KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 236–243

MANAGEMENT OF WEB PAGES USING XML DOCUMENTS

LEON ŢÂMBULEA AND HORIA F. POP

Abstract. The management of a web site is a very difficult task. The paper describes a way of automatic management by memorizing in a database the information sources in different pages. To describe the way to generate a page, a page context is used, which is stated through an XML document.

1. Motivation

Many models are proposed to allow the management of complex web sites (see [1, 2, 3, 4, 5]). In [6] a model allowing collaboration/approval of activity of many users in the development of site is introduced.

This paper describes a modality for automatic management of complex web sites through memorising the site contents in a database and stating the contents of a web page through an XML document.

2. Contents of a web site

The visualisation of a web site (by an Internet browser) meas the succesive display of more pages. We start from an initial page, and following the occurence of a certain event we continue with a different page.

A **page** is built (generated) from a set of elements (sources of information). These elements have different types:

- static elements: blocks of HTML text, XML documents, documents of different types (DOC, RTF, XLS, PPT, and so on);
- menus;
- result of the execution of server scripts;
- and so on.

For each **element type** more elements (realisations of this element type) may exist. Such an element may have, among others:

- an identification (id and type);
- a contents (a sequence of characters that defines this element).

2000 *Mathematics Subject Classification.* 68P99.
Key words and phrases. web sites, management, content management systems, XML.

These values may be memorised in independent files (and the identification is made through the file name), or in a database.

For an **element** e, a **value** $v(e)$ may be determined. This value is used to view the element. The way to obtain the value depends on the element type: for a static element, the value is the contents, and for a server script, the value is obtained by interpreting (executing) the script.

The value $v(e)$ may include all the information required by the browser to view. Alternatively, when viewing extrainformation is used, memorised in a **viewing style** s asociated to the element e.

In what follows we will denote by E the set of all elements, disregarding their type, and by S the set of all styles, defined in order to be associated to differet elements in E.

When generating a **web page** the following information is considered:

- A set of necessary elements in the page (subset of E). For each element e its value $v(e)$ (text to be interpreted by the browser) is determined. When generating the element value, a viewing style may be used.
- Positioning the value of each element in the generated page. Such a value will occupy a 2D area on the display where the page will be viewed.
- A viewing style for the whole page, which completes the viewing styles associated to elements.

After a page p is generated and displayed, a new page p' will be generated following an event. The event in page p (for example, the use of a link) appears in the value of an element e in this page. When defining the element a reference to an element $e' \in E$ is needed, or a link to an external document is produced. If there is a reference to an alement e', then from e' the page p' will have to be generated. This generation of the page p' may be done in two different ways:

(1) Each referred element e' is ssociated a set of elements, $A(e') \subset E$. From these elements the new page p' is generated. This association is accompanied by the position each element in $\{e'\} \cup A(e')$ will use on the newly generated page, p'. It is definitely possible that the same page to be generated by events in different elements, so all positioninginformation should be repeated. The main advantage of this approach is that the same elements may generate distinct views, based on the generating event.

(2) A set of **viewing contexts** C are defined. A context $c \in C$ contains:
   (a) a set of elements $c(e) \subset E$;
   (b) a [positioning of the vaues of these elements in the generated page,
   (c) a style associated to the conext.

   Using this information for a context, a page p may be generated. An element e is associated a context $c(e) \in C$. If $e \in c(e)$, then the page is generated using the context $c(e)$. If $e \notin c(e)$, then we need to state a

position of the value of element e in the page generated byu the context $c(e)$.

In what follows we will consider the second alternative. Thus, for each element $e \in E$, the following information is associated:

- a **context** $c(e) \in C$, if e is referred by a certain event;
- the value **null**, if the element e will be included in a viewing context, but is not referred by any possible event (for example, a commercial, banner or block of text in an HTML page, that appear in a certain context, but they are not pointed to by any links in the generated pages.

