Browsing the Semantic Web

Ora Lassila
Nokia Research Center Cambridge
Cambridge, MA 02142, USA

Abstract

Presenting RDF graphs to the end user as hypertext enables interactive exploration and inquiry of "semantic" data. In this paper we present a tool that allows users to "see" their RDF data, without having to decipher syntactic representations that are typically not human-friendly. The tool is part of a broader architecture designed for viewing many different types of data in Semantic Web formalisms, for experimenting with the caching of dynamically changing data, and for exploring the possible ad hoc integration of data from multiple sources. Additionally, this tool enables browsing of semantic data on a mobile device, by adapting the hypertext generation to better suit small screens.

1. Introduction and Related Work

The Semantic Web [2] could be characterized as the application of knowledge representation in the context of the World Wide Web. Emphasis is on *machine-based interpretation* of data and *autonomous* operation. It is equally important, however, to think of it as also intended for human users. To quote [2]:

> *The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*

In the case of "spontaneous" information integration – a task well suited to Semantic Web technologies – it is vital for the users to *see* their data. In this paper we will present a tool that allows users to explore Semantic Web data via a mechanism they already know well, namely *browsing*.

RDF [15, 4] is the fundamental building block of the Semantic Web. It can be thought of as *directed, labeled graphs* that can easily be presented as hypertext; at its simplest the presentation only has to rely on the RDF metamodel, making *any* RDF data "browsable" without any special knowledge of the schema(ta) involved. More specialized, *schema-aware* presentations can then be built on top of the basic solution. Enabling browsing is further encouraged by the fact that, despite its simple data model, users and developers are often put off by RDF's cryptic XML-based syntax.

Many tools have been constructed for searching, exploring and querying complex information spaces such as semistructured data representations or collections with rich metadata. These tools often offer a mixture of features such as faceted search, clustering of search results, etc. Examples of such systems include Lore [7, 8], Flamenco [26], and others. More specifically, recent years have also seen a number of "Semantic Web browsers" emerge; examples of this type of software include mSpace [16], Haystack [18], Magnet [23], DBin [25], semantExplorer [21], and others. These systems allow the browsing of Semantic Web data and provide various ways of searching and visualizing this data. Many of them combine semantic data with "classical" Web content or other human-oriented content in order to provide the user a comprehensive access to information.

More recently, the idea of a "Semantic Desktop" has emerged [6, 19, 20]; these systems largely focus on personal information management, and demonstrate the utility of exposing "legacy" data in Semantic Web formalisms, often involving transformations from both file-based data as well as databases. The large body of work on mapping data between relational databases and Semantic Web formalisms is described in [1].

Yet another twist on the browsing theme is Piggy Bank [10]. It allows semantic data to be *collected* while browsing "ordinary" Web pages. This data can originate either in some Semantic Web format, or it can be "scraped" or transformed from a number of known page formats.

2. Browsing RDF Data

Since the pervasive mainstream adoption of the World Wide Web, browsing has become a natural user interface paradigm. In order to enable users to browse RDF graphs we have constructed a simple tool called OINK ("Open Integration of Networked Knowledge"). It provides a *node-oriented view* of graphs, visualizing each node as a Web page. These pages show the various ways of identifying the

