Proc. of the 5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001), Darmstadt, Springer, LNCS 2163, 2001, pp.95-102

Enhancing Digital Library Documents by A Posteriori Cross Linking Using XSLT

Michael G. Bauer[1] and Günther Specht[2]

[1] Institut für Informatik, TU München
Orleansstraße 34, D-81667 München, Germany
bauermi@in.tum.de
[2] Fakultät für Informatik, TU Ilmenau
Postfach 100565, D-98684 Ilmenau, Germany
gspecht@prakinf.tu-ilmenau.de

Abstract. In this paper we describe a way to enhance existing digital library documents by adding links without modifying the stored documents themselves. We show how to use a combination of XSLT and a host language to access a database with linking information and how to merge documents and links at run-time (a posteriori cross linking). Our approach is already used in the system OMNIS/2, which is an advanced meta system for existing digital library systems and enhances existing digital library systems or retrieval systems by additional storing and indexing of user-defined multimedia documents, automatic and personal linking concepts, annotations, filtering and personalization.

1 Introduction

With the growing amount of information that is available to the individual and the different media types, many different digital library systems were developed. These systems are established tools, but users still miss features that would improve their ability to work with documents in digital library systems as it is common with books printed on paper (i.e. adding references, marking pages and annotating text). This led to the development of the OMNIS/2 system which is a meta system for various existing digital libraries [7]. It equips users with a tool that enables them to search several systems at once and especially to benefit from links between documents of different digital library systems (cross linking of documents). Users are able to add links by themselves (with an authoring tool that is part of OMNIS/2) and we will soon offer links that are generated automatically whenever possible (e.g. for bibliographic references or keywords). Merging links and documents does not require a modification of the source documents and can even be done without harvesting, which would collect all documents and store the documents in a huge private database. This is achieved by the presented technique which adds the links at run-time using XSLT[9] right before the documents are presented to the user. In addition it is possible to annotate external documents without a write permission for the library systems which hold the annotated documents. The annotations are not limited to text and can

P. Constantopoulos and I.T. Sølvberg (Eds.): ECDL 2001, LNCS 2163, pp. 95–102, 2001.

consist of multimedia documents. Users are able to use a personalization feature to create their own view on the documents and to "work" with digital library systems by themselves. In its current version the system has already access to some 80000 documents of the digital library system of the Faculty for Computer Science at TU München.

In the remaining parts of the paper we first discuss briefly the architecture of a meta system on the basis of OMNIS/2 which is necessary to understand the environment in which our a posteriori cross linking technique is implemented. We then explain details of the proposed technique and end with some hints on future developments and improvements.

2 Architecture of OMNIS/2

The meta system OMNIS/2 is separated basically into two layers (see Fig.1). We use established digital library systems as data providers and treat them as large containers with powerful query languages. We assume that the systems use XML to make their data available to the outside world as it is common for modern systems nowadays. It would also be possible to access the digital library systems through a tight coupling by using certain ports the systems provide, but we do not discuss this approach any further in this paper. Our system sits on top of the established digital library systems as a meta system and acts as a service provider to the users, who access OMNIS/2 through a common browser. We decided to implement the meta system as a Java servlet in the Apache webserver. The meta system stores and handles all of the linking information and also the annotations, i.e. multimedia documents that users can upload into the system, in its own relational database. With referential integrity dangling links are avoided within the OMNIS/2 system if referenced user-defined documents are removed. The system itself is not only a meta system for digital library systems, but it can also be used as a stand-alone multimedia digital library system as all features are available as well for user-defined documents.

3 A Posteriori Cross linking

Of course the user has to access all documents through OMNIS/2 to benefit from the additional features. This can either be done by using the extended search facility of OMNIS/2 which itself searches in the underlying systems and annotates the result sets or by following links in documents which are presented (and therefore have been processed) by OMNIS/2. In any case retrieving new documents is processed as follows. The OMNIS/2 system first queries the local meta database to retrieve the information in which external system the requested documents reside. This requires of course that every document has a persistent unique identifier (UID) within the underlying system. For documents in the local database this can be easily achieved; for other digital library systems it is an requirement so that they can be utilized by OMNIS/2. After returning the answer the meta database is queried a second time to retrieve whether there are links