A context may be associated to more elements, i.e. by referring different events, the same displayed page is obtained.

Viewing a page actually means viewing a context. The first step in this process is to determine the value of each element in the context and to include this value in a certain area in the generated page.

In many situations, the value of an element does not have a constant length, especially if the element is a server script and the value of this element depends on the contents of a database. So, in the generated page, the element value will ocupy an area with no exact area. If, when generating the value of such a context, values of more elements are used, then the HTML text generated from the context, used to view the context, will be formed by HTML tags `<table> ... </table>`. So, we will get a table where the cells will hold the value of an element, or a new included table, corresponding to a subcontext.

The size of a cell holding the value of an element:

- May be determined by the browser based on the values of the other cells in the table and the size of the parent area where the table is included;
- One of the sizes (width, height) is stated through an absolute or relative value with respect to the parent area. The other size will be determined by the browser.
- Both sizes are stated. If the value does not fit in the reserved area, then one of the sizes will be automatically increased by the browser.

3. Description of a web context as an XML document

Let us consider the context depicted in Figure 1, where we denoted by 1, 2, ..., 5a, ..., 10, elements of a particular type t, and the parentheses specify the width of the area where the value is included. The value p% specifies that the width of the cell is relative to the width of the parent area.

The HTML text that allows the extraction of this context should be generated from the "context description", defined as a suitable data structure. This data structure should allow:

(1) Easy run of operations to modify the context:
   (a) insert elements,

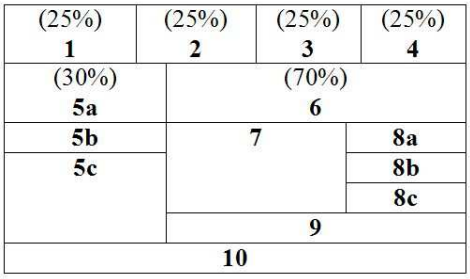

FIGURE 1. Example of a web site context

(b) remove elements of sets of elements that occupy a certain 2D area in the page,
(c) move elments.

(2) HTML text generation by determining the values of included elements and their positioning in the generated page, according the the pattern established by the context.

We will now describe a possibility to memorize contexts by using XML documents.

If two lines in a table may have different structures, i.e. different mumbers of included elements and/or different sizes for these elements, if a line has more than one cell, such a line requires the inclusion of an independent table.

The HTML text that will generate the context view will have `<table> ... </table>` tags. If the use of unions of columns or lines in the table is not esired (`rowspan` and `colspan` attributes in `td`), then not any context may be generated using `<table> ... </table>` tags. See for example the context in Figure 2.

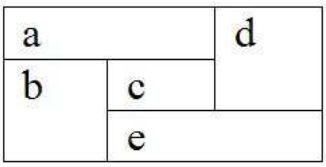

FIGURE 2. A web site context that cannot be generated with table tags and without unions of rows and columns

We are hereby assuming that the tables do not use unions of lines or columns. This particularity for tables is useful in order to easily and quickly perform context management operations (remove, insert, move elements). If, still, situations like that described in Figure 2 are needed, we may define a new element, to contain, by

its definition, a table with this structure, and the context will include this complex element.

Let us enumerate the possbilities to memorize the elements positions in a context.

(1) For the case mentioned above (table with no unions) an XML document may be used. This possibility will be analysed below.
(2) We start from a 2D grid over the display, and the columns may have szes associated. A rectangular area in tis 2D grid is stated for each element included in the contex. This situation is considered with the development of interfaces in 2D visual environments. With such a structure, elements overlaps are possible if the 2D grid has cells of fixed size. Insertions of new elements are more difficult with this way of memorizing the elements positioning.
(3) Even `<table>` tags may be memorized. In this way, the context includes elements, instead of their value.

We are analyzing here the first alternative. This choice does not restrict the possibilities to use this approach, because the largest amount of information on Internet corresponds to this alternative.

