Developer-Friendly Annotation-Based HTML-to-XML Transformation Technology

Lendle Chun-Hsiung Tseng
Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology
lendle_tseng@seed.net.tw

ABSTRACT

Nowadays, the amount of information accessible on the web is huge. Although web users today expect a more integrated way to access information on the web, it is still rather difficult to “integrate” information from different web sites since most web pages are authored in HTML format, which is actually a presentation-oriented language and is usually considered unstructured. Today, there are many research works aiming at extracting information from web pages. Existing works typically transform the extracting results into structured or semi-structured data formats, thus other applications can further process the results to discover more useful information. Nevertheless, the unstructured nature of HTML makes the transformation process complex and can hardly be widely adopted. In this paper, an annotation-based HTML-to-XML transformation technology is proposed. The mechanism is developed with both usability and simplicity in mind. With the proposed mechanism, ordinary web site developers simply add annotations to their web pages. Annotated web pages can then be processed by our software libraries and transformed into XML documents, which are machine-understandable. Software agents thus can be developed based on our technology.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Design

Keywords

XML, Mashup, Information Extraction, Annotation

1. INTRODUCTION

It is without doubt that adopting XML data format in electronic data exchange has advantages. XML data format is machine-understandable, which means an XML document can be “understood” by software agents. Furthermore, since an XML document can be associated with XML schemas, strict data type information is available for preventing misunderstanding. Hence, integration among several XML information data sources is possible. However, compared with the number of information sources on the web, the number of XML-enabled ones is much smaller. Most web sites deliver information in a data format that is usually considered as unstructured and not machine-understandable, say, the HTML data format. The original purpose of the HTML data format is for layout/presentation only. As a result, the data format lacks of some key concept for modeling data, such as type information. The situation reduces the usability of the huge amount of information.

To overcome these issues, several information extraction technologies have been developed. Basically, existing technologies provide means for locating desired regions of information from HTML documents and then transform them into more structured representations. Conceptually, existing technologies can be categorized as automatic ones and manual ones. The former ones are convenient, but with the cost of less accuracy and flexibility. The latter ones are accurate and flexible but require developers to define detailed rules and even write programs.

In this paper, the author proposes an alternative mechanism for extracting information from HTML documents. With the proposed mechanism, web site developers at first annotate their HTML pages with the provided set of annotations. These annotations are used for declaring the relationships between HTML nodes and the target XML nodes as well as the positioning rules for these XML nodes. After the transformation, HTML documents will be converted to XML data conforming to some specified XML schemas. As a result, the extracted information can be “understood” by software agents that have knowledge about the corresponding XML schemas. A software library is also provided for performing the transformation. The proposed technology is developer-friendly because of the following criterion:

1. The required annotations are easy to understand and easy to use.
2. Adopting these annotations will not affect the normal functionality of a web page.
3. The degree of flexibility that can be achieved by the proposed mechanism is close to which can be achieved by other programming-based approaches such as XSLT [9].

The resulting software library is named as H2X, which stands for “Html to XML transformation”. H2X is a java-based software library and can be adopted in both standalone applications and server-side applications.

2. RELATED WORK

Rather than simple web surfing, web users nowadays request for an integrated view of information on the web. Roughly speaking, web information extraction systems can be divided into four different types: manually constructed systems, supervised systems, semi-supervised systems, and unsupervised systems [2]. Manually constructed systems such as WebOQL [1] require users to use provided programming or query languages for extraction and thus become extremely expensive. Supervised systems such as DEByE [5] require users to provide a set of pre-labeled web pages for training and then will automatically infer rules for extracting data from similar web pages. Semi-supervised systems such as OLERA [3] require no pre-labeled training pages, but post-efforts from the user is required to choose the target pattern and indicate the data to be extracted. Unsupervised systems, on the other hand, require no user interactions. However, they may only be applicable for data-rich regions, e.g. the table part, of a web page. The work proposed by Wang and Lochovsky [10] is an unsupervised web information extraction system.

