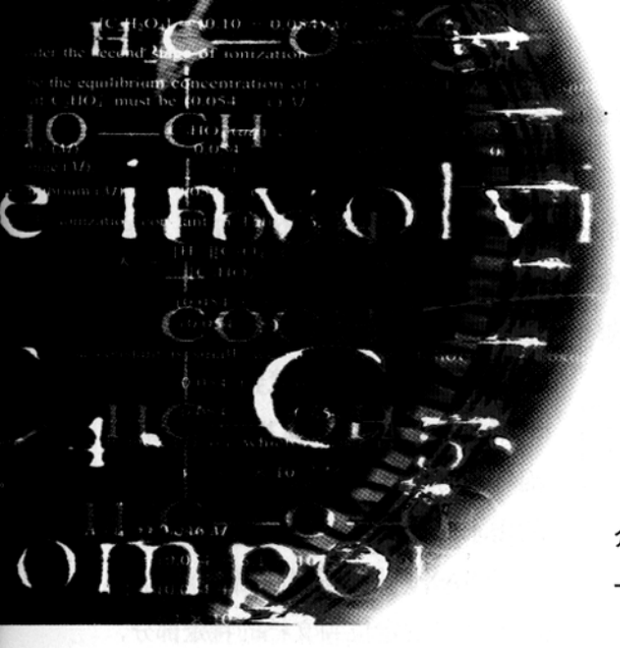

# XML解析器应用分析

徐 冰 李启炎 朱 茜 (上海同济大学52#信箱 200092)

**摘要：本文在简述XML一系列优越特性的基础上，结合当前研究热点，详细介绍了解析XML文档的两种方法：DOM和SAX，深入探讨了两者的利弊，并给出一个DOM解析器的实际应用。**

**关键词：** XML XSL DOM SAX

## 1 引言

XML是W3C制定的一组规范，用来定义文档标记语言的框架。它提供了一种描述结构数据的格式，使各种基于WEB的应用之间能方便的交换数据。XML以简单性、开放性、可扩展性、互操作性等一系列特性给基于Web的应用程序带来了强大的功能和灵活性。

XML不仅为三层 B/S/S 应用程序开发提供了所需的技术，也使本地化计算和操作成为可能。XML格式的数据递交给客户机后，可使用客户机应用程序的计算逻辑对该数据进行剖析，并在本地进行编辑和操作。XML另一特性是粒状更新，这极大的改善了服务器的可缩放性。粒状更新意味着当部分数据被更改后，无需再发送一个完整的结构化数据集，只有更改了的元素才从服务器发送到客户机上，并且无须刷新整个用户界面就可以显示更新的数据。

## 2 XSL

解析XML文档内容，就不得不对XSL（Extensible Stylesheet Language）进行分析处理。这是因为XSL是XML的样式表语言，定义了XML的语法规范。一个XSL样式表集合了一系列设计规则，用于从XML文件中抽取信息，并将其转换成HTML、XML或其他格式的文档。

XSL可分为两大部分：转换XML文本内容和格式化XML文本内容。首先通过XSL转换(过滤和整理)XML数据内容，再利用XSL的格式化显示方法定义数据内容的显示方式(比如字体、大小、颜色等)。

XSL能使Web浏览器直接根据用户的不同需求改变文档的显示法。例如，不需要与服务器进行交互通信，就可以改变数据的显示顺序。通过变换样式表，可以展开或折叠文档。基于XML的网站除了运行速度更快、更易使用外，而且对用户也是透明的。

XSL在网络应用中可分为两种模式，服务器端转换模式和客户端转换模式。前者在XML文件下载到浏览器前先转换成HTML，然后再将HTML文件送往客户端进行浏览。而后者则将XML和XSL文件都传送到客户端，由浏览器进行实时转换。当然，客户端转换模式要求浏览器支持XML+XSL。

## 3 XML解析器

简单的说，一个XML解析器就是一段可以读入XML文档并分析其结构的代码。目前，广泛使用的解析器主要有：IBM公司的XML4J，Microsoft公司的MSXML，Oracle公司的XML Parser for Java和SUN公司的Project X。就针对XML标准的支持程度而言，其中当属SUN公司的解析器表现最为出色。

有多种不同的方法划分解析器种类，例如是否支持完整性检查和处理文档的不同方式。根据对文档的不同处理方式，可分为基于SAX的解析和基于DOM的解析器。前者由事件驱动，通过串行的方式来处理文档，即当它遇到一个开始或者结束标记的时候，它向应用程序发送消息，由应用程序决定如何进行处理。后者则根据文档内容建立一个层次的数据结构，提供用户一个操作文档的接口。

### 3.1 SAX解析器

SAX(Simple API For XML)是一个基于事件的XML文档解析标准。与AWT(Abstract Window Toolkit)中的事件驱动机制相类似，SAX通过事件驱动来识别XML文档的内容，即当它在XML文档中发现特殊符号时，它就会解发相关的事件。由于SAX的这一特性，使应用程序开发人员可以在相应的事件中写入特定的处理代码。

SAX以序列的形式处理文档，不需要在内存中建立

整个文档的树结构，因此与DOM相比，SAX对内存的需求要少得多，可认为其是一个轻量级的接口集合。图1是SAX解析XML文档的流程。


图 1　SAX解析XML文档流程

### 3.2 DOM解析器

DOM(Document Objeçt Model)是W3C发展的浏览XML文档的一种标准API，其不仅提供了对存储在内存中的XML文档的一个完全的表示，也提供了随机访问整个文档的方法。因此，可将DOM看作为一个标准的连接文档和应用程序或脚本语言的结构体系，其提供给用户一个接口以装载、定位、操作和序列化XML文档。如图2所示，CSS(Cascading Style Sheet)是级联样式单，储存并控制元素的显示样式，例如字体，颜色等；Script脚本控制元素如何动作，例如利用条件控制语句对元素进行不同的操作等；而DOM则作为脚本和对象的通信平台，并将结果提交给浏览器。