We are going to describe the way to generate the HTML text for a context, and the correspondence between the XML tags and HTML tags.

For the XML document we consider the tag `<page [attributes]>...</page>` corresponding to the context generated page. The context, as it has been considered above, has oneor more independent lines. The structure of a line does not generally depend on the structure of another line. When generating the HTML text for a page, a table structure is needed. When generating the table, we need to consider that each line has one value. If one line has more values, then the line will be extracted as an included table. So, two types of lines are possible:

- Line that has one column only, i.e. the value of one element or one included subcontex. The generated HTML text will have `<tr>...</tr>`, and for the XML document we will consier a tag `<rowe [attributes]> ... </rowe>`.
- Line with more than one columns. Will generate an HTML text of the form `<tr><td><table><tr> ... </tr></table></td></tr>`. For such lines, we will consider the XML tag `<rowc [attributes]> ... </rowc>`.

With these considerations, for the context in Figure 1, the following associated XML document will be generated:

```
<table>                                    <page>
  line 1, as table with 1 line, 4 cols       <rowc> line 1 </rowc>
  line 2, as table with 1 line, 2 cols       <rowc> line 2 </rowc>
  line 3, with value of one element          <rowe> line 3 </rowe>
</table>                                   </page>
```

The third line has only one value, so the HTML text and the XML document are:

`<tr><td> v('t',10) </td></tr>`	`<rowe><elem type='t' id='10' /><rowe>`

In the HTML text, the value of an element of type 't' will be denoted by $v('t', id)$. In the XML document, the reference to an element 'id' of type 't' will be done by attributes `type='t' id='id'`.

The first line has four values. So, the HTML text will include a table, and this table, with only one line, will have four columns with the values of one element each:

`<tr>`
`  <td>`
`  <table>`
`  <tr>`
	`<rowc>`

`  <td> v('t','1') </td>`
`  <td> v('t','2') </td>`
`  <td> v('t','3') </td>`
`  <td> v('t','4') </td>`
	`  <elem type='t' id='1' />`
	`  <elem type='t' id='2' />`
	`  <elem type='t' id='3' />`
	`  <elem type='t' id='4' />`

`  </tr>`
`  </table>`
`  </td>`
`</tr>`
	`</rowc>`

The second line in the original table has a more complex structure. To describe a coluumn with more elements, we will consider a tag `<col [attributes]> ... </col>`. For this tag we will create a table cell that includes a new table. So, the generated HTML text is of the form `<td [attributes]><table> ... </table></td>`. In this subcontext, the first two columns are formed by values of more elements, so context tables are needed.

From the examples above, the correspondence between HTML tags and XML tags for memorizing contexts is deduced. This correspondence is given in Figfure 3.

XML tag	HTML tag
<page [attributes]> </page>	<table [attributes]> </table>
<rowe [attributes]> </rowe>	<tr [attributes]> </tr>
<rowc [attributes]> </rowc>	<tr><td><table [attributes]><tr> </tr></table></td></tr>
<elem [attributes] type='t' id='id' />	<td [attributes]> value of element of type 't' with identifier 'id' </td>
<col [attributes]> </col>	<td [attributes]><table> </table></td>

FIGURE 3. The correspondence between HTML tags and XML tags for memorizing contexts

By considering the tag attributes as well, we obtain the correspondence between the XML document and the HTML text from Figure 4 for the context in Figure 1.

