XML WEB TECHNOLOGIES

Chakib Chraibi, Barry University, cchraibi@mail.barry.edu

ABSTRACT

The Extensible Markup Language (XML) provides a simple, extendable, well-structured, platform independent and easily searchable means for data description, representation, exchange and integration. Although XML uses markup tags as HTML, it derives its power from its ability to allow users to define new tags related to the content, separate data and the presentation of data, and create self-describing hierarchical structures. XML and its associated technologies offer a powerful framework for organizing data and automating the exchange of information. It also facilitates the integration and interoperability of Web applications. This paper describes the main XML technologies and tools that make XML an essential instrument for data representation and exchange. It also shows that, thanks to its association with other portable technologies such as Java, XML is becoming the main engine of data exchange and management.

Keywords: XML, Web applications and services, XML tools, XML Programming, Java, HTML.

INTRODUCTION

The remarkable growth of the Word Wide Web has led to the creation of an enormous amount of information that needs to be processed, exchanged, and transferred. It has also spawned the development of sophisticated and powerful Web applications and services. The keys of this success are cheap means for worldwide communication, distribution, and publication. However, the increase in complexity has revealed the limitations of current technologies and the need for standards for data representation and transfer. There is a profound need for a powerful platform for data representation, storage, modeling, and exchange that has a high level of interoperability.

HTML (HyperText Markup Language), which now stands as the main means for Web publication, uses a fixed number of predefined tags that are unable to withstand high-level publications or interchange. EDI (Electronic Data Interchange), which is currently the standard method of information transfer in the business world, is expensive to purchase and implement. XML is a cheap, simple, portable, powerful and extensible technology for data storage, integration and interoperability both within and between organizations. It is fast becoming the standard for data interchange on the Web.

XML, developed by the World Wide Web Consortium (W3C), is a set of standards to exchange and publish information in a structured manner. As in HTML, an XML document is a text document consisting of character data and markup. However, the markup is concerned with the description and the structure of the data, rather than the browser display information. Furthermore, the markup is not defined, but rather has to be created, or extended. XML also comes with a set of technologies covering different aspects of Web applications. These include the description of the document structure using a Document Type Definition (DTD) or an XML schema, its presentation using CSS (Cascading Style Sheets) or transformation using XSL-T

(Extensible Stylesheet Language Transformations). Interface with other applications can be carried out using the SAX and DOM Application Program Interfaces.

In this paper, we aim to provide an overview of XML and its family of tools and technologies, including some examples. We also explain why XML offers a structured and solid way to describe and transfer data. First, we introduce XML and its extendable tags. Then, we describe the different processes for developing XML applications and analyze the possible impact of XML on Web applications and services.

XML EXAMPLE

XML is a less complex subset of the Standard Generalized markup Language (SGML), which was developed by the World Wide Web Consortium (W3C) to express structure and content in different types of electronic documents (1, 2). It offers a structured and consistent way to describe and transfer data. As in HTML, the structure in XML is built up using markup tags. There is however a very important difference between tags in HTML and tags in XML; unlike HTML tags, XML tags have no predefined meaning. In addition XML is extensible, structured, and can be validated.

The main characteristic of XML is data independence. XML separates content and presentation, so it can conceivably be processed by any application. This means that the same XML document can be displayed in a number of ways depending on the media of the presentation without changing the underlying structure of the data. Furthermore, XML documents are basically text-based, human-readable documents. This makes XML an ideal framework for data exchange as it simplifies business-to-business transactions. Two or more systems can send and receive XML-tagged data without having to know the other's system organization, as long as each system follows the same document rules and structure.

Let's now briefly illustrate the basic difference between HTML and XML. While HTML tags specify how a Web browser should display text and other multimedia contents on a page, XML tags contain information about the structure and content of the document. The appearance of the data is left for the application reading the document to decide. Furthermore, while HTML is "flat", XML can define a hierarchy. XML is also case sensitive, self-describing and can be validated. Some of these differences are illustrated in Table 1. The table shows two documents, the left one is written in HTML and the right one in XML. The first statement in each is called a declaration and identifies each document as either an HTML or an XML document.

In XML, each start-tag/end-tag pair, with the data that lies between them, is an element. Every XML document consists of elements organized in a logical tree structure. There is always one element that contains all the other elements; this element is called the root element (e.g., <courses>). Elements can also have other information attached to them called attributes. Attributes describe properties of elements. The values stored within the elements can basically be of three different types, parsed character data (PCDATA), unparsed character data (CDATA), or processing instructions (PI). Finally, an XML document can be broken up into many files on a hard disk or objects in a database and each of them is called an entity in XML terminology. Entities can even be spread across the Internet. Namespaces allow differentiating vocabularies.

