Comparing Document Object Model (DOM) and Simple API for XML (SAX) in Processing XML Document in Leave Application System

Juliana Wahid

College of Arts and Sciences, Information Technology Building,
Universiti Utara Malaysia, 06010 Sintok, Kedah
Tel : 04-9284715, Fax : 04-9284753
E-mail : w.juliana@uum.edu.my

ABSTRACT

The Extensible Markup Language (XML) consists of two Application Programming Interface (API) i.e. Document Object Model (DOM) and Simple Application of XML (SAX). Previous research has used the DOM API in the Leave Application prototype. In this research, the DOM API based code that developed in the prototype will be changed to SAX API based code. The performance measurement then is carried out to evaluate the used of DOM API and SAX API in the prototype. The evaluation process took place within in each API and also combination of both APIs. From the evaluation process, it was concluded that the use of combination of SAX based code in Domino Workflow environment and DOM based code in Microsoft Exchange 2000 Server produces the lowest processing time that will increase the performance of the leave Application system.

Keywords

Extensible Markup Language (XML), Document Object Model (DOM), Simple Application of XML (SAX), Performance Measurement

1.0 INTRODUCTION

Previous research by Juliana (2003) has shown that interoperability between Domino Workflow and Microsoft Exchange 2000 Server can be achieved using *Microsoft Exchange Connector for Lotus Notes* and *Document Object Model (DOM) of Extensible Markup Language (XML).*

As stated by Laddad (2000) and [9], Extensible Markup Language (XML) standard consists of two kinds of Application Programming Interfaces (APIs), i.e. Document Object Model (DOM) and Simple Application XML (SAX). Juliana (2003) had carried out her research using DOM because she noticed that DOM is more likely to be used in information exchanging and there are more examples of programming from various languages such as Java, VBScript and LotusScript on DOM.

The used of DOM as stated by Hunter et al. (2000) enables programmers to create documents and parts of documents, navigate through the document, move, copy, and remove parts of the document, add or modify attributes. However, these document manipulations might have a high price in terms of flexibility and resource consumption (Brownell, 2002). The used of SAX can perhaps minimizes time consuming in the prototype system developed previously using DOM because as stated by Bourret (2000) and Birbeck et al, (2001) the SAX-based code uses much less memory where it buffers only one row of data at a time while the DOM-based code buffers the entire document. Furthermore, the SAX-based code is faster because it does not have to spend time building a DOM tree.

The main objective of this study is to process the XML document in the prototype system developed previously by Juliana (2003) using SAX API. Then, a performance evaluation between the same prototype systems, i.e. Leave Application System with different an XML API will be carried out.

2.0 LITERATURE REVIEW

2.1 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a new standard that was produced by World Wide Web Consortium (W3C) in late 1998. It is set of syntax rules and guidelines for defining text-based markup languages. XML languages have a number of uses including exchanging information, defining document types and specifying messages.

According to Sills et al. (2002), in an XML document, the data are stored in a hierarchical fashion as shown in Figure 1.

Figure 1: Staff.xml

2.2 Document Object Model (DOM)

The XML Document Object Model (DOM) is a programming interface for XML documents. It defines the way an XML document can be accessed and manipulated. As a W3C specification, the objective for the XML DOM has been to provide a standard programming interface to a wide variety of applications. The XML DOM is designed to be used with any programming language and any operating system. With the XML DOM, a programmer can create an XML document, navigate its structure, and add, modify, or delete its elements.

A DOM implementation presents an XML document as a tree structure, or allows client code to build such a structure from scratch. It then gives access to the structure through a set of objects, which provided well-known interfaces. Figure 2 shows some XML text and its transformation to tree structure.
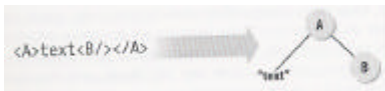

Figure 2: XML tree structure (Source: Brownell, 2002)

In order to extract the XML document to tree structure, XML parser will be used. Example of XML parser is MSXML; which is built-in parser in Internet Explorer. Once we have a DOM tree or document object, we can access the parts of our XML document through its properties and methods.

2.3 Simple API for XML (SAX)

SAX is a product of collaboration on the XML-DEV mailing list. According Birbeck et al. (2001), SAX uses a parser that reads from the input XML document and notifies the application of interesting events as shown in Figure 3. A SAX parser uses some predefined callbacks to notify an application of parsing events. These callbacks are defined as methods of several standard SAX interfaces to be implemented by the application.
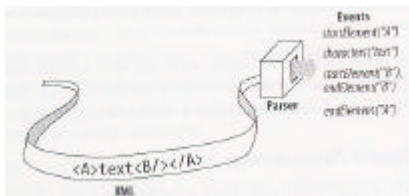

Figure 3: XML events structure (Source: Brownell, 2002)

