XML Based Course Websites

Michael Wollowski
Computer Science and Software Engineering Department
Rose-Hulman Institute of Technology
United States of America
wollowski@rose-hulman.edu

Abstract: XML, the extensible markup language is a quickly evolving technology that presents a viable alternative to courseware products and promises to ease the burden of web authors who edit their course pages directly. XML uses tags to label kinds of contents rather than format information. The use of XML enables faculty to focus on providing contents, leaving the task of rendering contents to experts who provide a single stylesheet used for formatting purposes. This stylesheet has to be edited once, saving time and effort and ensuring consistent appearance of course pages that reference it. However, the major benefit of XML is the ability to provide pinpoint search engines. Additionally, web-based editors can be provided to make editing pages easier.

Introduction

The *Extensible Mark-up Language (XML)* is a promising new technology for creating, maintaining, and searching course pages. XML shows the most promise when it comes to searching for information. Furthermore, the use of XML reduces the overhead of editing web-pages by separating form and contents. Content providers have to do just that, provide the contents. Similar web-pages are rendered by a common stylesheet. It only has to be edited once. A common stylesheet ensures uniform appearance of web-pages that are of the same kind. This in turn provides for ease of navigation and recognition of location.

When it comes to course web-sites, there are largely two alternatives: to edit and maintain them directly or to use courseware products such as (WebCT 2002) or (Blackboard 2002). The benefits of direct editing are complete control over appearance and contents; the major drawback is a lot of work formatting contents. The benefits of courseware products are ease of use and integration with other academic systems, such as the Banner; a drawback is that formatting and functionality are as provided. We propose a third option which is slowly become a viable alternative: the use of XML. We will show in this paper that XML shares most of the benefits of the two current alternatives: ease of contents creation and maintenance as well as complete control over appearance and contents. A major added benefit of XML is the ability to furnish very precise search engines, enabling users such as students, prospective students, or prospective employers to quickly locate relevant information.

Industry started using XML a while ago, chiefly to unify information between companies and their suppliers (Goldfarb & Prescod 2000). We show an application of this technology that demonstrates its benefits for academic use. We will briefly explain the main components of XML and in the process show how XML separates form and contents. We will demonstrate how common information can be placed into separate files, greatly aiding in the maintenance and consistency of course sites. We will show how to is to provide web-based editors as well as search engines for course pages. Finally, we will report on the status of the project.

Separation of Form and Contents

XML is three technologies in one. It consists of *XML* proper, which is a language for labeling contents. There are *Document Type Definitions (DTDs)*, which are used to specify kinds of documents. Among others, DTDs specify the labels to be used in XML documents. Finally, there are languages to render XML documents. Typically, web-authors use either the *Extensible Stylesheet Language (XSL)* or *Cascading Stylesheets (CSS)*. Since XSL is more powerful when it comes to the way final documents can

be composed, we chose XSL over CSS. At this point, only Internet Explorer 5.5 and higher support XSL. Most other browsers support only CSS.

In what follows, we present an example of an XML document containing a course description of one of our courses. We want to show how XML supports the separation of form and contents. The challenges of dealing with larger documents such as syllabi will be addressed in the conclusions. Figure 1 shows an XML file containing course description information for CSSE 100, one of our courses. In general, white space does not matter and should be used to enhance readability. XML *tags* are identified by opening and closing angle brackets. A closing tag is identified by a forward slash preceding its name. An opening and closing tag form an *element*. To aid in readability, we boldface the tags in this write-up.

The second line indicates that this document is of the type as specified in the *course_description.dtd* document. The third line asks web-browsers to use the *course_description.xsl* document to render this document. The contents proper is nested inside of the **course_description** element. There, we find information on the course **id**, its **title**, number of **credits**, and a brief catalog course **description**.

```
<?xml version="1.0"?>
<!DOCTYPE course_description SYSTEM "course_description.dtd">
<?xml-stylesheet type="text/xsl" href="course_description.xsl"?>

<course_description>
      <id>
            CSSE100
      </id>
      <title>
            Introduction to Programming and Problem Solving
      </title>
      <credits>
            2
      </credits>
      <description>
            An introduction to general methods of problem
            solving, structured algorithm design, object-oriented
            techniques, and elementary computer programming.
      </description>
</course_description>
```

Figure 1: XML document for a sample course description

Element names are chosen by the author of the DTD. A web-author has the option of using an existing DTD or to design their own. A key principle of XML is that element names are used to indicate kinds of contents. Notice that there is no formatting information present. This is in contrast to HTML, where tags are used to format contents. This is the primary difference between XML and HTML. We will now explore the ramifications of this difference.

