Ontology-based Information Presentation

Michal Štípek and Vladimír Grlický

Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
michal.stipek@gmail.com

Abstract. The aim of the framework (codenamed "Prescott") presented in this paper is to serve for a presentation of ontology-based information to end users. The presentation is realized by application of restricting and formatting criteria that are defined by the Fresnel vocabulary. Using the Fresnel vocabulary it is possible to reduce total amount of data presented to end user, and to present this data in a desired form. The main concept of the Fresnel vocabulary comprises using lenses and formats to create the final presentation. The Prescott enables to select appropriate instances with desired properties in the correct order according to applied lenses and it also allows applying formatting criteria to instances and properties according to defined formats. Unlike original Fresnel vocabulary, the framework supports additional functionality like sorting and creation of subsets of instances. The output of the framework is an XML document, which can be further transformed into another presentation format like XHTML, PDF, or SVG.

1 Introduction

Semantic Web browsers and other tools aimed at displaying RDF data to end users are all concerned with the same problem: presenting content primarily intended for machine consumption in a human-readable way. The problem is addressed by these types of applications using different representation paradigms. Tools like IsaViz [6] and Welkin[1] represent RDF models as node-link diagrams, explicitly showing their graph structure. Other tools use nested box layouts (Longwell[2]) or table-like layouts (Brownsauce [5]) for displaying properties of RDF resources with varying levels of details. A third approach combines these paradigms and extends them with specialized user interface widgets designed for specific information items like calendar data or tree structures, providing advanced navigation tools and other interaction capabilities (Haystack [4]). Their solutions differ but in the end they address the same two high-level issues, no matter the underlying representation paradigm: specifying *what* information contained in RDF models should be presented (content selection) and *how* this information should be presented (content formatting and styling). However, each

[1] SIMILE: Welkin, http://simile.mit.edu/welkin/
[2] SIMILE: Longwell RDF Browser, http://simile.mit.edu/longwell/

tool currently relies on its own ad hoc mechanisms and vocabulary for specifying RDF presentation knowledge, making it difficult to share and reuse such knowledge across applications. Recognizing the general need for presenting RDF content to users and wanting to promote the exchange of presentation knowledge, our Cocoon-based presentation framework Prescott is extended by the Fresnel ontology support that is a browser-independent extensible vocabulary of core RDF display concepts.

2 Main Features

One of the main Prescott's features is a transformation of original RDF graph to XML document, which can be additionally transformed to required output format like XHTML, PDF, or SVG. The transformation is realized using user-defined transformation rules that are specified within terms of the Fresnel vocabulary. The Fresnel vocabulary is a simple browser independent ontology that can be used to define the way how RDF graph should be displayed [1]. The vocabulary is based on two main concepts:

- *lenses*, which define sources and their properties that should be displayed,
- *formats*, which define visual style of sources to display together with their properties.

The tool implements the most of the functionality defined by the core vocabulary for lenses and formats and adds some new features. This functionality includes:

- selection of particular instances and their properties in particular order,
- displaying so-called referenced instances, i.e. instances, whose properties display properties of other instances,
- assignment of cascading style classes and additional content to particular instances, properties, property labels, and property values,
- affecting the way how property values are displayed.

Added functionality includes:

- sorting instances according to particular property values,
- instances pagination, i.e. selecting particular number of instances using exact offset and limit values.

Inputs of the transformation process comprise definition of ontology used, instances, and defined lenses and formats. The output is an XML document, which can be further transformed to relevant output format.

3 RDF Manipulation

For manipulation and storing RDF data the Prescott uses the open source framework Sesame. By using this framework we can focus on transformation process and do not need to concern with RDF data manipulation.

The Prescott uses four main functional data repositories that are used for storing and manipulation ontology definition, instances, the Fresnel lenses and formats, and selected instances and properties, which are enriched with formatting criteria.

4 Transformation Process

The process of transformation of RDF graph to XML consists of three main steps:

1. Selection of instances and properties according to defined lenses. The output of this step is ordered tree of RDF nodes.
2. Assignment of formatting criteria to RDF nodes in ordered tree where the assignment of formatting criteria is realized according to defined formats.
3. Output creation. Result is an XML document that is usually additionally transformed into desired output format.

