Linkbase Access Protocol Design

Erik Wilde, ETH Zürich (Swiss Federal Institute of Technology)

Abstract

XML itself does not support hypermedia, but the XLink standard has been defined to make XML usable for hypermedia. One of XLinks most interesting features is its support for external links and linkbases, which makes it possible to create links between resources without having to change the resources. In order to use these links, user agents must access linkbases and query them for relevant links, and we present our approach to create a protocol for linkbase access.

1 Introduction

The *Extensible Linking Language (XLink)* [4] is one of the most important developments for the future of the Web as a hypermedia system, because it defines how to use links with XML documents. XLink's features are far more powerful than HTML's linking capabilities, and a fully XLink-enabled Web will provide many new features, in particular *Open Hypermedia* functionality, with links being treated as separate entities (in HTML, links are always embedded in the resource that they are linking).

XLink itself is a stable standard, but some of the problems which also need to be solved in order to make XLink usable are still under investigation. One area is the question of how to display resources which have been assembled from different sub-resources, and W3C has published a note [7] dealing with this problem. Another area is the question of how linkbases can be managed and accessed, and this is the question we address in this paper.

1.1 XLink Linkbases

XLink introduces the notion of a *linkbase*, which is defined by the standard as: "Documents containing collections of inbound and third-party links are called link databases, or linkbases". XLink defines a special type of link (technically, a special *arc role* for links) for locating linkbases, so from XLinks point of view a resource can refer to a linkbase, and an application processing this resource can access the linkbase and extract all relevant links from it. This is a pretty simple way of dealing with linkbases, but from XLink's point of view, it is sufficient.

However, one of the core ideas of *Open Hypermedia* is that linkbases contain a huge number of links, and the resources that are being linked by these links are not aware of it (very similar to a bookmark store in a browser, which also contains links without the resources being linked being aware of it). Thus, linkbase access must be handled in a more powerful and flexible way.

1.2 The Open Web

Our vision of the *Open Web* are browsers which not only access resources, but simultaneously access linkbases every time they access a resource, and query the linkbase for links for this resource. The links being exchanged between the linkbase and the browser should be XLinks, but so far there is no access protocol for linkbases which could be used.

2 Design Axes

Thinking of a *Linkbase Access Protocol*, two major design aspects come into play. The first aspect is the service which is provided by the linkbase, ie the functionality that can be used by the browser. The second aspect is the actual protocol being used for accessing the service, ie the embedding into a specific encoding and some kind of data transport mechanism.

It is important to keep in mind that we do not expect linkbases to store their information as XML/XLink linkbases. We simply define a mechanism for accessing and querying linkbases, and we expect a linkbase to send the response as an XLink linkbase, but the linkbase's internal design is of no concern to us. In fact, it is likely that large linkbases will often use relational database systems rather that XML/XLink for storing the links.

2.1 Service Axis

At first sight, querying a linkbase could be regarded as a special case of querying an XML document, so that the *XML Query Language (XQuery)* [1] would be a good candidate as a query language. However, we believe that XQuery is not the ideal candidate because of two reasons:

- *Complexity*

  XQuery is a rather complex standard, and it is much more complex and powerful than required by the linkbase scenario. Profiling XQuery (if possible) might help, but even then the second reason still would be valid.

- *Data model*

  XQuery builds on the *XML Information Set (XML Infoset)* [3], the abstract data model of XML. XLink extends this model (a first approach to formalizing this model has been given in a W3C note [7]), so the problem of using XQuery for querying linkbases is a mismatch in the data model.

Because of these reasons, we have decided to define our own query method, which is tailored to XLink and rather simple, but powerful enough to query linkbases. Every request in our model has four distinct elements, which are

- *request properties* (describing the request as a whole and specifying what should be returned),
- *size limits* (defining limits for the size of the response, such as the maximum number of links being returned),

- *filters* (filtering links according to certain characteristics, such as arc roles), and
- *attribute assertions* (making further assertions about attributes of XLink elements).