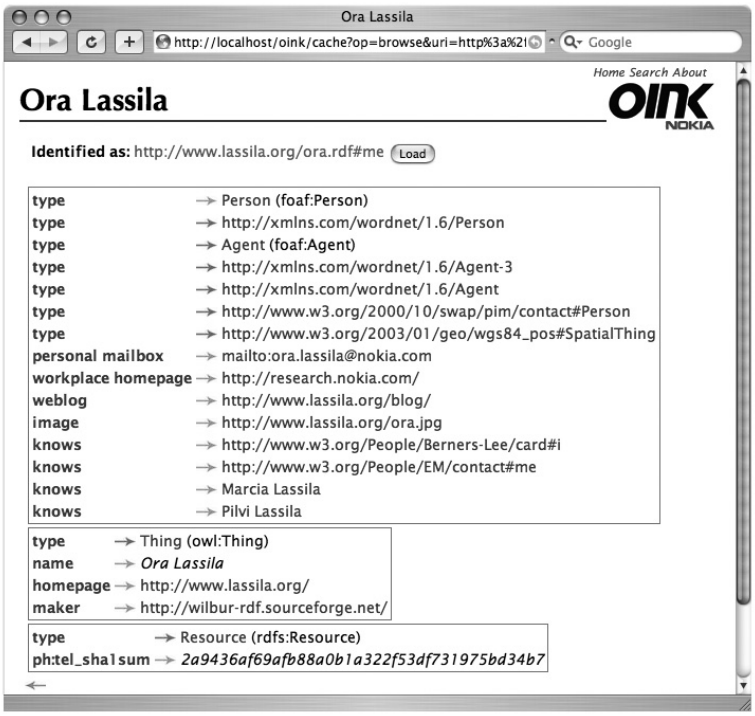

Figure 1. A typical page from OINK

node in question, as well as a list of properties and their values. Figure 1 shows a typical page from OINK, as rendered in a Web browser. OINK allows the simultaneous viewing and exploration of any number of RDF graphs, thus supporting the use of RDF in information integration (technically, all documents loaded into OINK form a *single* graph; OINK's reasoner supports not only RDF(S) but also *inverse functional properties* – allowing, say, automatic integration of FoaF [5] profiles).

In addition to visualizing the *outbound* edges, OINK shows the *inbound* edges as well. This allows users to navigate the edges of a graph in the reverse direction (a typical example of this would be when one wants to navigate from an RDF class to any of its subclasses). Since OINK is based on the RDF metamodel – where everything has a representation in RDF itself – all items on an OINK page may be navigated to (this includes the definitions of the RDF properties *and* their values). The simple metamodel translates directly to a clear and understandable user interface model.

We originally envisioned OINK as a lightweight tool primarily for "debugging" RDF data, but subsequent use of the system has encouraged its development as a platform for building customized browsing solutions for complex data. The use of the underlying reasoning engine also allows automatic integration of data from multiple sources, further reinforcing the idea that Semantic Web technologies can be used for "spontaneous" end-user tasks.

3. Architecture and Implementation

The overall architecture of the OINK system is illustrated in Figure 2. In broad strokes, OINK is built around a storage and query engine for RDF graphs. An HTTP *servlet* queries the data in the RDF store, and renders it as XHTML. The current prototype implementation of OINK is written in Common Lisp [24] on top of the WILBUR Semantic Web toolkit [12, 13], and uses the Portable AllegroServe web application server [22, chapter 26].

The main-memory RDF database (= "triple store"), acting as a cache, is the central component of OINK.[1] The user registers *data sources* that are loaded into the cache; various facilities are available to the user to maintain this cache:

- Data sources can be "refreshed" manually. OINK relies on the underlying WILBUR mechanism where "old" data is replaced by freshly parsed "new" data as an atomic transaction of the database. Every RDF statement in OINK is linked to the data source(s) that asserted it. Sources are treated as nodes, and any statements about these show up in OINK as well.

- Data sources can also be *automatically* refreshed using two different methods. The user or the document

[1] So as not to confuse the reader, it should be pointed out that this is a lower-level caching mechanism than the "Semantic cache" we have reported on earlier [11].