Since in XML, elements are used to indicate kinds of contents, content providers do not have to be concerned about formatting. This of course greatly simplifies the life of the typical academic web-author, who lacks time to memorize HTML formatting tags and who oftentimes spends considerable effort fiddling with HTML tags to obtain a halfway decent looking web-page. While authors who prefer to edit their documents with plain text editors still have to memorize XML elements, we will show in the section on "Ease of Editing" how even this can be circumvented.

In order to render an XML document, XML elements have to be translated into HTML formatting tags. We decided to use XSL over CSS, because of its wider range of transformations. XSL stylesheets are rather complex and can be lengthy. We will not present a stylesheet here, but refer the interest reader to (Wollowski 2002) which contains the stylesheet used to render the XML document from figure 1. The result is displayed in figure 2.

XSL stylesheets are complex. They should be edited by someone familiar with them. Alternatively, it may be advisable for contents providers to adopt a DTD and stylesheet from other organizations rather than editing their own.

There are several benefits to a well-designed stylesheet. (i) If all pages of a given kind reference the same stylesheet, then they all appear the same. This greatly aids users in finding information across courses. (iii) If changes to the appearance of course pages have to be made, only one document has to be edited. (iii) Content providers only have to reference a stylesheet. This leaves the worry over appearance to specialist versed in web-design. Since every HTML based web-page has to be concerned with form in addition to contents, not having to worry about appearance offers a significant time savings.

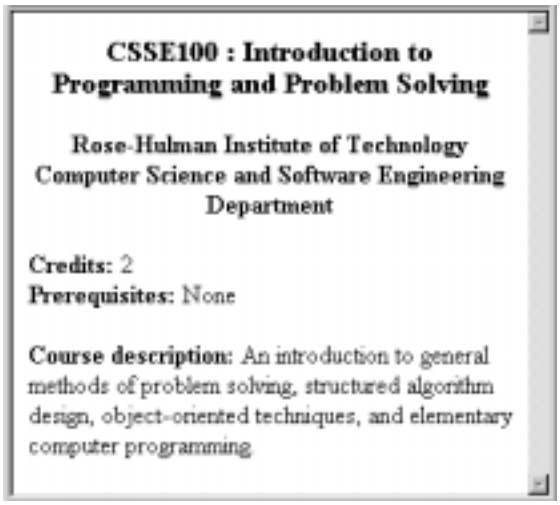

Figure 2: Sample XML document as rendered by our stylesheet

A further benefit of XML is the ability to take information common to documents and place it into a separate file. For example, the document of figure 2 contains information about the department and institution. Notice that they are not part of the file displayed in figure 1. Since this information is common to all course description pages, we placed it into a separate file which is displayed in figure 3.

```
<?xml version="1.0"?>
<!DOCTYPE course_description SYSTEM "course_description.dtd">

<departmental_information>
	<institution>
		Rose-Hulman Institute of Technology
	</institution>
	<department>
		Computer Science and Software Engineering Department
	</department>
	<term>
		Fall
	</term>
	<year>
		2002
	</year>
</departmental_information>
```

Figure 3: XML document containing departmental information

In our experience, one of the most time consuming and menial tasks is the updating of web-pages between terms as this requires changing term specific information. By referencing files with common information, only one (or few) documents have to be changed, with obvious benefits for sanity and consistency.

Notice that the file from figure 3 contains information that is not displayed on the course description page. The additional information is used for syllabi pages.

An often overlooked aspect of XML are Document Type Definitions (DTDs). DTDs serve to specify XML documents. As such they can be regarded as specifying classes of XML documents. A contents provider should consult the DTD for the kind of document that they plan to author, in order to learn about the available element names, the order in which they should appear in the document, and whether they are necessary or not. XML documents are currently not validated against their associated DTDs, hence a lot of authors do not create DTDs. However, there is an important reason why one should prepare DTDs. Stylesheets are used to render XML documents. In order to properly render an XML document, the stylesheet should have formatting directives for all elements to be expected in an XML document. The author of a stylesheet should consult the DTD for the type of document they wish to render. As such, a DTD serves as much to document a stylesheet as it serves to document an XML page.

Another important reason for providing DTDs, is that they enable the automatic generation of XML editors, a point that will be revisited in the following section.

