第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)20—0059—03 文献标识码：A 中图分类号：TP393.02

基于链式结构 XML 文档的生成方法

陈再良，徐德智，陈学工，沈海澜
（中南大学信息科学与工程学院，长沙 410083）

摘 要：提出了一种基于链式结构的 XML 文档生成方法，设计了一个利用 Java 中的 stream tokenizer 类实现 HTML 文档解析的算法，将解析得到的元素内容及文本内容生成的结点插入到相应的位置上，同步生成 DOM 解析树，对 DOM 解析树进行遍历，将遍历得到的信息以二叉链表的形式存储，采用改进的先根遍历算法对该二叉链表遍历，提取相应的信息构建 DTD，完成整个转换生成的过程。
关键词：HTML；XML；DOM；解析

Generation Method of XML Document Based on Chain-link Structure

CHEN Zailiang, XU Dezhi, CHEN Xuegong, SHEN Hailan
(College of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 This paper puts forward the method of XML document based on linked-structure. It uses stream tokenizer to design an algorithm of HTML document parse. The element and text contents are inserted into the correct position to create DOM-parsing-tree as parsing. Lastly. After visiting the tree and storing the information into a binary-linked-list, it uses a modified preorder algorithm visiting the linked-list, then extracts corresponding information to build DTD and finish the whole generation process.
【Key words】 HTML; XML; Document object model(DOM); Parse

当前网络上大多数的网页是由HTML写成的，而它不能表达数据本身的含义，因而不能很好地满足实际应用中的一些需求。XML[1]具有很多优于HTML的技术，XML数据形式的网页，易于进行网页信息的集成、提取、检索、过滤和挖掘分析等。随着XML在网络的广泛应用，以XML代替HTML来进行网络上的信息发布将会成为一种趋势。

在此情况下，不但要继续发展 XML 的相关技术使新的信息以 XML 的方式存储和传播，同时必须考虑到原有数据的保留问题，因此把 HTML 中的信息以 XML 的格式描述出来，具有十分重要的现实意义。

1 常用的转换方法

要完成从 HTML 到 XML 的转换，首先必须考虑 HTML 文档的结构。HTML 文档的内容是有顺序关系的，在转换过程中要求保持这种有序性，同时还要保证转换前后信息的完整性。其次虽然二者语法上都源于 SGML 的思想，但 HTML 不允许用户定义自己的扩展标签，而 XML 可以让用户自行定义标记及属性名，从而结构化地描述信息内容。目前的转化方案主要有 3 种：(1)基于多叉树的 HTML 到 XML 的转换方案；(2)基于栈的 HTML 到 XML 的转换方案；(3)基于内容的 HTML 到 XML 的转换方案。具体地说，前 2 种属于同一类型，即利用 XHTML 为中介，按照 HTML 到 XHTML 再到 XML 的过程来进行转化，而后一种直接实现了从 HTML 向 XML 的转化。

对 HTML 向 XML 转换的研究主要是集中在形式转换的研究上，实现真正意义上的 HTML 向 XML 的转换，所需工作量大且方法非常复杂，而现在最迫切的任务就是要研究合适的转换方法来实现将大量的 HTML 文档转换为 XML 的形式。

2 基于链式结构的转换方法

本文提出了一个基于链式结构由 HTML 向 XML 转换的方法。

(1)利用 Java 语言中提供的 StreamTokenizer 类来进行 HTML 解析器的编写，同时在文档解析的过程中，将解析结果生成一棵 DOM 解析树，即边解析边将解析得到的元素内容及文本内容生成结点插入到树中相应的位置上，在文档解析完毕后，DOM 树的构建也同步完成；

(2)以 DOM 中的 Document 接口为入口，根据 DOM 接口所提供的结点信息来实现对 DOM 树的先根遍历；

(3)根据对 DOM 树中结点的遍历顺序的特点，选用二叉链表形式来存储遍历 DOM 树时所得到的各结点信息，即记录遍历 DOM 树得到的结点序列信息，生成该序列所对应的二叉链表形式；

