XSLT+SPARQL: Scripting the Semantic Web with SPARQL embedded into XSLT stylesheets

Diego Berrueta[1], Jose E. Labra[2], and Ivan Herman[3]

[1] Fundación CTIC
Gijón, Spain
diego.berrueta@fundacionctic.org
[2] Departamento de Informática
Universidad de Oviedo, Spain
labra@uniovi.es
[3] Centre for Mathematics and Computer Sciences (CWI)
Amsterdam, the Netherlands
Ivan.Herman@cwi.nl

Abstract. Scripting the Semantic Web requires to access and transform RDF data. We present XSLT+SPARQL, a set of extension functions for XSLT which allow stylesheets to directly access RDF data, independently of any serialization syntax, by means of SPARQL queries. Using these functions, XSLT stylesheets can retrieve, query, merge and transform data from the semantic web. We illustrate the functionality of our proposal with an example script which creates XHTML pages from the contents of DBpedia.

1 Introduction

The semantic web has adopted RDF as its preferred data model for resource description. However, the document web is based on markup languages like XML and HTML. Transforming data and gluing between the two worlds with the current set of tools is not always easy. In particular, XSLT [16] and the newer XQuery [10] have been around since 1999 and 2007 respectively, and they offer excellent functionality to query and transform XML documents into other formats. Although they do not directly produce RDF data, indeed they can output RDF serialized in RDF/XML or N3.

However, the inverse transformation, from RDF to XML, is insufficiently covered by the current tool set. Most popular web scripting languages, such as Python and PHP, have their own non-standard APIs for accessing RDF data (conversely, there are standardized APIs for XML, such as DOM and SAX). This is an hindrance to the use of these scripting languages with RDF.

Unfortunately, standard languages such as XSLT and XQuery are inappropriate for this task. Even if RDF data can be apparently addressed as XML through its RDF/XML serialization syntax [5], a second analysis reveals that this is only possible in a very controlled environment in which the serialization is specially produced under certain constraints, or normalized in a previous step [2].

We propose XSLT+SPARQL, an extension to the XSLT function set to embed SPARQL SELECT and ASK queries in the XPath selection expressions. This makes it possible to process RDF data directly within the RDF model, i.e., regardless of any particular serialization syntax. We believe XSLT+SPARQL can contribute to the semantic web by providing a new platform for scripting and transforming RDF into XML, which is often the last processing step of many semantic web scripts. XSLT+SPARQL can easily produce XHTML reports, SVG graphs, SOAP messages or, more generally, any valid XSLT output format (XML or text documents). Moreover, developers can exploit the expressivity of XSLT to create complex scripts that combine, filter and transform data retrieved from different web sources or even from remote SPARQL endpoints.

The rest of the paper is organized as follows. Next section examines related work prior to the description of XSLT+SPARQL in Section 3. Implementation is briefly discussed in Section 4, and a use case is presented in Section 5. The discussion in Section 6 concludes the paper.

2 Related work

After a number of proposals from different groups [15], W3C finally came with SPARQL [17], a flexible query language for RDF. There are two companion specifications: the query protocol [13] and a XML Schema to represent the results [6]. Many have observed that the latter makes SPARQL results tractable with XSLT stylesheets in order to generate XML (XHTML, RSS...) from RDF data. This approach is indeed useful for simple tasks, where a single SPARQL query is enough to extract all the relevant data from an RDF graph, but it falls short for complex transformations. Nevertheless, this technique has been suggested as a mechanism to implement the "lowering" operation of semantic web services grounding (cf. section 4.2 of [3]).

The most direct precedent to our work is [19]. The authors of the Topia project introduce a multi-stage processing architecture that consumes RDF and XML data with XSLT stylesheets. They use XSLT extension functions to query a Sesame RDF repository [11] using RDQL [18], RQL and SeRQL, three RDF query languages which preceded in time to SPARQL. In this sense, our work is an updated version of theirs, but in addition to the change of the language, there are other differences. Our queries return binding tables (for SELECT queries) or just Boolean results (for ASK queries), while Topia queries return sub-graphs (sets of triples) serialized in XML.

More recently, XSPARQL [2] has been proposed to unify SPARQL and XQuery in a single language that extends both of them. XQuery and XSLT are W3C recommendations with an important overlap in their functionality. XSPARQL is built on top of the XQuery syntax and semantics. On the other hand, XSLT+SPARQL retains the XSLT syntax and processing model and exploits the extensibility of the XPath function set. The two proposals transform between RDF and XML in both directions, and both can read and produce RDF serialized in different syntaxes (Turtle or RDF/XML).

