support.

We compared X3S to the other approaches described in the related work section (see Table 1). In terms of 'Separation of Concerns', all tools but Dido access their data from a source saved separately. Only X3S and LESS have a 'SPARQL-Endpoint Support' and are able to request their data directly from an external semantic database. All approaches provide means for 'Property Selection', so only a subset of the data can be selected for presentation. 'Filtering' of entries with regard to particular data values is also possible. 'Styling' of the data cannot be specified freely with Fresnel. All other candidates convert the data into HTML and can style them with CSS. It is simple with X3S and OWL-PL to display 'Nested Object-Properties' that are completely integrated into a single stylesheet. With the other procedures, a stylesheet author has to build a separate stylesheet that must be embedded into the primary one. All candidates are able to reuse the stylesheets by a 'Templating' mechanism.

We conducted an initial user study to determine the effectiveness, efficiency and user-friendliness of the X3S editor in comparison to the LESS editor. We chose LESS as a reference point because, in our review, it turned out to be the most closely related system in terms of functionality. The evaluation was designed as a user study in which ten participants had to solve five tasks with both editors (within subject design). All five tasks combined created a semantic widget, where each task covered an essential step of the overall creation process – property selection, data styling, adding static data, filtering, and nesting further objects. No Participant had previous experience of using the evaluated tools.

Using the X3S editor, the participants could immediately solve more than 80% of the tasks. Using the LESS editor instead, only 40% of the tasks could be solved without any assistance.

Across all five tasks, participants needed 7:26 minutes on average to create the whole widget with the X3S editor. With 16:36 minutes on average, it took more than twice as long to do the same work with the LESS editor.

In a finally completed questionnaire, the participants subjectively rated the X3S editor as intuitive to use for the given tasks but felt inadequately supported by the LESS editor.

DISCUSSION

The ongoing growth of the Semantic Web as an extension and complement of the conventional World Wide Web poses the problem of enabling non-expert users to interact with semantic data in an intuitive, yet flexible and powerful manner. In this paper, we have presented a method how users can define personalized views into the large semantic data space. As our main technical contribution, we have presented X3S as a technique and format for specifying 'semantic widgets'. Semantic widgets combine the querying and filtering of large semantic data sets with defining the layout and presentation style of the data. Semantic relationship between data elements can be transformed into suitable presentation structures.

To support the specification of semantic widgets, an editor has been developed that allows the creation of templates for data exploration and visualization based on the X3S specification. X3S supports the separation of data and layout as well as arbitrary filtering of data and visualization via CSS.

The technique presented supports a variety of ways of using semantic information in a wide range of web application scenarios. For example, predefined X3S templates in different configurations can be used for searching product information in the Web of data. Due to the intuitive direct manipulation style of defining semantic widgets, they are suitable for quickly building interfaces both in single-user as well as in collaborative situations, for instance, when teams are exploring data in front of large interactive surfaces.

Future work will firstly focus on optimizing the functionality developed so far. Especially, the combination of dynamically retrieved semantic data and statically added text can be improved. Furthermore, we aim at integrating data from different sources, extending the X3S technique to cover typical mash-up functions. A challenging future extension of the work would be to support transferring a semantic widget defined on some specific configuration of concepts and properties to other data graphs that are similarly structured but have different concept and object property types. This would require techniques for automatically analyzing the content of the new concepts and relations, extending the semantic widget concept towards more intelligent behavior.

REFERENCES

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. DBpedia: A Nucleus for a Web of Open Data. In K. Aberer, K.-S. Choi, N. Noy, et al., eds., The Semantic Web. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, 722-735.
2. Auer, S., Doehring, R., and Dietzold, S. LESS – Template-Based Syndication and Presentation of Linked Data. In ESWC '10: Proceedings of the 7th Extended Semantic Web Conference, number 6089 in Lecture Notes in Computer Science, pages 211 – 224. Springer, 2010.
3. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. Scientific American, May 2001.
4. Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'ommeaux, E. and Schraefel, m.c. Tabulator Redux: Browsing and writing Linked Data. In Proc. WWW 2008 Workshop: LDOW, (2008).
5. Brophy, M. OWL-PL: A Presentation Language for Displaying Semantic Data on the Web. Master's thesis, Lehigh University, 2010.
6. Feigenbaum, L., Herman, I., Hongsermeier, T., Neumann, E., and Stephens, S. The Semantic Web in Action. Scientific American 297, Dec. 2007 (2007), 90-97.
7. Heath, T., and Bizer, C. Linked Data - Evolving the Web into a Global Data Space. Morgan & Claypool Publishers, 2011.
8. Heim, P., Ertl, T., and Ziegler, J. Facet Graphs: Complex semantic querying made easy. In ESWC '10: Proceedings of the 7th Extended Semantic Web Conference, number 6088 in Lecture Notes in Computer Science, pages 288 – 302. Springer, 2010.
9. Karger, D. R., Ostler, S., and Lee, R. The Web Page as a WYSIWYG End-User Customizable Database-Backed Information Management Application. In UIST'09: Proceedings of the 22nd Annual ACM Symposium on User interface Software and Technology, pages 257–260, New York, NY, USA, 2009. ACM.
10. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., and Morbidoni, C. 2009. Rapid prototyping of semantic mash-ups through semantic web pipes. In *Proceedings of the 18th international conference on World wide web* (WWW '09). ACM, New York, NY, USA, 581-590.
11. Pietriga, E., Bizer, C., Karger, D. R., and Lee, R. Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In ISWC '06: Proceedings of the 5th International Semantic Web Conference, volume 4273 of Lecture Notes in Computer Science, pages 158–171. Springer, 2006.
12. Quan, D. and Karger, D. R. Xenon: An RDF Stylesheet Ontology. In WWW '05: Proceedings of the 14th International World Wide Web Conference, 2005.