(4)采用改进的先根遍历方式遍历该二叉链表，构建 DTD，完成整个转换过程。

2.1 解析 HTML 文档为 DOM 解析树

DOM[2]是W3C制定的一个接口规范，主要由 3 个部分组成：核心(Core)，HTML接口和XML接口。核心部分是结构化文档底层对象的集合，这一部分所定义的对象已经完全可以表达出HTML和XML文档中的数据；HTML接口和XML接

基金项目：高等学校优秀青年教师教学科研奖励计划基金资助项目(20025)
作者简介：陈再良(1972－)，男，讲师，主研方向：XML数据库；徐德智，教授、博士；陈学工，副教授，博士；沈海澜，讲师
收稿日期：2005-12-28 **E-mail：**xxxyczl@163.com

口这两部分是专为操作具体的HTML文档和XML文档而提供高级接口，目的是为了更方便地操纵这两类文件。

解析HTML文档时，利用Java提供的StreamTokenizer类[3]，可大大简化解析过程。

(1)Java 的输入流类 StreamTokenizer

StreamTokenize 类的作用是将一个输入流变成令牌流。令牌流中的令牌实体有 3 类：单词(多字符令牌)，单字符令牌和空白。

令牌是 HTML/XML 文档中有完整语义的单元。例如一个元素通常对应的令牌有元素的起始标签、元素的内容、元素的结束标签、“ < ”和“ > ”。而一个元素通常对应 DOM 树上两个结点，起始标签和结束标签对应一个元素结点，元素内容则对应另一个结点，通常是文本结点。由于元素的起始标签中可能有属性信息，因此 token 中也存储有关于元素属性的信息。

(2)文档解析为 DOM 树的过程

解析时，HTML 文档各标签元素的处理情况通过预设一个堆栈来记录；同时，预设一链表，其结点中都存储有一个常见的 HTML 标签及其对应的 ID 值。

整个解析树的构建从根结点开始。首先读入待分析的 HTML 文档，并将输入流转化为令牌流，接着依次读入令牌对文档进行解析。

(3)解析文档为 DOM 树的基本算法

DOM 解析树生成算法如下：

```
void parserHTMLtoDOM( )
{int token;
 switch(token=nextToken( )) //读入文档
 {case 'TT_EOF': //读到流的尽头
          return TT_EOF;
          break;
   case '<':
             if(nextToken( )= ='/')
{nextToken( );
getTagID( );     //获取对应标签 ID 名
OutStack(ID);
getCurrNode( );//树的当前活动结点
       return parserHTMLtoDOM( ); }
          else
//对应起始标签生成元素结点插到 DOM 中
     {nextToken( );
                getTagID( );
                InStack(ID);
              getElement(currToken);
              insertNode( );
              getCurrNode( ); }
           break;
 case 'text':                //读入文本部分
 getElement(currToken);
 insertNode( );
          getCurrNode( );
          break;
 case '>':
          return parserHTMLtoDOM( );
          break;
 default:
          alert("Unkown token");    }
}
```

```
void getCurrNode( )
{currStackTop=getStackTop( );
if(currNode.nodeValue( )=getTagID(root))
   return(currNode);
else{
//从该活动结点开始回溯查找新的活动结点
for(currNode;(currNode.nodeValue(   )!=getTagID(root))&&(currNode.nodeValue( )!=currStackTop); currNode=currNode.parent( ))
{   }
return(currNode); } }
```

2.2 DOM 解析树的遍历

DOM 只是一棵普通的树，先根遍历、后根遍历都可以，但采用先根遍历的方式所得到的数据序列与原文档中数据的组合方式更为贴切，后根遍历所得到的数据序列在一定程序上破坏了原文档数据的顺序性。实现对 DOM 树的遍历时先获得其入口结点 Document，然后从此结点开始，用递归的方法对 DOM 树进行先根遍历。