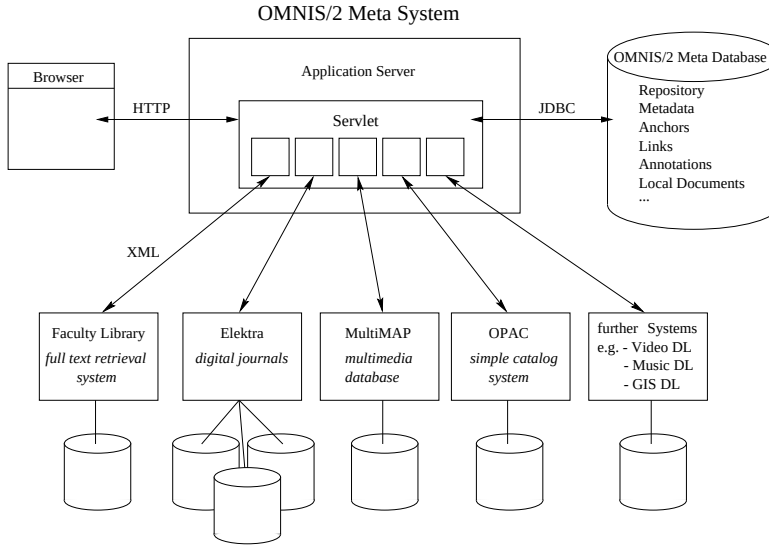

Fig. 1. Architecture of OMNIS/2

or annotations on the retrieved documents. If so we receive all corresponding anchor information. Anchors in this context follow the definition of the Dexter Hypertext Model [1] and in general are an abstraction of link sources and link destinations. Besides their exact position the anchors also need unique identifiers (AIDs) so that they can be associated with a document. After this second query the main part of our a posteriori cross linking is performed. We add the links to the documents and present them to the user.

Merging original XML documents and their linking information can be done with the help of XSLT. The W3C promotes XSLT as a language to transform XML documents into other XML documents, HTML documents or ASCII text. The manipulations of XML are performed by an XSLT processor using an XSLT stylesheet. XSLT processors are available for a variety of host languages (C++, Java, Perl, etc.). XSLT itself though does not provide a way to retrieve data (i.e. links) from a database. It is possible however to use the host language to perform this task and to exchange data with the XSLT style sheet by setting parameters. This combination is the central technique of our idea to utilize a posteriori cross linking in digital libraries. In our case the host language is Java and we chose Xalan[8] from the Apache project as the XSLT processor.

In the following sections we present our techniques using a short XML fragment of a document (in this case it is a simple metadata record) (Fig. 2). In the example we assume that a link to an annotation should be created which gives further information on the title of the XML document. For optimization reasons we combine two steps into one. Links are added to the documents, while the documents are transformed from XML into HTML for the users' browsers.

```
<item>
<dc>
  <identifier>omn:1</identifier>
  <creator> Guenther Specht, Michael G. Bauer </creator>
  <title>
     OMNIS/2 - A Multimedia Meta System for existing Digital Libraries
  </title>
  <subject/>
  <publisher>Springer Verlag, Berlin Heidelberg</publisher>
  <date>2000</date>
</dc>
</item>
```

Fig. 2. Sample XML document (fragment)

3.1 Simple Linking

At first we retrieve the link information (with UID being the key) from the meta database via JDBC (as we use Java as the host language). The next step is to build a complete URL which points to the linked document (which contains e.g. additional information). The information on the URL is available from the target anchor. This URL is the parameter which is passed on to the XSLT stylesheet. With the data provided by the parameters the XSLT stylesheet then creates a valid HTML link at the desired position. If the link includes the whole content of a tag then the content can be easily enclosed into the <A HREF>...</A> hyperlink statement of HTML. The following excerpt (Fig. 3) shows part of a simple XSLT stylesheet in detail.

```
 1 <xsl:param name="titlelink" select="''"/>
 2 <xsl:template match="title">
 3  <H3>Title:
 4     <xsl:if test="$titlelink!=''">
 5       <A><xsl:attribute name="HREF">
 6          <xsl:value-of select="$titlelink"/>
 7          </xsl:attribute>
 8       <xsl:apply-templates/></A>
 9     </xsl:if>
10     <xsl:if test="$titlelink=''">
11       <xsl:apply-templates/>
12     </xsl:if>
13  </H3>
14 </xsl:template>
```