4.1 Instance Selection

First step of instance selection process is creation of internal representation of all lenses and formats stored in the RDF repository. Use of internal representation is necessary for achieving higher effectiveness during instance selection. After lenses extraction, all lenses are selected step by step in the order that ensures avoiding conflicts between lenses. Conflict between lenses is a situation, when more than one lens is applicable to an instance. After lens selection step, all instances defined by lens domain are selected and already processed instances are removed. Then, if necessary, instances are sorted and according to the number of instances to display and defined offset, particular subset of instances is created. All these instances are then inserted into the repository with selected instances, and particular properties, which are defined in appropriate lens, are added to each instance. If selected repository contains desired instance, the value of original property is set to reference to this instance. Otherwise the new instance in selected repository is created. If property does not contain nested lens, concrete value is inserted to this property. Creation of nested instances is constrained by maximum recursion depth, which is defined directly in a lens.

Conflicts between lenses. Conflicts solving process defined in original specification uses approach where all instances from the source are checked step by step, and to each instance the adequate lens is used. In the case when more than one lens is applicable to instance, the most specific lens is used. The specificity of lens is determined by their domain. It means that instance domain is more specific than class domain, and instance domain defined with FSL selector is more specific than instance domain defined with simple selector. In the case of class domains the specificity is determined according to depth of classes in class hierarchy, and in the case of FSL domains the specificity is determined according

to specificity of FSL selector. Finally, if there is still more than one lens applicable to instance after all of these comparisons, the lens with default purpose is used. In the transformation process we used opposite approach. It means that instead of sequential passing of instances and finding relevant lens, we gradually process lenses according to specificity of their domain. Next, we find appropriate instances to each lens, while already processed instances are bypassed. Also only lenses that are not used as sublenses in other lenses and lenses that do not have label purpose defined are used. The overall lens selection process involves these steps:

1. Select lenses with default purpose, and that have an instance domain defined with FSL selector.
2. Select other lenses that have an instance domain defined with FSL selector.
3. Select lenses with default purpose, and that have an instance domain defined with simple selector.
4. Select other lenses that have an instance domain defined with simple selector.
5. Select lenses with default purpose and having a class domain. Lenses are ordered by the value of class depth in class hierarchy, starting with the deepest.
6. Select other lenses that have a class domain. Lenses are ordered by the value of class depth in class hierarchy, starting with the deepest.

Repository with selected instances. During the design of the repository with selected instances, it was necessary to create the structure of the repository, which would ensure that every occurrence of the same instance and property can be represented independently. The reason is that one instance or property can be used more times and each occurrence can be formatted using other formatting criteria. The solution is to represent every occurrence of instance or property as RDF blank node. When using blank nodes, we are able to add some metadata to each instance and property, which can be used for ordering these entities, and also for adding some formatting criteria.

Sorting. Sorting instances according to simple property that does not contain sublens is relatively easy. On the other hand, if instances should be sorted according to properties that contain sublenses, it is necessary to find out values of properties in referenced instances, and to use sorting parameter defined in nested lens. The problem is that sorting property in referenced instance can also reference other instance, etc. The solution is to iterate through all nested instances and create the set of pairs containing original top level instance and sorting value which originate from iteration process through all nested instances. The pairs are then sorted using these sorting values, while numeric values are sorted before string values.

4.2 Assignment of Formatting Criteria

Assignment of formatting criteria is similar process to instance selection. At first, all formatting criteria defined by formats, which were defined in lenses

used for instance selection, are assigned to occurrences of instances and properties. Then, all formats are selected step by step in order, which ensures avoiding conflicts between formats. Resolving conflicts between formats is similar to resolving conflicts between lenses. Finally, all formatting criteria defined in the format are assigned to particular occurrence of an instance or a property.

