Programming Languages & Translators

(COMS W4115)

Department of Computer Science

Columbia University

Summer 2007

XML Document Manipulation Language (XDML)

White Paper

Luba Leyzerenok
ll2310@columbia.edu
June 11, 2007

1. Abstract

XDML is a simple, high-level language for XML documents manipulation and analysis. This paper will discuss the use of the language, features that I intend to implement and its syntax. I will provide a sample XDML program in the last section.

2. Introduction

XML (the Extensible Markup Language) is a W3C-endorsed standard markup language for documents containing structured information. Its main purpose is to facilitate the sharing of data across different information systems, particularly via the Internet.
In the last few years, XML has been adopted in fields as diverse as law, aeronautics, finance, insurance, robotics, multimedia, hospitality, travel, art, construction, telecommunications, software, agriculture, physics, etc. XML has become the syntax of choice for newly designed document formats across almost all computer applications. It's used on various OS, including Linux, Windows, Mac OS, and many others.
With such wide use of XML, there is obviously a need for a language/technology that lets users manipulate XML documents. There is a wide range of technologies and applications available for programmers' use with XML. XDML is a simple, high-level language for XML document manipulation that can be used by people in difference fields, who do not have any prior experience with computer programming.
XDML will be written entirely in Java, with use of existing Java APIs, such as DOM, SAX and JAXP. The compiler will convert the XDML source code into Java code that will be compiled by a Java compiler and executed by Java virtual machine.

3. Functionality

XDML provides an easy way for users extract data from XML objects. I intend to implement the following functionality:

- Count elements: given the path to an element, such as "Class/Student" for example, the count operator will return a number of Student elements in the Class element.
- Print – this function will print the value given the path to an element.

4. Language Features

- *Simple* – The language targets audience are non-programmers from different industries. Therefore, people without programming experience should be able to read and understand the intent of the code.
- *Intuitive* – Keywords and syntax are carefully chosen to imitate natural for users (English) language. Reading the code aloud should resemble the instructions dictated by a person to another person.
- *Portability* – Since the XDML source code is translated into Java code and eventually into Java byte code and executed in Java virtual machine, XDML programs can be executed on any platform and that has Java virtual machine.

5. Syntax

5.1 Data types

The following basic data types will be available in XDML:

- *Integer* **–** will hold an integer value and identified by an **`int`** keyword.
- *String* **–** will hold one or more characters and identified by a **`string`** keyword.
- *Element* **–** will hold a reference to an XML element and identified by an **`element`**
- *Boolean* – will be available to hold 0/1 values and will be identified by **`boolean`** keyword.

5.2 Programming Constructs

Loops and flow control structures will be available and have similar syntax to other high-level languages, in particular, Java.

6 XDML Program Example

```
start ()
{

element class = “<Class>
                    <Student>
                            <FirstName>Jane</FirstName>
                            <LastName>Miller</LastName>
                            <SSN>987-65-4321</SSN>
                            <Phone>112-112-1212</Phone>
                    </Student>
            </Class>”;


element student1 =  “<Student>
                            <FirstName>John</FirstName>
                            <LastName>Smith</LastName>
                            <SSN>123-45-6789</SSN>
                            <Phone>112-112-1212</Phone>
                            <Phone>334-334-3434</Phone>
                    </Student>”;

// returns 1
int students =class.count (“Class/Student”);

//prints a Student element
class.print (“Class/Student”);

// prints Phone 112-112-1212 and phone 334-334-3434.
student1.print (“Student/Phone”);

}
```