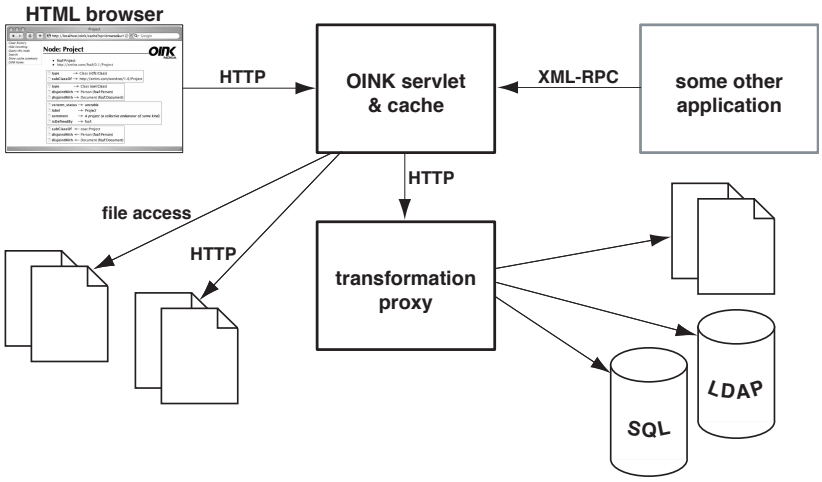

Figure 2. The overall architecture of OINK

itself can designate an interval after which the data in the cache is considered "old"; refreshing can also rely on the HTTP caching semantics, based on the HTTP headers from the response when the data was loaded.

The OINK cache merely remembers *where* the data comes from, so reinitializing the cache causes all the data sources to be loaded anew from their original locations.

The cache can be searched for substring matches on literals or queried using WILBURQL path queries [13]. Current work on OINK involves a mechanism where queries are generated automatically from the user's paths through the graph; these queries, in turn, can be used to find similarly related items [14].

Based on the WILBUR triple store, the OINK cache supports RDF(S) entailments [9] via on-demand generation of the *deductive closure* of the graph stored in the cache. Reasoning is based on a technique where access queries to the triple store are rewritten so that they reflect the "virtual" deductive closure [13]. In addition to RDF(S), the reasoner supports integration-related features such as owl:sameAs and *inverse functional properties* [17].

The OINK servlet merely responds to HTTP requests by querying the cache and producing pages – one per each node in the graph, on demand – that reflect the graph structure. Essentially, the generation of a page for an RDF resource (node) is divided into three components:

1. The node is identified: if present, the value of rdfs:label property (or some subproperty thereof) is used. A link is also provided to the Web resource itself: this is useful, for example, if the node described is a real Web page or some other document that can be rendered in the Web browser.

2. The properties are visualized. The properties of the node are split into clusters corresponding to the most specific non-overlapping classes of the node, using associated domain specifications. Any "leftover" properties are assumed to be associated with rdfs:Resource and are visualized last. The classes that are considered include both asserted and inferred classes of the node. Finally, inbound arcs are visualized separately.

   Each cluster of properties can be associated with a special markup generator that can emit class-specific visualizations. The markup generator associated with rdfs:Resource is assumed to be the default. We are currently implementing a subset of the Fresnel vocabulary [3] for customized presentations.

3. OINK tracks the path the user has taken to navigate to any specific node; this history is shown and allows the user to invoke queries generated automatically from the navigational path (Figure 4).

OINK supports multi-byte characters, and tracks the xml:lang attribute of literals. Figure 3 shows an example of Japanese text with the appropriate language attribute.

Any statements that have been *reified* [15, section 4] have an indicator next to them (as if a footnote or endnote had been associated with them). Clicking the indicator takes one to the page representing the reified statement.

Finally, the servlet will be able to generate structurally simplified pages when it detects that the requesting client is a limited, small device, such as a mobile phone.[2] Browsing semantic data on a low-bandwidth, small-screen device is appealing, since the information is in a more compact and possibly more succinct form.

In order to be able to load various "legacy" data into OINK, we use a transformation engine that allows data format transformations to be written in XSLT or as CGIs. In

[2] The current prototype only knows about Nokia S60 phones.