2.4 DOM versus SAX

In reviewing both of DOM and SAX, diagrammatically, DOM as shown in Figure 4 will parse the document into the DOM tree and then use the DOM to navigate around the document.
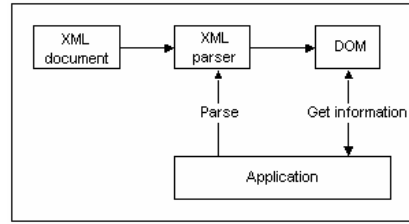

Figure 4: DOM framework (Source: Hunter et al, 2000)

Whereas the SAX approach as shown in Figure 5 tell the parser to raise events whenever it find something (data).
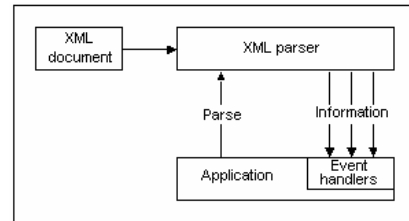

Figure 5: SAX framework (Source: Hunter et al, 2000)

If our XML document were 20 mega bytes large, it would be very inefficient to construct and traverse an in-memory parse tree just to locate one piece of contextual information. In the other hand, an event-based interface; which is SAX would allow us to find it in a single pass using very little memory. The using of DOM will not only take up space (memory) but also time in building the tree. However, as SAX only allows a view of one bit of the document at a time, it is extremely useful using DOM for random-access applications.

As objectives of this research is to minimize the time consumed in the interoperability between Domino Workflow and Microsoft Exchange 2000 Server in Leave Application prototype, the advantages of SAX is most probably highlighted. According to Hunter et al. (2000), Birbeck et al, (2001), Brownell, (2002), and Sills et al, (2002) SAX parsing is faster than DOM parsing. From this point of view the next chapter will describe the code changing process from DOM code-based to SAX code-based in the Leave Application prototype system to realize the evaluation of performance in both environments.

3.0 IMPLEMENTATION AND EVALUATION

This section will describe the performance measure of both DOM and SAX in terms of their responsiveness, which evaluate how quickly a given task of processing time i.e. XML document parsing and leave form creating, can be accomplished by both APIs. The measure was carried out in two category of measurement i.e. within each APIs and combination of APIs.

3.1 Within Each APIs

This measurement was carried out either using DOM based code or SAX based code in both Domino

Workflow and Microsoft Exchange 2000 Server environment of the Leave Application System.

3.2 Combination of APIs

This measurement was carried out using combination of DOM based code and SAX based code in both Domino Workflow and Microsoft Exchange 2000 Server environment of the Leave Application System.

Both Domino Workflow and Microsoft Exchange 2000 Server user were set up to be an applicant and approver of the Leave Application System. As stated by Joines, Willenborg, and Hygh (2003) same test must be run at least three times, the leave application process was carried out ten times for each setup mentioned above. For each leave application process that also involved the approval part, the processing time was obtained from time start and time end in event log file in both Domino Workflow and Microsoft Exchange 2000 Server environment.

According to Joines et al. (2003) the performance measure demands repeatable and reliable results. From this point of view this research will consider only repeatable results of processing time. If the times that captured are repeatable in more then one specific times then the average of the times will be calculated.

The processing time for leave application process with the first setup i.e. DOM based code environment is shown in Table 1.

Table 1: DOM based code environment processing time result

Iteration	Applicant Environmment	Approver Environmment	DOMINO Agent "import XML using DOM" Processing Time (Second) DOM based code	MICROSOFT EXCHANGE Agent "parsing xml document" Processing Time (Second) DOM based code	Total (Second)
1	Exchange	Domino	9	4	13
2	Domino	Exchange	4	4	8
3	Exchange	Domino	6	4	10
4	Domino	Exchange	5	3	8
5	Exchange	Domino	6	3	9
6	Domino	Exchange	4	3	7
7	Exchange	Domino	8	2	10
8	Domino	Exchange	6	4	10
9	Exchange	Domino	5	3	8
10	Domino	Exchange	6	2	8

The processing time for leave application process with the second setup i.e. SAX based code environment is shown in Table 2.

Table 2: SAX based code environment processing time result

Iteration	Applicant Environmment	Approver Environmment	DOMINO Agent "import XML using SAX" Processing Time (Second) SAX based code	MICROSOFT EXCHANGE Agent "parsing xml file" Processing Time (Second) SAX based code	Agent "create form" Processing Time (Second) SAX based code	Total (Second)
1	Exchange	Domino	16	2	3	21
2	Domino	Exchange	6	2	4	12
3	Exchange	Domino	5	2	3	10
4	Domino	Exchange	7	2	3	12
5	Exchange	Domino	12	5	3	20
6	Domino	Exchange	5	2	4	11
7	Exchange	Domino	9	2	3	14
8	Domino	Exchange	5	3	4	12
9	Exchange	Domino	4	5	5	14
10	Domino	Exchange	4	2	8	14

The processing time for leave application process with the third setup i.e. SAX based code Domino and DOM based code Microsoft Exchange 2000 Server environment is shown in Table 3.