Fig. 3. XSLT stylesheet for simple linking of tags (fragment)

It applies a posteriori cross linking on a <title> tag. The parameter which holds the target URL is called titlelink and it defaults to the empty string (no link). Its value is set by the Java part of OMNIS/2 at run-time. The resulting title string is enclosed in <H3> tags (Fig. 4).

```
<H3>Title:
<A HREF="http://www3.in.tum.de/servlet/omnis2?action=DisplayAnnotation&
id=anno:4">
OMNIS/2 - A Multimedia Meta System for existing Digital Libraries</A>
</H3>
```

Fig. 4. generated HTML code (fragment)

3.2 Linking substrings within tags

If only part of a tag is linked the technique becomes a bit more sophisticated (Fig. 5) and we have to address substrings of the tag and enclose those with the <A HREF>...</A> hyperlink statement (Fig. 6). The anchors have to carry information about the position of the substring within the tag content, but mostly simple integer identification is sufficient for this purpose. In the example below the parameters start and length, which are set by the Java part of OMNIS/2 at run-time, denote the postition of the link. For each anchor these values are stored in the meta database.

```
 1 <xsl:param name="titlelink" select="''"/>
 2 <xsl:template match="title">
 3  <H3>Title:
 4     <xsl:if test="$titlelink!=''">
 5       <xsl:value-of select="substring(., 1, $start - 1)"/>
 6       <A><xsl:attribute name="HREF">
 7          <xsl:value-of select="$titlelink"/>
 8          </xsl:attribute>
 9          <xsl:value-of select="substring(., $start, $length)"/>
10       </A>
11     <xsl:value-of select="substring(., $start + $length + 1)"/>
12     </xsl:if>
13     <xsl:if test="$titlelink=''">
14       <xsl:apply-templates/>
15     </xsl:if>
16  </H3>
17 </xsl:template>
```

Fig. 5. XSLT stylesheet for linking substrings (fragment)

```
<H3>Title: OMNIS/2 - A
<A HREF="http://www3.in.tum.de/servlet/omnis2?action=DisplayAnnotation&
id=anno:4">
Multimedia</A> Meta System for existing Digital Libraries
</H3>
```

Fig. 6. generated HTML code for linking substrings (fragment)

3.3 Advanced linking using Java extensions within XSLT

The above presented techniques show very simple linking mechanisms. They link only exactly once in a tag and (without fundamental changes) work only for standalone documents. If a list of documents is returned (e.g. as a result of querying a digital library system) the single result items cannot be linked with the above presented methods. Each item can be uniquely identified but with the techniques so far it was not possible to transfer information from the XSLT stylesheet to the host language. (The above shown techniques use only the way from the host language to the XSLT stylesheet.) Another problem is that it is not possible to transform only part of an XML document and then update the variables for the remaining transformations with the XSLT stylesheets we have presented up to now. This is of course tightly linked to the above mentioned problem.

Most XSLT processors offer extensions, which enable the use of programming languages from within XSLT stylesheets. As mentioned above we use the Xalan XSLT processor, which offers Java extensions besides others. With a new namespace defined at the beginning of the stylesheet it is then possible to directly call Java methods in a way similar to XSLT functions. As the Java methods can be called with arbitrary parameters we can transfer parts of the XML data to the Java method, manipulate it (e.g. query a database) and return data to the XSLT stylesheet for further transformations. This gives us the flexibility to work beyond the functionality of simple XSLT stylesheets and to achieve the desired results. For the sake of readability we only provide a very short excerpt of a Java extended stylesheet omiting the source code of the Java classes (which carry the JDBC database calls, etc.). `Line 9` in Fig. 7 shows the relevant call to the Java class.