3 Description of XSLT+SPARQL

XSLT+SPARQL exploits the extensibility mechanism provided by the host language, to introduce a number of extension functions. They are grouped in two different namespaces.

3.1 Basic query functions

The core functionality of XSLT+SPARQL is provided by just two functions:

1. `sparql:sparql(query [, documentUrl, ...])`
2. `sparql:sparqlEndpoint(query, endpointUrl)`

These functions execute a SELECT or ASK query (CONSTRUCT and DESCRIBE queries are not allowed because they return RDF/XML documents, which are notoriously difficult to handle in XSLT). The first function executes the query locally, while the second one sends a request to a remote SPARQL endpoint and retrieves the results. The documents pointed by the (potentially empty) list of URLs are merged with the ones referred by the "FROM" clauses of the query to create the default graph.

3.2 Advanced functions

Some applications may need to efficiently execute multiple queries against the same dataset, or even to build custom datasets by merging multiple graphs. To address the requirements of these applications, a second set of extension functions is defined in a different namespace:

1. `sparqlModel:parseString(serializedRdf [, serializationSyntax])`
2. `sparqlModel:readModel(documentUrl [, serializationSyntax])`
3. `sparqlModel:readModel(nodeset)`
4. `sparqlModel:mergeModels(firstModel, secondModel, ...)`
5. `sparqlModel:sparqlModel(query, model)`

The first three functions read RDF data from different sources: (1) a string; (2) a document pointed by a URL; or (3) any fragment of the XSLT input tree, the result tree –only in XSLT 2.0– or even the XSLT stylesheet itself. When the input is a string or a web document, an optional parameter selects the RDF serialization syntax to be parsed (N3 [7], Turtle, RDF/XML, GRDDL [14], RDFa [1]). These three functions return a handler for an in-memory local RDF model that can be used in the next functions.

The "mergeModels" function combines any number of input models into a new one. Due to the functional character of XSLT, it is not possible to do "in-place" modifications of a model, therefore this function returns a new model with the result of combining the data. Consequently, scripts that require to merge data from a sequence of sources cannot iterate through the sequence, but have to

be re-casted as recursive templates using well-known patterns from functional programming.

Finally, the fifth function executes a SPARQL query over an in-memory model. Any "FROM" or "FROM NAMED" clause in the query is ignored.

Together, these advanced functions bring great flexibility to the developer, and also a performance boost to execute multiple queries against the same dataset, avoiding the need for repeated parsing of the input documents.

4 Implementation

We have implemented the above functions in Java as an extension for Apache Xalan[4], an open-source XSLT processor, although it should be easy to adapt our code to other XSLT processors. Our implementation uses the Jena framework [12] to manage RDF documents, and to execute queries locally and remotely. We note that a pure XSLT implementation of "sparql:sparqlEndpoint" is also possible using the "document" function of XSLT. However, due to the lack of control on how that function performs HTTP content negotiation, the pure XSLT implementation may be unable to query some endpoints –notably, DBpedia's endpoint– when certain XSLT processors are used. On the other hand, our Java-based implementation can manage content negotiation to retrieve RDF or XML documents as needed, therefore it can access the Linked Data web [8,9].

A limitation of our current implementation concerns the "sparql:sparql" and "sparqlModel:mergeModels" functions, which do not accept an arbitrary number of arguments, due to the inability of Xalan 2.7 to access Java5 vararg methods using reflectivity. Particularly, the former accepts zero or one document URLs, and the latter merges exactly two models.

5 Use case: DBpedia to XHTML

Practical use of XSLT+SPARQL is demonstrated by a sample script (Figure 1) that extracts data from DBpedia [4] and creates a simple report in XHTML. The stylesheets contains only two templates. The first one efficiently accesses DBpedia data through its public SPARQL endpoint (i.e, the script does not retrieve the RDF dataset, but just the results of the query in XML format). The second one generates XHTML mark-up for each row of the results bindings table.

The functionality of XSLT+SPARQL goes far beyond this small example. More complex examples can be found in the project web page[5].