Generating XML documents from information sources is not an uncommon request for web information extraction. There are several technologies and research works targeting at providing such functionality. For example, XSLT [9] is a popular technology aiming at transforming an existing XML document (probably an XHTML document) into a new one. Besides programming-based transformation, some technologies may rely on pre-specified mappings to generate XML documents from user inputs on GUI components. Examples of applications/software libraries adopting such technology are XForms [8], FormsXML [4], InfoPath [7], and XUIB [6].

Compared with existing works, the proposed mechanism appears to be both feature-rich and user-friendly. XSLT-based approaches require programming skills. XForms-based technologies need special browser support and thus have not been widely adopted. Most mapping-based technologies lack of flexibility due to the constraints caused by XML schemas. XUIB is flexible, but probably too complex for ordinary developers. On the other hand, the proposed mechanism requires developers to add only needed annotation and can determine some transformation rules automatically.

3. Annotation-Based HTML-to-XML Transformation

The proposed mechanism is aimed at providing a transformation framework for generating XML documents from annotation-augmented HTML documents. Here, it is assumed that the HTML documents to be processed are always XHTML-compliant. If not, a tidy process will be invoked in advance to ensure XHTML-compliance. An annotation-augmented HTML document is an HTML document with HTML nodes augmented by annotations. The augmenting information is then used as rules for generating the target XML document. Two types of rules are needed during the generation process:

1. node construction rules that are used for providing the specification of the target XML nodes
2. positioning rules that are used for determining the final positions of generated XML nodes

That is, for all nodes (including element nodes, attribute nodes, and text nodes) within the HTML document to be processed, a *(node, node construction rule, positioning rule)* tuple is maintained. The transformation process is listed below:

1. walk through the HTML DOM tree and construct a meta tree, the transforming tree, which reflects the original tree structure; nodes on the transforming tree, the transforming tree nodes, maintain the original HTML DOM nodes and associated annotation information
2. for each HTML node, create the corresponding XML node according to its associated construction rules and store the generated XML node into the meta tree
3. traverse the meta tree and position XML node contained in each tree node according to its explicit or implicit (will be explained later) positioning rule

The proposed mechanism defines the following extended attributes for specifying the augmenting information:

1. *as_root*: an element node annotated with this attribute is regarded to as a root node in the resulting XML document; if there are more than one element HTML nodes annotated with this attribute, more than one XML documents will be generated from this HTML document
2. *root*: specify the root XML element of the resulting XML node generated from the annotated target
3. *def*: specify how to generate a resulting XML node from the annotated target
4. *relation*: define the relationship among the resulting XML node and other XML nodes; allowable relationships are *child* and *next*; these relationships are used for positioning resulting XML nodes

Most extended attributes expect values in the form: (*target_exp, parameter*). *target_exp* is used for locating the target nodes to apply this annotation. The design of *target_exp* allows developers to annotate HTML nodes more freely. For example,

<input type="text" value="book1" h2x:def="@{}value, @{}bookName"></input>

indicates that the attribute node "value" is associated with (and will be transformed to) the "bookName" attribute on the target XML page. Multiple rules separated by ";" can be defined within one annotation if they have different *target_exp*. For instance, *h2x:def="@{}value, @{}bookName; ., {}bookMetaData"* defines an additional rule that will map the annotated HTML node itself to the "bookMetaData" element on the target XML page.

Furthermore, two types of positioning rules, the explicit and the implicit ones, are used for determining the final position of a generated XML node. Explicit positioning rules are specified with the *relation* annotation and have the highest priority. Implicit rules are adopted when no explicit rule is specified. Two types of implicit rules will be adopted when no explicit rule declared (listed from highest to lowest priority):

1. schema position: to position a transforming tree node according to the restrictions imposed by the associated XML schema
2. natural position: to position a transforming tree node according to the relative position of nodes with *def* annotations

Schema position needs more explanation. For an XML element with complex content type, the allowable sequential order of its child nodes has to fit the constraint imposed by the corresponding schema. Without loss of generality, one can construct a state machine to represent the constraint. Each transition arc of the state machine is labeled with the qualified name of a legal child node. By traversing along the transition path from the start state to the end state, one or more legal child node sequences can be determined. The proposed mechanism utilizes an approach based on this to discover a legal child node sequence. As a result, developers do not have to explicitly position every node and hence their efforts can be greatly reduced.