2.3 二叉链表结构的 Java 实现

采用 Java 中的“ 引用 ”功能来实现链表的操作。引用是用来“ 引用 ”某一个数据对象的，它除了表示某块内存地址外，还能表示其他信息，如该数据对象的类型等。

树的二叉链表表示法，即以二叉链表作为树的存储结构[4]。二叉链表结点的 2 个链域分别为指向该结点的第 1 个孩子结点和下一个兄弟结点，命名为firstchild和nextsibling，链表结点如图 1 所示。

图 1 二叉链表结点

依据对 DOM 树遍历过程中访问各结点的顺序特点，可以很容易地抽取树中的信息并以一个二叉链表的形式进行存储。

二叉链表结点的 Java 实现程序如下：

```
class CSNode
{ public String data;
public CSNode firstchild，nextsibling;
public CSNode(String value)
{data = value;
firstchild = null;
nextsibling = null; }
}
```

二叉链表的实现算法如下：

```
public static void createCSList(CSNode cs)
{DataInputStream dis = new DataInputStream(System.in);
String listStr;
listStr = dis.readLine( ).trim( );
 //将序列以字符串形式存储
createListNode(cs，listStr，1); }
public static void createListNode(CSNode cs，String listStr，int index)
{CSNode root = cs;
if (listStr.charAt(index-1)==' ')
root = null;
else //生成根结点
root.data =String.valueOf(listStr.charAt(index-1));
if ((2*index<listStr.length( ))&&(listStr.charAt(2*index-1)!=' '))//寻找左子树的根
{root.firstchild = new CSNode(' ');
```

```
createListNode(root.firstchild，listStr，2*index);}
else
root.firstchild = null;
if ((2*index+1<listStr.length( ))&&(listStr.charAt(2*index)!=' '))
//寻找右子树的根
{root.nextsibling = new CSNode("");
createListNode(root.nextsibling，listStr，2*index+1); }
else
root.nextsibling = null;
return; }
```

2.4 二叉链表的遍历算法的改进及 DTD 的构建

(1)二叉链表的遍历

若把二叉链表在结构形式上看作一个二叉树的话，对二叉链表的遍历有先根遍历、中根遍历、后根遍历等方式。根据 DOM 解析树映射到二叉链表的过程，对于一个结点，它的 firstchild 域指向其下一级，而 nextsibling 域则指向其同一级。遍历时采用改进的先根遍历算法，在访问一个结点后，接下来就访问它的兄弟结点，然后再去访问它的子结点。改进的遍历算法复杂度没有发生变化，这有利于 XML 文档的生成。

(2)DTD 的构建

DTD(Document Type Definitions)描述了一个置标语言的语法和词汇表，也就是定义了文档的整体结构以及文档的语法。DTD 规定了一个语法分析器，解释有效的 XML 文档所需规则的细节。XML 的精髓是允许文档的编写者制定基于信息描述、体现数据之间逻辑关系的自定义标记，确保文档具有较强的易读性、清晰的语义和易检索性。一个完全意义上的 XML 文档不仅是格式良好，而且还应该使用了一些自定义标记的有效的 XML 文档，也就是说，它必须遵守 DTD 中已声明的规定。一个 DTD 可以是内部的，包含在 XML 文档的前导说明部分；也可以是外部的，作为一个外部文档被引用。二叉链表在遍历的同时，将遍历得到的信息生成相应的 DTD。

3 结束语

目前万维网存在着明显的不足，如计算机不理解网页内容的语义、网上有用信息难找，即使借助搜索引擎，查准率也较低。Berners T L 提出了下一代万维网——“语义网”(Semantic Web)的理念。目前大多数网页还是采用 HTML，利用二叉链表能方便、准确地将 HTML 转换为 XML，有利于网络上相关信息的查找、提取。

参考文献

1 T Bray, Paoli J, Sperberg C M. Extensible Markup Language (XML)1.0[EB/OL]. http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

2 Le Hégaret P, Whitmer R, Wood L, et al. Document Object Model[EB/OL]. http://www.w3.org/DOM-2005/07/07, 2005.

3 Liu L, Calton P. Han W. XWRAP: A XML-enabled Wapper Construction System for Web Information Sources[C]. Proc. of International Conference on Data Engineering, San Diego, USA, 2000-02: 611-621.