Ease of Editing

There are several ways in which content providers may edit XML documents. The most straightforward way is by using a regular text editor. In this case, authors have to memorize element names as well as the structure imposed by the DTD, or they have to have the DTD handy. Due to the semantic nature of the elements, memorizing element names is not too much of a chore.

For web authors who prefer the use of text editors, we provided a template for a sample course document. Naturally, a document of an existing course could be copied, but it may not contain all elements.

For web authors who are not willing to use text editors, we provide a forms based web-page which prompts users for information to be placed into given elements. Figure 4 contains a screenshot of the editor for our course description pages. When using this editor, authors do not have to look up the DTD or memorize element names. We set up the editor so that certain elements come with default information which can either be accepted or edited. An example of such default information is the prefix for course numbers. The editor, which is password protected, can be used to generate new documents or to modify existing documents.

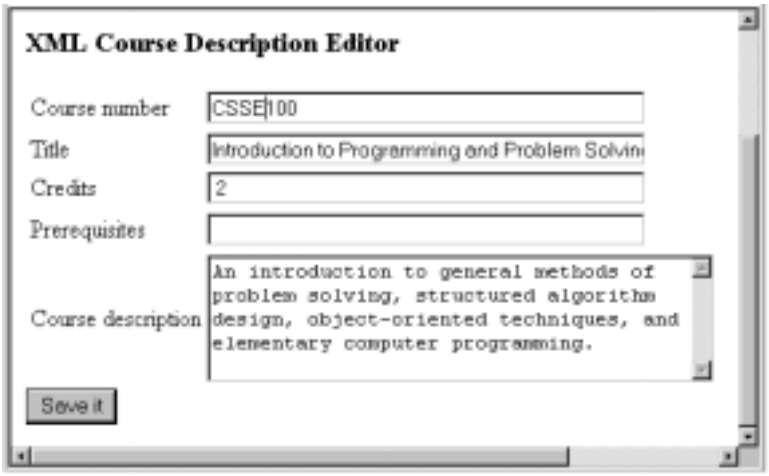

Figure 4: Interface to editor for course description pages

Since XML documents are only concerned about contents, it is feasible to provide a form field for each XML element. Right now, the fields are hard coded into the editor. We are currently working on ways to automate the editor so that it dynamically generates form fields based on a DTD.

One of the concerns when using XML are last minute additions to a document which fall outside of any given XML element. In general, the contents are rendered. However, there will likely not be any formatting. In order to format such additions, the stylesheet would need to be modified or expanded, with appropriate modifications of the corresponding DTD. This is a very time consuming effort and such

changes should probably be made only in consultation with the department. To enable authors to make last minute additions to an XML page, we provide them with the ability to use regular HTML tags.

In XML documents, each element must be closed by its corresponding tag. This means that when including HTML tags into an XML document, matching end tags must be present. As such, HTML code has to conform to the original specification of HTML, something that current browsers do not enforce.

If authors follow this simple rule, our stylesheet can render inline HTML code properly. We edited a stylesheet that can render HTML documents, and reference it from the stylesheet for the course description pages. This is another case where XSL shines. With XSL, one can include additional stylesheets into a particular stylesheet.

Another benefit of XSL is that stylesheets can be set up to process missing information. For example, the XML document of figure 1 does not contain information on the prerequisites for that course, yet the web-page from figure 2 states that there are none. This information was purposefully placed there by the stylesheet. Knowing about this ability of a particular stylesheet, web authors can further reduce their work.

Pinpoint Searching

A major benefit of XML is the possibility for vastly improved web-search. Due to the semantic nature of XML elements, search can be restricted to elements, thereby pinpointing searches. XML searching is an acknowledged research problem and we are only beginning to explore the exact benefits of searching XML files. Figure 5 contains a screenshot of the search engine for our course description pages.