4. AN EXAMPLE

In this section, an example is used for demonstrating the concept of the proposed mechanism. At first, image an online book-shopping agent service that is designed as a portal for browsing/buying books from several other online book stores. The agent may want to provide more advanced services such as price comparison. To achieve the goal, the agent has to extract information form several online book stores. A commonly-used approach is web page scraping. This can be difficult and unstable since web pages of different book stores will organize information in different ways. Note that although different web sites have different layouts, it is still possible that they adopt the same data model. Assume the following (simplified) XML schema is used as the common data model:

```
<xs:schema targetNamespace="test1"
 xmlns:editix="test1"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="books">
   <xs:complexType>
    <xs:sequence maxOccurs="unbounded">
     <xs:element ref="editix:book"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="book">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="editix:title"/>
     <xs:element ref="editix:author"/>
     <xs:element ref="editix:price"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="author" type="xs:string"/>
  <xs:element name="price" type="xs:double"/>
</xs:schema>
```

Instead of relying on web page scraping for extracting information from web pages, here, we demonstrate how to associate a normal XHTML web page with the above data model through the proposed approach. Below is an annotation-augmented web page:

```
<html xmlns:h2x="h2x">
 <head>
   <title>TODO supply a title</title>
 </head>
 <body h2x:def="{test1}books"
  h2x:as_root="books.xsd">
 Books:<br/>
 <ul>
  <li h2x:def="{test1}book" id="book1">
   <table border="1">
    <tbody>
     <tr h2x:def="{test1}price" id="price1">
     <td>Price</td>
     <td h2x:def="text(),text()">100</td>
    </tr>
     <tr h2x:def="{test1}author" id="author1"
         h2x:relation="id(title1),.,next">
     <td>Author</td>
     <td h2x:def="text(),text()">Author1</td>
    </tr>
    <tr h2x:def="{test1}title" id="title1">
     <td>Title</td>
     <td h2x:def="text(),text()">Book1</td>
    </tr>
    </tbody>
   </table>
  </li>
 </ul>
 </body>
</html>
```

As shown in the above example, the <body> element is associated with the {test1}books schema element via the *def* annotation and is defined as a root node. Note there are several different syntaxes of the *def* annotation shown in the example. For instance, the <td> element with "Book1" as its text content is annotated with "*h2x:def="text(),text()"*". The annotation specifies that the text content of the <td> element is mapped to a text node in the target XML document. Another example is the <tr> element with id "*author1*". According to the annotation, "*h2x:def="{test1}author"*", this <tr> element is mapped to an element node with "*{test1}author*" as the qualified name.

It is possible that, the hierarchical structure of the XHTML document is different from the target XML document. The "*relation*" annotation is used for dealing with this scenario. For instance, the <tr> element with id *author1* is annotated with "*h2x:relation="id(title1),.,next"*". The annotation states that the

XML node generated by this XHTML node is the next sibling node of the XML node generated by the XHTML node with id *title1*. Furthermore, XHTML nodes not associated with the *relation* annotation will be positioned according to implicit positioning rules, so there is no need to specify positioning rules for all XHTML nodes. For example, XHTML node with id *price1* and *title1* will be automatically positioned according to the associated XML schema.