The most interesting aspect of the requests is that they are executed on the XLink Information Set, which we defined according to the W3C note [7]. However, we think that, in the same way as the XML Infoset was a very important specification for a number of activities XLink's information model should also be defined more formally than the prose used in the XLink specification. The W3C note [7] provides a good starting point (however, it does have holes and was never intended to provide a complete model of XLink's contributions to the XML Infoset).

2.2 Protocol Axis

After finishing the service definition, the question remains how to access the service. Looking at today's Web infrastructure, it becomes clear that the transport protocol must be HTTP. However, even when looking at HTTP there are several possibilities how to use it:

- *URI Query Strings*

  In the simplest case, information can be appended to the URI in the request. This is most frequently used with HTML forms using the `GET` method. However, this method is not well suited for larger amounts of data, and also requires all data to be URI encoded.
- *HTTP Extensions*

  HTTP can be extended with new methods and/or headers by using the *HTTP Extension Framework* [5]. This method can be used to introduce new features into HTTP. However, it requires all participating HTTP implementations to be extended to support the new methods and/or headers.
- *HTTP Entities*

  Information can also be transmitted as HTTP entity. This is the approach of the *Simple Object Access Protocol (SOAP)* [6], the most widely used general-purpose method for using HTTP as the foundation for Web services.

We decided to choose the third approach, modelling the linkbase access protocol as a Web service. This gives us the advantage of available tools and infrastructure, in particular the *Web Services Description Language (WSDL)* [2], which we use to define the linkbase access service.

Using this WSDL definition, it is very easy to access the linkbase access service. From the WSDL definitions, stubs can be generated which automatically encode linkbase requests as SOAP messages and automatically decode linkbase responses, interpreting the SOAP message that has been sent as response.

So far, we only provide read-only services, but in future versions, the service definition could be updated to also include write access, such as deleting links from the linkbase or inserting new links into it.

3 Implementation

We have finished a prototypical implementation of a linkbase server, which takes an XML linkbase as input and then uses XSLT to generate an XLink Infoset from it. Another XSLT then interprets the request and selects all links form the XLink Infoset which satisfy the request. A third XSLT then transforms the resulting XLink Infoset into an XML linkbase, which is then returned as response. This implementation is not capable of handling large linkbases, but it is sufficient for test purposes.

In the process described above, we use our own XML Schema for representing the XLink Infoset in XML, so that we can use XSLT for working on it.

4 Further Work

Even though XLink should be one of the most important standards of the XML-based Web, it has not received a lot of attention. This is partly due to the fact that, even though XLink itself has been finalized, some of the really important specifications that would be required to actually implement full XLink support are not available. We hope that the linking and style activity and the linkbase access protocol activity will contribute to make XLink more popular.

Using an open source browser such as Mozilla one could easily think of adding linkbase access protocol support to the browser and making it accessible via a preferences panel, but as long as it is not clear how external links have to be styled, simply adding the linkbase access would not provide much benefit. We therefore hope that the activities surrounding XLink will continue and that in two to three years we will see linkbase support in the major browsers.

References

[1] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu. XQuery 1.0: An XML Query Language. World Wide Web Consortium, Working Draft WD-xquery-20011220, December 2001.

[2] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. World Wide Web Consortium, Note NOTE-wsdl-20010315, March 2001.

[3] John Cowan and Richard Tobin. XML Information Set. World Wide Web Consortium, Recommendation REC-xml-infoset-20011024, October 2001.

[4] Steven J. DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[5] Henrik Frystyk Nielsen, Paul J. Leach, and Scott D. Lawrence. An HTTP Extension Framework. Internet experimental RFC 2774, February 2000.

[6] Nilo Mitra. SOAP Version 1.2 Part 0: Primer. World Wide Web Consortium, Working Draft WD-soap12-part0-20011217, December 2001.

[7] Norman Walsh. XML Linking and Style. World Wide Web Consortium, Note NOTE-xml-link-style-20010605, June 2001.