4 Related Work

Some of the concepts of OMNIS/2 (section 2) and a posteriori cross linking have occurred in different forms in the scientific literature over the last years. The first project that we know of which describes shared annotations for webpages is ComMentor [5] from Stanford University. Commentor supports server side

```
 1 <?xml version="1.0"?>
 2 <xsl:stylesheet
 3     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 4     version="1.0"
 5     xmlns:java="http://xml.apache.org/xslt/java"
 6     exclude-result-prefixes="java">
 7  <xsl:template match="title">
 8    <xsl:variable name="titlelink"
 9      select="java:Anchor.getLink(string(../identifier), string(.))"/>
10    <H3>Title: <xsl:value-of select="$titlelink"\> </H3>
11  </xsl:template>
12 </xsl:stylesheet>
```

Fig. 7. A Java call from an XSLT stylesheet (fragment)

annotations but the functionality is limited as the implementation is directly integrated into the source code of the browser.

The OSF Research Institute developed GrAnT (Group Annotation Transducer), which is a proxy based service and support annotations for the World Wide Web [6]. Annotations in GrAnT can only consist of texts and the system does not use a database system for managing annotations.

Project Clio of Grinnell College also deals with annotations on webpages and suggests user permissions and anchors similar to OMNIS/2. The available literature [3] though especially concentrates on user experiences with annotations.

Phelps and Wilensky have described the multivalent document model which supports multivalent annotations [4]. Their concept describes a solution where the annotations are stored on the client side and a special viewer is used to be able to benefit from the additional features.

5 Summary and Future Work

We have shown how to enhance existing digital library documents by adding links without modifying the source documents. We are using accepted standards like XML, XSLT and Java for this purpose which is an advance over solutions using proprietary implementations. The added links are stored separately from the source documents and can reference either documents within the same digital library, documents from other digital library systems or user-defined annotations. We call our presented technique a posteriori cross linking. Users can benefit from it because they can work with the digital library documents (add references and annotations) in a fashion that comes close to working with classic paper documents. Further improvements we want to implement in a later version target the transformation formats within the OMNIS/2 system. In our description above we have inserted the links directly while the XML source documents were transformed into HTML. The design of the transformation could be improved by creating XLink and XPointer from the stored link data. This would lead to an

XML only data format that can then be transformed with XSLT to HTML in a final step. This would result in a greater flexibility as XML shall replace HTML within the next years. In the future browsers will be able to display the XML format directly and OMNIS/2 could avoid the final step of transformation. We also plan to investigate the ability of OMNIS/2 to act as a service provider for the Open Archives Initiative [2], as the initiative proposes an XML based exchange format, which OMNIS/2 can work with as presented in this paper.

6 Acknowledgments

OMNIS/2 is funded by DFG (German Research Foundation) within the research initiative "V3D2" ("Distributed Processing and Delivery of Digital Documents") and is part of the Global Inventory Project (an initiative of the G7-countries).

References

1. Halasz F., Schwartz M., *The Dexter Hypertext Reference Model*, Comm. of the ACM, 37(2), Feb. 1994, pp. 30-39.
2. *Homepage of the Open Archives Initiative*, http://www.openarchives.org
3. Luebke S.M., Mason H.A., Rebelsky S.A., *Annotating the World-Wide-Web*, Proc. of ED-Media 99 World Conference on Educational Multimedia, Hypermedia and Telecommunications, June 1999, pp. 409-414.
4. Phelps T.A., Wilensky R., *Multivalent Annotations*, Proc. of the 1st European Conf. on Research and Advanced Technology for Digital Libraries (ECDL 1997), Springer, LNCS 1324, 1997, pp. 287-303.
5. Röscheisen M., Mogensen C., Winograd T., *Beyond Browsing: Shared Comments, SOAPs, Trails, and On-Line Communities*, Proc. of the 3rd World Wide Web Conference, April 10-14, 1995, Darmstadt, Germany.
6. Schickler M.A., Mazer M.S., Brooks C., *Pan-Browser Support for Annotations and Other Meta-Information on the World Wide Web*, Proc. of the 5th World Wide Web Conference, May 6-10, 1996, Paris, France.
7. Specht G., Bauer M.G., *OMNIS/2: A Multimedia Meta System for existing Digital Libraries*, Proc. of the 4th European Conf. on Research and Advanced Technology for Digital Libraries (ECDL 2000), Springer, LNCS 1923, 2000, pp. 180-189.
8. *The Apache XML Project*, http://xml.apache.org/
9. *XSL Transformations (XSLT), Version 1.0*, http://www.w3.org/TR/xslt