Conflicts between formats. When resolving conflicts between formats, the same approach as the one used in resolving conflicts between lenses is applied. It means that formats are selected according to the specificity of their domain gradually, and already processed occurrences of instances and properties are bypassed. The overall formats selection process involves these steps:

1. Assign all formatting criteria defined by groups and formats that were exactly assigned in lenses during instance selection to all particular occurrences of instances and properties.
2. Select formats that have an instance domain defined with FSL selector.
3. Select formats that have an instance domain defined with simple selector.
4. Select formats that have a class domain. Formats are ordered by the value of class depth in class hierarchy, starting with the deepest.
5. Select formats that have a property domain. Formats are ordered by the specificity of the property domain. It means that the first ones are formats with the property domain defined with FSL selector, and then with simple selector.

When conflicts can not be resolved, undetermined format will be used first.

Formatting criteria assignment. All formatting criteria are assigned to blank nodes representing occurrences of instances and properties. All style classes and additional content for the whole resource are assigned to a blank node representing occurrence of instance. On the other hand, all styles, additional content, and displaying style of the whole property, property label, and property values are assigned to a blank node representing an occurrence of a property.

4.3 Output Generation

Due to the designed structure of the repository, generating of output XML document is relatively easy. The only necessary thing is to find all instances on the top level, and then iterate through all subgraphs of these instances and generate particular XML elements. This is possible because all properties do not contain only values, but also can reference other occurrences of nested instances.

Output XML format originates from XML output of native Fresnel implementation from the SIMILE project[3]. The only difference is in elements representing additional contents.

It is possible to transform XML document using XSLT stylesheet to other XML document, e.g. XHTML document, which can be used for displaying instances and their properties in some ontology browser.

[3] SIMILE Project, `http://simile.mit.edu`

5 Experiments

Functionality of the tool was verified using test sets of data that were used to generate output XHTML documents. As a test data there were used the FOAF[4] and the JobOffer ontologies. The latter has been developed within the scope of the NAZOU project [3]. In the context of testing, two main groups of lenses were also created. The first group of lenses is used to show general overview of all instances in ontology. These instances are showed only with few the most relevant properties. This overview can be used to select one concrete instance. The second group of lenses is used to view only one instance, but in higher detail, showing almost all properties including properties in referenced instances.

To generate appropriate XHTML output, it was also necessary to create XSLT stylesheet file, which was used for XML transformation, and also CSS stylesheet documents, which was used to adjust final appearance of instances and their properties.

As there is also the original implementation of the Fresnel transformation from the SIMILE project, it can be useful to compare the old and the new implementation.

As a test data we used instances from the JobOffer ontology stored in external file that contained about 7300 statements and the file size was about 1MB. The results of the test can be viewed in fig. 1.

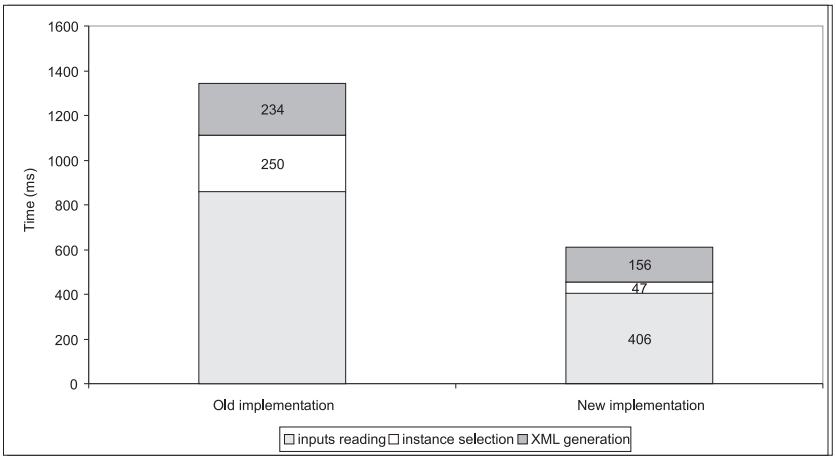

Fig. 1. Test results for the JobOffer ontology

The new implementation is faster than the old one approximately by 33 %, where the input reading is faster by 17 %; the instance selection is faster by 26 %, and the XML generation is faster by 55 %.

[4] The Friend of a Friend (FOAF) Project, `http://www.foaf-project.org`

The next test (fig. 2) was performed on a smaller set of instances from the FOAF ontology, which contained only 60 statements.