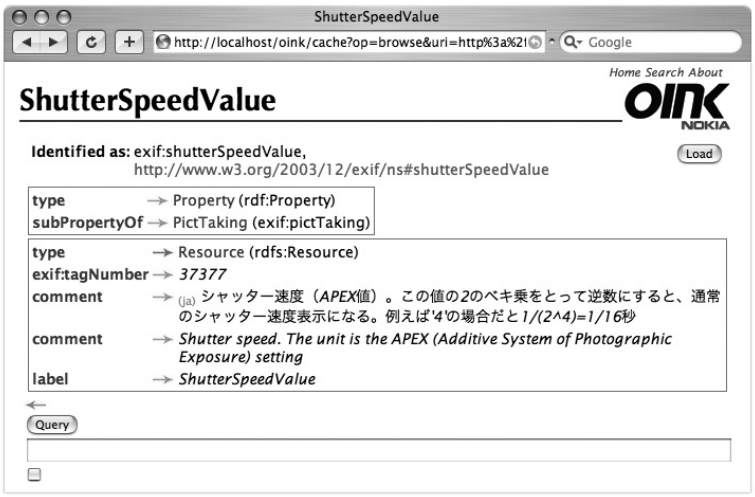

Figure 3. Showing support for non-Western text and xml:lang

a loose sense the engine takes the form of an HTTP *proxy*. OINK makes HTTP requests to the proxy, designating not only the original data source but also the desired transformation. The proxy loads the data, applies the transformation, and hands the resulting RDF to OINK. The purpose of the transformation engine, from the architectural viewpoint, is to separate data format conversions from the rest of the system. OINK is a "pure" RDF application in the sense that it does not know anything about other data formats.

4. Conclusions

In a metaphorical sense, OINK brings the Semantic Web "closer" to the user. Various tools can be applied to the data in the cache, without concern of the "outside world" (e.g., one could locally close the world with respect to reasoning). Generally, the cache separates issues of data access and cache management from higher-level considerations such as reasoning and other "ontological" mechanisms.

OINK has proven useful in visualizing RDF data. Despite the typical Semantic Web focus on machine interpretation of information, there are benefits to letting users explore semantic representations. These benefits extend beyond tasks like debugging ontologies; with suitable visualization, data integrated from multiple sources can be used for building new applications.

OINK was originally created as an easy-to-use debugging tool for RDF data; subsequent development towards customized visualizations suggests that it could be seen as an application platform. There is a large class of applications that are *little more* than queries to Semantic Web data, associated with suitable visualization.

References

[1] D. Beckett and J. Grant. Mapping Semantic Web Data with RDBMSes. SWAD-Europe Deliverable 10.2, World Wide Web Consortium, 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[3] C. Bizer, R. Lee, and E. Pietriga. Fresnel – A Browser-Independent Presentation Vocabulary for RDF. In *End User Semantic Web Interaction Workshop at the 4th International Semantic Web Conference*, Galway, Ireland, 2005.

[4] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium, Jan. 2003.

[5] D. Brickley and L. Miller. FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/, Sept. 2004.

[6] S. Decker and M. Frank. The Social Semantic Desktop. Technical Report DERI-TR-2004-05-02, DERI, 2004.

[7] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 436–445, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[8] R. Goldman and J. Widom. Interactive Query and Search in Semistructured Databases. In *WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases*, pages 52–62, London, UK, 1999. Springer-Verlag.

[9] P. Hayes. RDF Semantics. W3C Recommendation, World Wide Web Consortium, Feb. 2004.

[10] D. Hyunh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the Semantic Web Inside Your Web Browser. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005, 4th International Semantic Web Conference*. Springer-Verlag, 2005.