<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN"> <HTML> <HEAD> <TITLE>XML</TITLE> </HEAD> <BODY> <H1>XML Web Technologies</H1> <H3>CS 456</H3> <H4>Instructors</H4> Chakib Chraibi Lauren Hernandez <P>Semester: Spring 2003 – Room: Garner 106</P> <P>This course provides a study of XML and its impact on Web technologies.</P> </BODY> </HTML>	<?xml version="1.0"?> <courses> <course> <title>XML Web Technologies</title> <code>CS 456</code> <instructors> <instructor_name>Chakib Chraibi</instructor_name> <instructor_name>Lauren Hernandez</instructor_name> </instructors> <GeneralInformation semester="Spring 2003" room="Garner 106"/> <description> This course provides a study of XML and its impact on Web technologies.</description> </course> </courses>

Table 1. Examples of an HTML and an XML document.

XML VALIDATION

A particular XML format can be defined in a Document Type Declaration (DTD) or in an XML schema. This enables any application to understand an XML document and check if it complies with its definition and structure. This process is referred to as the validation process. In other words, a DTD needs to lay the structure of the document markup, i.e., the DTD defines what markup tags can be used in the document, what order they can appear in, what other tags they can contain and so on. XML schemas, much like DTDs allow the description of the structure of an XML document. However, XML Schemas use XML syntax. Table 2 shows examples of a DTD and an XML Schema developed for the XML document shown in Table 1.

<!ELEMENT courses (course+)> <!ELEMENT course (title,code,instructors,GeneralInformation,description)> <!ELEMENT title (#PCDATA)> <!ELEMENT code (#PCDATA)> <!ELEMENT instructors (instructor_name+)> <!ELEMENT instructor_name (#PCDATA)> <!ELEMENT GeneralInformation EMPTY> <!ATTLIST GeneralInformation semester CDATA #IMPLIED room CDATA #IMPLIED> <!ELEMENT description (#PCDATA)>	<?xml version=”1.0”?> <schema xmlns ="http://www.w3.org/2001/XMLSchema"> <element name="courses"> <complexType> <sequence> <element name="course"> <complexType> <sequence> <element name="title" type="string/"> <element name="code" type="string/"> <element name="instructors"><complexType> <sequence maxOccurs="unbounded"> <element name="instructor_name" type="string/"></sequence></complexType> <element name="GeneralInformation"> <attribute name="semester" type="string/"> <attribute name="room" type="string/"> <element name="description" type="string/"> </sequence></complexType></element> </sequence></complexType></element> </schema>

Table 2. DTD and XML Schema

The DTD uses a special syntax to declare every object (elements, attributes, and so on) that can appear in an XML document. To link the XML document to a particular DTD, a document type declaration may be added after the XML declaration as follows:

```
<!DOCTYPE courses SYSTEM "course-dtd.dtd">
```

where "course-dtd.dtd" is a text file where the DTD is stored. Note that the root has to be specified in the declaration.

The XML schema uses an XML syntax and vocabulary to describe the structure of the document and its elements. To link the XML document with an XML Schema, the following declaration must be added after the XML declaration:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
                                    noNameSpaceSchemaLocation = "course-xsd.xsd">
```

where "course-xsd.dtd" is a text file where the XML Schema is stored.

XML PRESENTATION AND TRANSFORMATION

An XML document has no style characteristics because of the separation of data and presentation. That is one of its greatest strengths since it means that the content of an XML can be delivered in many different formats, but to deliver it, it has to be styled. Two styling tools may be used with XML, namely, Cascading Style Sheet (CSS) or Extensible Style Sheet (XSL). Table 3 shows examples of style sheets with CSS (on the left) and XSL.