Below is the XML document generated from the sample XHTML document:

```
<books xmlns="test1">
 <book>
   <title>Book1</title>
  <author>Author1</author>
 <price>100</price>
 </book>
</books>
```

Now, consider another web page with a different layout:

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h2x="h2x">
   <head>
      <title>TODO supply a title</title>
   </head>
   <body h2x:def="{test1}books" h2x:as_root="books.xsd">
      Books:<br/>
      <ol>
         <li h2x:def="{test1}book" id="book1">
            <ul>
               <a h2x:def="{test1}title" id="title1"></a><li
h2x:relation="id(title1),text(),child"
h2x:def="text(),text()">Book1</li>
               <a h2x:def="{test1}price" id="price1"></a><li
h2x:relation="id(price1),text(),child"
h2x:def="text(),text()">100</li>
               <a h2x:def="{test1}author" id="author1"></a><li
h2x:relation="id(author1),text(),child"
h2x:def="text(),text()">author1</li>
            </ul>
         </li>
      </ol>
   </body>
</html>
```

Note that in the above web page, dummy <a> tags are introduced as the container node holding the information for the generation of some XML elements. The resulting XML document is compatible with the previous one since they adhere to the same XML schema. Hence, software agents can understand information extracted from both web sites. As a result, instead of performing web page scraping, the agent service mentioned above can simply extract information from the resulting XML documents. This will be simpler and more stable.

5. CONCLUSION AND FUTURE WORK

In this paper, an annotation-based technology for HTML to XHTML transformation is proposed. Utilizing the provided set of annotations, web site developers can specify transformation rules directly on HTML documents. These annotations can configure not only the generation of XML nodes but also the positioning rules of the generated XML nodes. Despite of the richness in functionalities, the proposed mechanism is carefully designed to reduce the usage complexity. By employing schema information associated with the HTML document, the mechanism is capable of determining the position of most generated XML nodes. Hence, web site developers do not have to specify positioning rules for all generated XML nodes. Furthermore, additional tool support is not needed for adopting the proposed technology. Developers can utilize the proposed technology with their acquainted HTML tools.

In the near future, it is scheduled to extend the current work to provide scripting support. By allowing scripting in the value part of annotations, flexibility can be further enhanced; take the "id(*#id*)" style target expression as an example, with scripting support, the "*#id*" can be determined by evaluating an scripting function.

Moreover, it is planned to implement the web database concept based on H2X. With H2X, it will be easier to extract information from the web; as a result, it appears reasonable to regard the web as a huge database of machine-understandable information; by extending current design with ontology concept and query functionalities, it will be possible to perform more advanced queries than current search engines can support. For example, type information can be included in a query thus the result will be more accurate.

6. REFERENCES

[1] Arocena, G.O. and Mendelzon, A.O. 1998. WebOQL: Restructuring documents, databases, and webs. In Proceedings of the 14th IEEE international conference on Data engineering, 1998, 24-33

[2] Chang, C.H. and Girgis, M. 2006. A survey of web information extraction systems. IEEE Transactions on Knowledge and Data Engineering 18, 1411-1428

[3] Chang, C.H. and Kuo, S.C. 2004. OLERA: A semisupervised approach for web data extraction with visual support. IEEE Intelligent Systems 19, 56-64

[4] Kuo, Y.S., Shih, N.C., Tseng, L., and Hu, H.C.: Generating form-based user interfaces for xml vocabularies. In Proceedings of the 2005 ACM symposium on Document engineering, pp. 58–60. ACM, New York, NY, USA, 2005

[5] Laender, A.H.F., Ribeiro, B., and Silva, A.S. 2002. DEByE---Data extraction by example. Data and Knowledge 40, 121-154.

[6] Lendle Tseng, Y.S. Kuo, Hsiu-Hui Lee, and Chuen-Liang Chen: XUIB: XML to User Interface Binding. In Proceedings of the 2010 ACM Symposium on Document Engineering, 2010, page 51-60

[7] Microsoft: Infopath. http://office.microsoft.com/en-us/default.aspx

[8] W3C: XForms. http://www.w3.org/MarkUp/Forms/

[9] W3C: XSLT. http://www.w3.org/TR/xslt

[10] Wang, J. and Lochovsky, F.H. 2003. Data extraction and label assignment for web databases. In Proceedings of the 12th international workshop on World wide web, 2003, 187-196