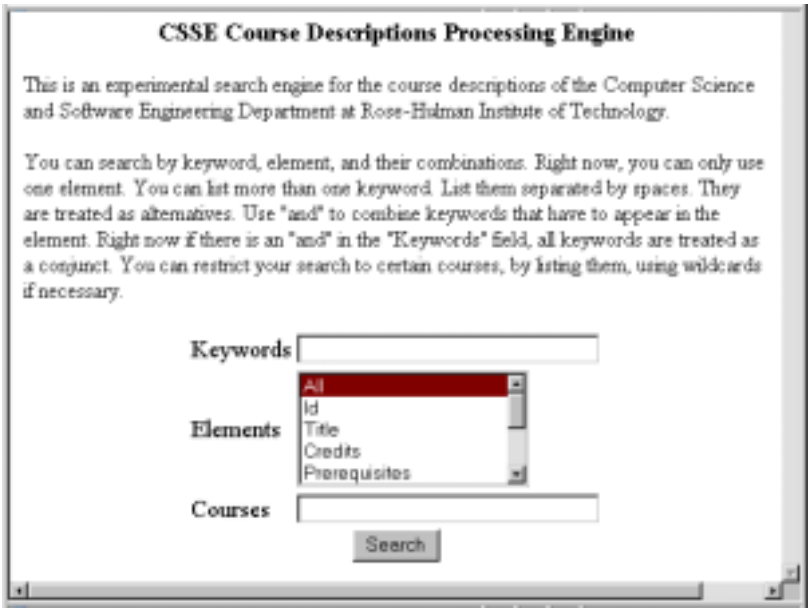

Figure 5: Interface to search engine for course description pages

There are a variety of searches that can be performed with it: (i) One can search for one or more keywords inside of one or more selected elements. The keywords can be treated as a conjunct by including the keyword “and” in the search term. Alternatively the keywords are treated as a disjunct. (ii) One can search for keywords in entire documents. (iii) One can browse pages by element, by leaving out keywords. In this latter case, information of the selected element is displayed for each of the documents accessible to the search engine. In the “courses” textbox, one can restrict the search to the documents specified.

The type of search engine made possible by XML is advantageous for many kinds of users. Consider searching the course description pages. If a student has taken a certain course, they can search the prerequisites fields with that course’s number to see which other courses they can now take. Prospective students may search course descriptions for reference to certain programming languages.

Given information on which courses a students has taken, prospective employers can search syllabi for a whole host of interesting information. For example, they may be interested in the programming languages to which a student was exposed, or the kinds of projects in which they participated, or the kinds of books they were supposed to study. As such, XML is more than a search engine. We like to refer to XML based search engines as *information processing engines.*

Our search engine is custom tailored to the XML document (as specified in the DTD.) Next to the editor, a customized search engine is the second piece of software that can be automatically generated from a DTD. At this point, our search engine is very primitive. There are many open issue that need to be addressed. We are working on some of them.

Multiple formats

The existence of a single stylesheet for documents of a given type ensures a uniform appearance of those documents. However, sometimes instructors prefer a certain appearance over another. For example, some instructors prefer a navigation bar at the top of a web-site, while others prefer to locate it at the left-hand side. While violating the principle of uniformity, those wishes can be adapted by providing stylesheets which implement those ideas. To go one step further, we are currently working on a way to enable students to select the stylesheet that most suits their web-surfing habits.

Conclusions

Using a sample course description, we have shown how to harness the power of XML to provide tools for convenient editing and maintenance of course description pages. Due to the separation of form and contents in XML, it is very natural to obtain a uniform appearance of departmental course pages. Last but not least, we have shown how to easily generate precision search engines, which are useful for current and prospective students as well as prospective employers.

During the presentation we will demonstrate the software for editing and searching syllabi. We will also report on faculty feedback when it comes to editing pages and student feedback in regards to ease of search. At this point, we want to gain more experience with our chosen set-up. In order to be scaleable, further work has to be done on the design of the search engine. We are currently working on this problem and will give a status report at the meeting.

Realistically, XML presents an exciting avenue for those departments that have not moved to courseware products. However, even departments that have adopted courseware products way wish to investigate the benefits of XML technology. The current lack of integration with systems such as Banner does not pose a problem, as course pages are fairly independent of information stored in Banner type systems. In our department, we have several pieces of software aimed at providing some of the functionality provided by courseware sites, giving us the power to tailor them to our needs.

For those interested in learning more about XML, I found the following books very useful (Harold & Means 2001, Williamson 2001).

References

Blackboard (2002). http://www.blackboard.com/

Goldfarb, C.F., Prescod, P. (2000). *The XML Handbook*, 2nd Ed. Prentice Hall.

Harold, E.R., Means, W.S. (2001). *XML in a Nutshell*. O'Reilly.

WebCT (2002). http://www.webct.com/

Williamson, H. (2001). *XML: The Complete Reference*. Osborne.

Wollowski, M. (2002). http://www.cs.rose-hulman.edu/~wollowsk/xml/dept/course_description.xsl