```
title       {
            display: block;
            font-weight: bold;
            font-size: 20pt;
            text-align: center;
            }
code {
            display: block;
            font-weight: bold;
            font-size: 20pt;
            text-align: left;
            }
instructors {
            display: block;
            font-weight: bold;
            font-size: 14pt;
            text-align: left;
            }
schedule {
            display: block;
            font-size: 12pt;
            text-align: left;
            }
description {
            display: block;
            font-size: 10pt;
            text-align: left;
            }
```

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>
<xsl:template match="/">
<html>
<head><title>
<xsl:value-of select="courses/course/title"/>
</title></head>
 <xsl:apply-templates/>
</html>
</xsl:template>
<xsl:template match="courses">
<body><xsl:apply-templates/></body>
</xsl:template>
<xsl:template match="title">
<h1><xsl:apply-templates/></h1></xsl:template>
<xsl:template match="code">
<h1><xsl:apply-templates/></h1></xsl:template>
<xsl:template match="instructor_name">
<h2><xsl:apply-templates/></h2>
</xsl:template>
<xsl:template match="schedule">
<h3><xsl:value-of select="@semester"/></h3>
<h3><xsl:value-of select="@room"/></h3>
</xsl:template>
<xsl:template match="description">
<p><xsl:apply-templates/></p>
</xsl:template>
</xsl:stylesheet>
```

Table 3. CSS and XSL Stylesheets

In CSS, a stylesheet is basically a set of rules that describes which styles to use to display the text. CSS is not as powerful as XSL, but it is very similar to HTML styling directives. In the example, the CSS stylesheet can be stored in a separate file, say "course-css.css", and linked to the XML document by adding the following statement after the XML declaration: <?xml-stylesheet href="course-css.css" type="text/css"?>.

XSL (eXtensible Stylesheet Language) includes XSLT or XSL Transformations, a scripting language specifically designed to manipulate XML documents. XSLT can transform XML documents into any text-based format, XML or otherwise. The stylesheet shown in Table 3 can be saved in a file entitled "course-xsl.xsl". To carry out the transformation, an XSLT processor such as Xalan, Saxon, or MSXSL is needed. An XSLT processor applies an XSLT stylesheet to an XML document. In this example, the XML document is transformed into a simple HTML file for presentation.

XML PROGRAMMING

For XML data processing, two interfaces were developed, namely SAX and DOM. At the heart of both models is an XML parser that processes the XML document, so that the document elements can be retrieved and transformed into data that can be understood by the application and task in hand. SAX stands for simple Application Programming Interface (API) for XML. In SAX, the XML parser generates events (e.g., a start tag, end tag, attribute, etc.) as the file is read. In the DOM (Document Object Model) model, the XML parser builds a tree of objects that contains all the elements in the XML document. The XML parser also checks the syntax and structure (i.e., validity and well-formedness) of the document.

The SAX and DOM APIs basically allow data in an XML document to be accessed and modified by programs written for example in JavaScript or Java. XML and Java are well suited for each other. Both of them are portable. XML data is platform-independent and Java processing is also platform-independent. Both are extensible, XML through its XML Schemas or DTDs and Java through its class loaders. XML is a technology for describing data, which in turn needs applications to process this data. Java is the so far the preferred language for server and client-side XML application development. Two APIs have been developed for reading, writing, and manipulating Java code from within Java, namely the Java DOM (JDOM) and the Java API for XML parsing (JAXP).

XML APPLICATIONS

XML applications can be broadly divided into two categories: document applications and data applications. Document applications are intended for publication and B2C communications while data applications are intended for data interchange and B2B communications. It is important to note that for both applications, the same standard and tools are used. This allows the reuse of all XML elements across applications.

Document applications are developed for Web publishing. The main advantage is that an XML document has no style characteristics because of the separation of data and presentation. That is one of the main characteristics of XML since it means that the content of an XML can be delivered in many different formats and on many different devices, as Figure 1 shows. Before XML, in order to show a document on different devices, multiple versions of the document must

be developed. This leads to a tight coupling to the device, which means a major development to adapt to a new device. With XML, a new device means a new layer, not a rewriting of the application. The delivery to the market is then quicker.

Document applications have to be styled before been delivered. As we have seen, two styling tools may be used with XML, namely, CSS or XSL. This allows different renderings of the same document and can be defined as the concept of "write once, publish many". This might address one of the fundamental issues of Web design, which is audience targeting. Different stylesheets may be developed to address different audiences and strategies, after an initial screening.