[11] D. Khushraj and O. Lassila. Ontological Approach to Generating Personalized User Interfaces for Web Services. In

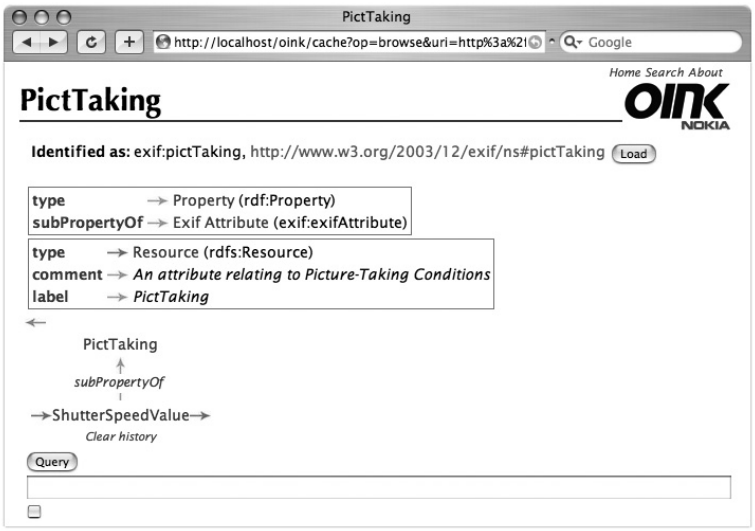

Figure 4. Visualization of navigation history and automatically generated queries

Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005, 4th International Semantic Web Conference*, number 3729 in Lecture Notes in Computer Science, pages 916–927, Galway, Ireland, Nov. 2005. Springer-Verlag.

[12] O. Lassila. Enabling Semantic Web Programming by Integrating RDF and Common Lisp. In *Proceedings of the First Semantic Web Working Symposium*. Stanford University, 2001.

[13] O. Lassila. Taking the RDF Model Theory Out for a Spin. In I. Horrocks and J. Hendler, editors, *The Semantic Web - ISWC 2002, 1st International Semantic Web Conference*, volume 2342 of *Lecture Notes in Computer Science*, pages 307–317. Springer-Verlag, 2002.

[14] O. Lassila. Generating Rewrite Rules by Browsing RDF Data. to appear, 2006.

[15] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium, Feb. 1999.

[16] monica schraefel, M. Karam, and S. Zhao. mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia. In *AH2003: Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2003.

[17] P. F. Patel-Schneider, P. J. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, World Wide Web Consortium, Cambridge, MA, Feb. 2004.

[18] D. Quan and D. Karger. How to Make a Semantic Web Browser. In *Proceedings of the 13th International Conference on the World Wide Web*, pages 255–265. ACM Press, 2004.

[19] L. Sauermann. The Gnowsis Semantic Desktop for Information Integration. In *1st Workshop on Intelligent Office Appliances*, 2005.

[20] L. Sauermann and S. Schwartz. Gnowsis Adapter Framework: Treating Structured Data as Virtual RDF Graphs. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005, 4th International Semantic Web Conference*, number 3729 in Lecture Notes in Computer Science, pages 1016–1028. Springer-Verlag, Nov. 2005.

[21] S. Scerri, C. Abela, and M. Montebello. semantExplorer: A Semantic Web Browser. In P. Isaías and M. B. Nunes, editors, *IADIS International Conference WWW/Internet 2005*, pages 35–42, October 2005.

[22] P. Seibel. *Practical Common Lisp*. APress, Berkeley, CA, 2005.

[23] V. Sinha and D. R. Karger. Magnet: Supporting Navigation in Semistructured Data Environments. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 97–106, New York, NY, USA, 2005. ACM Press.

[24] G. L. Steele. *Common Lisp the Language, 2nd edition*. Digital Press, 1990.

[25] G. Tummarello, C. Morbidoni, P. Puliti, and F. Piazza. The DBin Semantic Web Platform: An Overview. In *Workshop on the Semantic Computing Initiative (SeC 2005)*, Chiba, Japan, May 2005.

[26] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted Metadata for Image Search and Browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM Press.

Acknowledgements

Work described in this paper was funded in part by Nokia Technology Platforms. The author would like to thank the following people for their help and support: Sadhna Ahuja, Jan Bosch, Barbara Heikkinen, Deepali Khushraj, Marcia Lassila, Heikki Saikkonen, and Marko Suoknuuti.