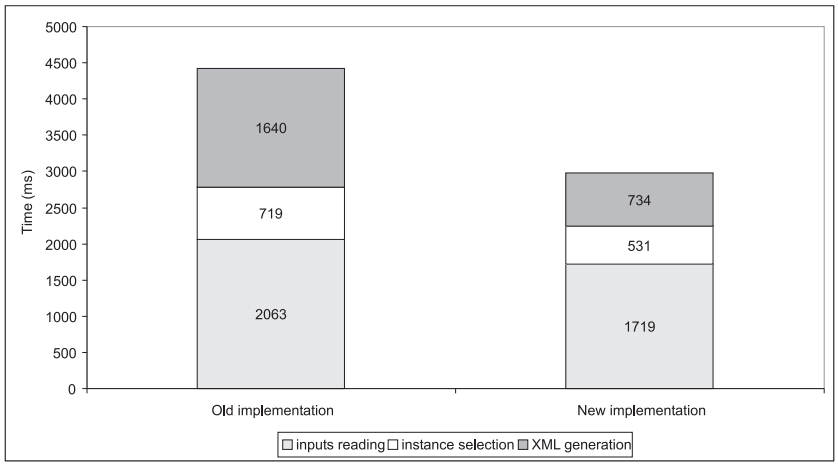

Fig. 2. Test results for the FOAF ontology

The new implementation is again faster than the old one approximately by 55 %, where the input reading is faster by 47 %; the instance selection is faster by 81 %, and the XML generation by 33 %.

As we can see from both tests, when using smaller set of data, speedup is higher than in tests, where using larger set of instances. The difference in speeds is increasing in XML generation and decreasing in instance selection. It is hard to say whether the main reason for speedup in XML generation reposes in library used for XML generation or in the structure of selected repository and used algorithms. If we look at instance selection process, we are still unable to say, if this process is faster, until we will test both implementations on much larger set of data.

6 Conclusions

In this paper we have described the design of ontology-based information presentation tool, with emphasis on its RDF-to-XML transformation part based on the Fresnel vocabulary. The Prescott is a universal tool for creating web applications with the aim of displaying information resources specified by metadata, which is based on architectural style "filters and pipes". The main advantage of this solution is ability to display any domain ontology (RDFS or OWL based one) along with displaying "classical" XML data. Another plus for the proposed tool is reuse of already created transformation filters and its high flexibility because of built-in transformation filter chaining mechanism. One of possible usages of

the proposed framework is a universal faceted browser, which allows selecting data from a domain ontology using a set of restrictions. An example of a faceted browser is project SWED[5] or already mentioned project Longwell. The second one is also the implementation based on the Fresnel ontology. Among other implementations that are built on this ontology there also belongs the system Arago [2].

Acknowledgements

This work was partially supported by the Slovak State Programme of Research and Development "Establishing of Information Society" under the contract No. 1025/04.

References

1. Bizer, Ch., Lee, R., Pietriga, E. (eds.): Fresnel – Display Vocabulary for RDF: User Manual [online]. 2005 [cited 2006-04-04] `http://www.w3.org/2005/04/fresnel-info/manual/`
2. Gassert, H., Harth, A.: From Graph to GUI: Displaying RDF Data from the Web with Arago. Workshop on Scripting for the Semantic Web, Colocated with ESWC 2005, Heraklion, Greece, 2005
3. Návrat, P., Bieliková, M., Rozinajová, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: CompSysTech 2005, B. Rachev, A. Smrikarov (eds.), Varna, Bulgaria. pp. IIIB.7.1–IIIB.7.6, 2005
4. Quan, D., Karger, D.: How to Make a Semantic Web Browser. In: Proceedings of the 13th international conference on World Wide Web, 2004. pp. 255-265
5. Steer, D.: BrownSauce: An RDF Browser. XML.com [online]. 2003 [cited 2006-07-14] `http://www.xml.com/pub/a/2003/02/05/brownsauce.html`
6. W3C: IsaViz: A Visual Authoring Tool for RDF [online]. 2001 [cited 2006-07-14] `http://www.w3.org/2001/11/IsaViz/`

[5] Semantic Web Enviromental Directory, `http://www.swed.org.uk`