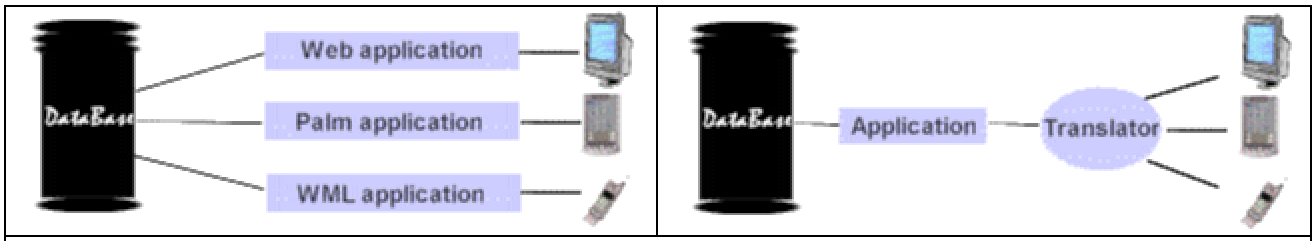

Figure 1. Handling of Web applications before and after XML.

Data applications of XML include B2B Web applications such as e-commerce, supply-chain management, and application integration. The structure of a document can be expressed in XML, so can the structure of a database. In this context, XML is used to exchange information between businesses. Because of XML simple text-based syntax and its independence from proprietary formatting, it is bound to play a prominent role in data interchange as it simplifies it and lowers its cost. Figure 2 shows how XML can be used for both documents and databases (1).

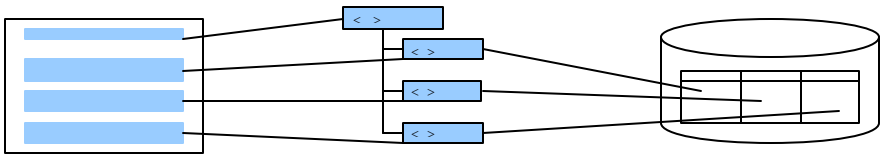

Figure 2. Document and database structures in XML

XML text-based structure allows an easy sharing of the data within the organization or between suppliers and customers without integrating the back end systems. Appropriate vocabularies for any specific field can be created to accommodate existing business structures and methods. For the e-business, XML provides an open, cross-platform way for B2B transactions and management. Since XML is a metalanguage, this may lead to a proliferation of vocabularies that may hurt interoperability and increase cost and complexity. For this purpose, there have been several initiatives to introduce public repositories of XML message definitions as an attempt to ease any translation problem. These include ebXML and BizTalk.

EbXML is a framework for electronic trading developed by industry-led consortia OASIS and UN/CEFACT. It consists of a set of specifications that together enable a modular electronic business framework, which should be accessible to all businesses regardless of their size. The vision of ebXML is indeed to enable a global electronic marketplace where enterprises of any size and any geographical location can meet and conduct business with each other through the exchange of XML-based messages. ebXML is rapidly gaining momentum, and many leading software developers are currently working on implementations of each of the components. Developing with ebXML requires a sound knowledge of XML, XML technologies, and the business framework and processes the company is involved with.

BizTalk is a cross-platform framework based on XML that was developed by Microsoft to describe business processes and integrate software applications. Microsoft is working with a number of partners to define BizTalk schemas. Microsoft and SAP have begun defining schemas for exchanging both product catalogue information and business documents between companies. The BizTalk framework specifies a set of guidelines for implementing an XML schema and a set of tags used in messages sent between applications. BizTalk will be supported by Microsoft through its .NET servers, and the next version of SQL Server will include native support for BizTalk through its support for XML documents.

CONCLUSION

XML is destined to play an outstanding role in Web publishing, data interchange and B2B communications, as every major application is bound to be at least partially, if not totally, web-based. XML strengths include ease of use, cross-platform standard, and data description power. XML provides many advantages to businesses, including standardization, simplicity, portability, extensibility, easy manageability, and longevity (3). In a sense, XML has raised the Internet/Web to another level. XML comes with a family of technologies that help provide a simple, flexible, and powerful framework for organizing data and automating the exchange of information. We have presented only few of them in this paper. HTML and XML can complement each other. HTML can still be used for display, while XML's main use is data representation and exchange. Furthermore, XML complemented with Java allows data exchange and processing across systems as well as across businesses. The development of XML vocabularies speeds up the convergence to XML but a coordinated effort between industries and organizations is a must for the success of XML.

REFERENCES

1. Benoit, M. (2002). XML by Example. Indianapolis, Indiana: QUE.

2. Hunter, D. and al. (2001). Beginning XML. Birmingham, UK: WROX.

3. Kamthan, P. and Hsueh-leng P. (2000). Perspectives of XML in E-Commerce. *http://tech.irt.org/articles/js215/*.