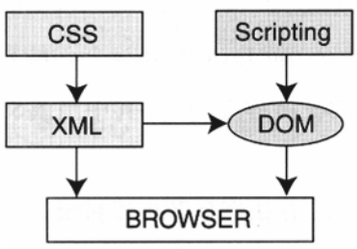


图 2

通过DOM，用户能够把文档看成是一个有结构的信息树，而不仅仅是简单的文本流。这样应用程序或脚本即使不知道XML的语义细节也能够方便的操作该结构。DOM包含两个关键的抽象：一是树状层次，二是表示文档内容和结构的节点集合。树状层次包括了所有这些节点，节点本身也可以包含其他的节点，这就使得开发人员能够通过这个层次结构找到并修改特定节点的信息。DOM把节点看成是一个通常的对象，这样就有可能创建一个脚本来装载一个文档，然后遍历所有的节点，显示感兴趣的节点的信息。

### 3.3 SAX和DOM的比较

DOM操作XML文档时，首先读取该文档，然后将其分割成单个的对象（例如元素、属性、注释等），再在内存中创建一个关于该文档的树结构。DOM解析方式使得开发人员能够反复使用该文档信息，但当文档很大时，所需消耗的内存空间就非常可观。而SAX不会因为XML文档尺寸的增大而增加对内存的需求，因此其对内存的需求比DOM要小得多。然而，正因为SAX解析器没有将整个文档存放到内存中，因此其不能随机的定位到文档的特定部分，也不能实现复杂的搜索。开发人员在处理过程中也必须按顺序处理信息。

SAX允许用户在任何时候终止对XML文档的解析，这使得处理文档的部分信息成为可能——在得到某些特定信息后，就可终止解析。这一特性使得系统资源得到极大的优化。

## 4 MSXML解析器

MicroSoft公司开发的MSXML解析器支持DOM Level 1 1.0标准，其根据XML文档生成一棵DOM树，能够读取XML文档并据其内容创建一个节点的逻辑结构，文档本身被认为是一个包含了所有其他节点的节点。

MSXML解析器读取一个XML文档，然后将其内容解析到一个抽象的节点容器中，这些节点代表了文档的结构和内容。应用程序可读取和操作文档中的信息而无须显式的知道XML的语义。

在一个文档被解析以后，它的节点能够在任何时候被浏览而不需要保持一定的顺序。

下文将介绍MSXML的主要COM接口。

### 4.1 DOMDocument

DOMDocument对象是XML DOM的基础，也是开发人员最重要的编程对象，可利用其公有属性及方法来浏览、查询或修改XML文档的内容和结构，并导航到树的其他对象。以下为该接口的公有属性：

— async

— validateOnparse

— resolveExternals

— perserveWhiteSpace

每个属性均可接受或返回一个布尔值。前三者的缺省值为TRUE，perserveWhiteSpace的值与XML文档的设置有关。

### 4.2 XMLDOMNode

IXMLDOMNode是DOM的基本的对象，不论是元素、属性、注释、过程指令，或是其他的文档组件，都可认为是IXMLDOMNode。

### 4.3 IXMLDOMNodeList

IXMLDOMNodeList是节点对象的集合，对节点的各种操作诸如增、删、改均可在其中立刻反映出来。

### 4.4 IXMLDOMParseError

IXMLDOMParseError接口返回解析过程中出现的信息，包括错误号、行号、字符位置和文本描述。

下面是一个应用MSXML的简单示例：

```
.....
IXMLDOMDocument *pIXMLDOMDocument = NULL;
wstring strFindText (_T("author"));
IXMLDOMNodeList *pIDOMNodeList = NULL;
IXMLDOMNode *pIDOMNode = NULL;
long value;
BSTR bstrItemText;
HRESULT hr;
Try
 {
.....
// 得到一个和标签名称author相关的所有节点的集合
…
hr = pIXMLDOMDocument->getElementsByTagName
(
  (TCHAR*)strFindText.data(), &pIDOMNodeList);
SUCCEEDED(hr) ? 0 : throw hr;
// 得到所包含的节点的个数
hr = pIDOMNodeList->get_length(&value);
if(SUCCEEDED(hr))
{
  pIDOMNodeList->reset();
  for(int ii = 0; ii < value; ii++)
  {
    // 得到具体的一个节点
     pIDOMNodeList->get_item(ii, &pIDOMNode);
     if(pIDOMNode )
     {
     // 得到该节点相关的文本信息
     pIDOMNode->get_text(&bstrItemText);
     ::MessageBox(NULL, bstrItemText,strFindText.data(),
MB_OK);
     pIDOMNode->Release();
     pIDOMNode = NULL;
        }
           }
  }
  pIDOMNodeList->Release();
  pIDOMNodeList = NULL;
    }
    catch(...)
    {
        if(pIDOMNodeList)
          pIDOMNodeList->Release();
        if(pIDOMNode)
          pIDOMNode->Release();
        DisplayErrorToUser();
    }

.....
```

## 5 结束语

DOM解析器和SAX解析器各有优劣，各擅胜场。在需要了解文档结构、或需要操作文档的某些部分（如对某些元素排序），或需要反复使用文档信息的场合，DOM解析器是非常适宜的。而如果只需从XML文档中提取若干元素，或没有太多内存空间，或者只需使用文档中的信息一次，则可使用SAX解析器。


参考文献

1 www.sun.com

2 www.xml.org

3 www.microsoft.com