Table 3: SAX based code domino and DOM based code exchange environment processing time result

Iteration	Applicant Environmment	Approver Environmment	DOMINO Agent "import XML using SAX" Processing Time (Second) SAX based code	MICROSOFT EXCHANGE Agent "parsing xml document" Processing Time (Second) DOM based code	Total (Second)
1	Exchange	Domino	5	3	8
2	Domino	Exchange	5	3	8
3	Exchange	Domino	9	3	12
4	Domino	Exchange	5	4	9
5	Exchange	Domino	6	3	9
6	Domino	Exchange	6	3	9
7	Exchange	Domino	8	4	12
8	Domino	Exchange	4	3	7
9	Exchange	Domino	6	3	9
10	Domino	Exchange	8	4	12

The processing time for leave application process with the forth setup i.e. DOM based code Domino and SAX based code Microsoft Exchange 2000 Server environment is shown in Table 4.

Table 4: DOM based code domino and SAX based code exchange environment processing time result

Iteration	Applicant Environmment	Approver Environmment	DOMINO Agent "import XML using DOM" Processing Time (Second) DOM based code	MICROSOFT EXCHANGE Agent "parsing xml file" Processing Time (Second) SAX based code	Agent "create form" Processing Time (Second) SAX based code	Total (Second)
1	Exchange	Domino	18	7	4	29
2	Domino	Exchange	5	3	7	15
3	Exchange	Domino	8	2	3	13
4	Domino	Exchange	2	2	3	7
5	Exchange	Domino	10	5	3	18
6	Domino	Exchange	5	1	4	10
7	Exchange	Domino	7	1	3	11
8	Domino	Exchange	6	2	4	12
9	Exchange	Domino	8	2	3	13
10	Domino	Exchange	4	2	3	9

From the above results, Table 5 summarizes the total processing time for four different setups mentioned above.

Table 5: Total processing time for each setup

Setup	Total Processing Time
DOM based code Environment	9.5
SAX based code Environment	10
SAX based code Domino and DOM based code Exchange Environment	8.5
DOM based code Domino and SAX based code Exchange Environment	11.5

4.0 RESULTS AND CONCLUSION

Result shown in Table 5 shows that combination of SAX based code in Domino Workflow environment and DOM based code in Microsoft Exchange 2000 Server produces the lowest processing time of Leave Application process.

The statement "SAX parsing is faster than DOM parsing", can be proven by observing the processing time

in Domino Workflow environment. Within APIs implementation that referring to Table 1 and Table 2, processing time for SAX based code is 5 second while processing time for DOM based code is 6 second. By referring Table 3 and 4 for combination of APIs implementation, processing time for SAX based code is 5.5 second while processing time for DOM based code is 6.5 second.

However in the Microsoft Exchange 2000 Server environment, DOM based code produces the lowest processing time then the SAX based code. This happens because the implementation of SAX based code in the Microsoft Exchange 2000 Server uses two public folders that drag the processing times.

5.0 ACKNOWLEDGMENT

REFERENCES

Birbeck, M., Duckett, J., Gudmundsson, O. G., Kobak, P., Lenz, E., Livingstone, S., Marcus, D., Mohr, S., Pinnock, J., Visco, K., Watt, A., Williams, K., Zaev, Z., and Ozu, N. (2001). **Professional XML 2nd Edition**. Wrox Press Ltd. United Kingdom.

Bourret, R. (2000). **Data Transfer Strategies**. Transferring data between XML documents and relational databases. Retrieved Jun 11, 2001 from the World Wide Web: http://www.rpbourret.com/xml/DataTransfer.htm

Brownell, D. (2002). **SAX2**. O'Reilly & Associates, Inc. United States of America.

Hunter, D., Cagle, C., Gibbons, D., Ozu, N., Pinnock, J., and Spencer, P. (2000). **Beginning XML**. Wrox Press Ltd. United Kingdom.

Joines, S., Willenborg, R., and Hygh, K. (2003). **Performance Analysis for Java Web Sites**. Pearson education, Inc. United State.

Juliana Wahid (2003). Establishing Effective Workflow Interoperability Framework in the case of Domino Workflow and Microsoft Exchange 2000 Server. **MSc(IT) UUM Thesis**.

Laddad, R. (2000). XML APIs For Databases. **Java World January 2000**. Retrieved November, 2001 from th eWorld Wide Web: http://www.javaworld.com/javaworld/jw-01-2000/jw-01-dbxml.html.

Sills, A., Ahmed, M., Boumphrey, F., and Ortiz, J. (2002). **XML.NET Developer's Guide**. Syngress Publishing, Inc. Canada.

Skonnard, A. (2000). The XML Files. **MSDN Magazine 2000**. Retrieved November, 2001 from the World Wide Web: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmag00/html/xml0500.asp