<table border=1 width="90%">	<page width="90%">
<tr><td><table border=1 width="100%"><tr>	<rowc width="100%">
<td width="25%" valign="top">v('t','1')</td>	<elem type='t' id='1' width="25%"/>
<td width="25%" valign="top">v('t','2')</td>	<elem type='t' id='2' width="25%"/>
<td width="25%" valign="top">v('t','3')</td>	<elem type='t' id='3' width="25%"/>
<td width="25%" valign="top">v('t','4')</td>	<elem type='t' id='4' width="25%"/>
</tr></table></td></tr>	</rowc>
<tr><td><table border=1 width="100%"><tr>	<rowc>
<td width="30%" valign="top">	<col width="30%">
<table border=1 width="100%">	
<tr><td>v('t','5a')</td></tr>	<rowe><elem type='t' id='5a'/></rowe>
<tr><td>v('t','5b')</td></tr>	<rowe><elem type='t' id='5b'/></rowe>
<tr><td>v('t','5c')</td></tr>	<rowe><elem type='t' id='5c'/></rowe>
</table>	</col>
</td>	
<td width="70%" valign="top">	<col width="70%">
<table border=1 width="100%">	
<tr><td>v('t','6')</td></tr>	<rowe><elem type='t' id='6'/></rowe>
<tr><td><table border=1 width="100%"><tr>	<rowc>
<td width="60%" valign="top">v('t','7')</td>	<elem type='t' id='7' width="60%"/>
<td width="40%" valign="top">	<col width="40%">
<table border=1 width="100%">	
<tr><td>v('t','8a')</td></tr>	<rowe><elem type='t' id='8a'/></rowe>
<tr><td>v('t','8b')</td></tr>	<rowe><elem type='t' id='8b'/></rowe>
<tr><td>v('t','8c')</td></tr>	<rowe><elem type='t' id='8c'/></rowe>
</table>	</col>
</td>	
</tr></table></td></tr>	</rowc>
<tr><td>v('t','9')</td></tr>	<rowe><elem type='t' id='9'/></rowe>
</table></td>	</col>
</tr></table></td></tr>	</rowc>
<tr><td>v('t','10')</td></tr>	<rowe><elem type='t' id='10'/></rowe>
</table>	</page>

FIGURE 4. The correspondence between the XML document and the HTML text for the context in Figure 1

4. Conclusions

The management of a web site is a very difficult task. The paper describes a way of automatic management by memorizing in a database the information sources in different sources. To describe the way to generate a page, a page context is used, which is stated through an XML document.

The construction of this XML document is difficult to accomplish, especially for pages with a lot of information. For this reason, an application to allow the interactive editing of such an XML document is desired.

References

[1] *PhpWebSite project*, Appalachian State University, 2004-2006, http://phpwebsite.appstate.edu.

[2] A. Bonifati, S. Ceri, P. Fraternali, A. Maurino, *Building multi-device, content-centric applications using WebML and the W3I3 Tool Suite*, in Proceedings of Conceptual Modeling for E-business and the Web, Lecture Notes in Computer Science, vol. 1921, 2000, pp. 64–75.

[3] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, Al. Maedche, B. Motik, D. Oberle, Ch. Schmitz, St. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, V. Zacharias, *KAON – Towards a large scale Semantic Web*, Proc. of the 3rd Intl. Conf. on E-Commerce and Web Technologies (EC-Web 2002), 2002, pp. 304–313.

[4] Y. Jin, S. Decker, G. Wiederhold, *OntoWebber: model-driven ontology-based web site management*, in Proceedings of the first international semantic web working symposium (SWWS'01), Stanford University, Stanford, CA, 29 July – 1 August 2001.

[5] L. Ţâmbulea, H. F. Pop, *Web sites management*, BABES-BOLYAI University of Cluj-Napoca, Faculty of Mathematics and Computer Science, Proceedings of the Symposium "Zilele Academice Clujene", 2006, p. 21–26.

[6] L. Ţâmbulea, H. F. Pop, *Cooperative Model For Web Sites Authoring*, International Workshop in Collaborative Systems, Cluj-Napoca 2006, Annals of the Tiberiu Popoviciu Seminar, Cluj-Napoca, 28–29 October 2006, 329–336.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1, M. Kogalniceau Street, 400084 Cluj-Napoca, Romania

E-mail address: leon@cs.ubbcluj.ro

E-mail address: hfpop@cs.ubbcluj.ro