6 Conclusions and future work

We presented XSLT+SPARQL, a simple extension for XSLT to embed SPARQL queries in XPath expressions, making it possible to process RDF data from

[4] `http://xml.apache.org/xalan-j/`

[5] `http://berrueta.net/research/xsltsparql`

```
<xsl:variable name="query">
  PREFIX skos: &lt;http://www.w3.org/2004/02/skos/core#&gt;
  PREFIX cat:  &lt;http://dbpedia.org/resource/Category:&gt;
  PREFIX rdfs: &lt;http://www.w3.org/2000/01/rdf-schema#&gt;
  PREFIX foaf: &lt;http://xmlns.com/foaf/0.1/&gt;
  SELECT ?name ?person ?img
  WHERE { ?person skos:subject   cat:Spanish_actors .
          ?person rdfs:label     ?name .
          ?person foaf:depiction ?img .
          FILTER (lang(?name)="es") }
  ORDER BY ?name
</xsl:variable>

<xsl:template match="/">
  <html> <body>
      <h1>Spanish Actors</h1>
      <ul>
        <xsl:apply-templates select="sparql:sparqlEndpoint(
           $query, 'http://dbpedia.org/sparql')"/>
      </ul>
  </body> </html>
</xsl:template>

<xsl:template match="results:result">
  <li>
    <xsl:value-of select="results:binding[@name='name']"/>
    <img> <xsl:attribute name="src">
        <xsl:value-of select="results:binding[@name='img']"/>
    </xsl:attribute> </img>
  </li>
</xsl:template>
```

Fig. 1. An XSLT+SPARQL stylesheet to create a XHTML page with data extracted from DBpedia by querying its SPARQL endpoint.

XSLT stylesheets without bothering with the tricky RDF/XML serialization. XSLT+SPARQL is a new standards-based platform to write declarative scripts for the semantic web. It is straightforward to use for developers experienced in XML and RDF technologies because it does not require to learn a new language or processing model. Furthermore, it can be implemented by re-using current XSLT processors.

The usage of XSLT+SPARQL has been illustrated by means of a sample application to create XHTML reports from DBpedia data. We envisage the usage of XSLT+SPARQL in a wide variety of scenarios. They mainly involve transforming RDF data into other formats which are more suitable for human consumption. However, more complex semantic web agents can be developed as well. For instance, we are working on a semantic web crawler/browser that

collects relevant information about a subject from different linked data sources and creates a comprehensive report in XHTML.

We are also considering new additions to the current set of functions. One area of particular interest would be to add support for RDFS and OWL reasoning.

References

1. B. Adida and M. Birbeck. RDFa primer 1.0. Working draft, W3C, March 2007.
2. W. Akhtar, J. Kopecky, T. Krennwallner, and A. Polleres. XSPARQL: Traveling between the XML and RDF worlds and avoiding the XSLT pilgrimage. Technical Report 2007-12-14, DERI Galway, 2007.
3. R. Akkiraju and B. Sapkota. Semantic annotations for WSDL and XML schema - usage guide. Working group note, W3C, 2007.
4. S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference*, 2007.
5. D. Beckett. RDF/XML syntax specification (revised). Recommendation, W3C, February 2004.
6. D. Beckett and J. Broekstra. SPARQL query results XML format. Recommendation, W3C, January 2008.
7. T. Berners-Lee. Notation 3. Available at `http://www.w3.org/DesignIssues/Notation3.html`, 1998.
8. D. Berrueta and J. Phipps. Best practice recipes for publishing RDF vocabularies. Working draft, W3C, 2007.
9. C. Bizer, R. Cyganiak, and T. Heath. How to publish linked data on the web?, Jul 2007.
10. S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language. Recommendation, W3C, 2007.
11. J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF. In *International Semantic Web Conference*, 2002.
12. J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *13th international World Wide Web conference Alternate track papers*, 2004.
13. K. G. Clark. SPARQL protocol for RDF. Recommendation, W3C, January 2008.
14. D. Connolly. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). Recommendation, W3C, September 2007.
15. P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of RDF query languages. In *Third International Semantic Web Conference, Hiroshima, Japan*, 2004.
16. M. Kay. XSL transformations (XSLT) version 2.0. Recommendation, W3C, 2007.
17. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Recommentation, W3C, January 2008.
18. A. Seaborne. RDQL - a query language for RDF. Member submission, W3C, January 2004.
19. J. van Ossenbruggen, L. Hardman, and L. Rutledge. Towards smart style: combining RDF semantics with XML document transformations. Technical Report INSE0303, CWI, October 2003.