4 严蔚敏, 吴伟民. 数据结构(C 语言版)[M]. 北京: 清华大学出版社, 2002-02.

(上接第 3 页)

相邻像素间的差别通常在一定范围之内。因此，图像在小波域中的系数幅度衰减规律就较强，这样，用 VDEZW 方法编码图像时的率失真性能和 EZW 算法十分接近或相同。

4 结论

嵌入零树小波压缩算法具有很高的压缩比，而且具有优良的压缩性能。为了进一步提高编解码速度，由图像小波变换能量聚集和子带间的幅度衰减性等特性，在小波分层嵌入编码过程中，对不同的重要图，部分小波子带的系数幅度是不会超过当前的阈值，从而可以忽略不记，即小波树的深度随着重要图的不同而改变，也就是变深度嵌入编码。实验结果显示，对于自然图像来说，不需要考虑的层中的小波系数幅值几乎全部小于当前的阈值，从而和 EZW 算法相比，编码不会影响压缩率（或图像质量），而编解码过程中的运算量却明显下降。同时，该算法思想也可以应用于其他具有嵌入特性的小波压缩算法中，以提高编解码速度。

参考文献

1 Shapiro J M. Embedded Image Coding Using Zerotree of Wavelet Coefficient[J]. IEEE Trans. on Signal Processing, 1993, 41(12): 3445-3462.

2 Said A, Pearlman W A. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees[J]. IEEE Trans. on Circuits Syst. Video Technol, 1996, 6(3): 243-250.

3 Zandi A, Allen J D, Schwartz E L, et al. CREW: Compression with Reversible Embedded Wavelets[C]. Proc. of Data Compression Conf., 1995, 1: 212-221.

4 Gormish M J, Schwartz E L, Keith A, et al. Lossless and Nearly Lossless Compression for High Quality Images[C]. Proc. of SPIE, 1997-02.

5 Creusere C D. A New Method of Robust Image Compression Based on the Embedded Zerotree Wavelet Algorithm[J]. IEEE Trans. on Image Processing, 1997, 6 (10): 1436-1442.

6 Creusere C D. Fast Embedded Compression for Video[J]. IEEE Trans. Image Processing, 1997, 6 (12): 1811-1816.

7 张孝杰, 张专成, 李　研. 一种改进的嵌入式零树小波图像编码算法[J]. 计算机应用, 2004, 24 (7): 52-53.

8 杨　旭. 基于小波等级树的分组集合划分图像编码[J]. 信号处理, 2005, 21(1): 70-73.

9 张德丰. 基于嵌入式零树小波编码算法的研究[J]. 模糊系统与数学, 2004, 18(2): 121-125.

10 王祥林, 吴国威, 林行刚. 一种基于零树的多码率小波图像编码方法[J]. 电子学报, 1997, 25(4): 48-51.

11 黄佳庆, 翁默颖. 一种基于子波变换的改进型零树量化编码算法[J]. 华东师范大学学报(自然科学版), 1999, (3): 47-51.

12 王向阳, 杨红颖. 基于多阈值与嵌入零树小波的图像压缩算法[J]. 通信学报, 2001, 22(12): 88-93.

13 付文秀, 郗来浩, 窦　攀等. EZW 图像编码算法的改进及仿真实现 [J]. 吉林大学学报(信息科学版), 2004, 22 (2): 93-97.

14 Carey W K, Pischke L A V, Hemami S S. Rate-distortion Based Scalable Progressive Image Coding[C]. SPIE Conference on Mathematics of Data / Image Coding, Compression, and Encryption, San Diego, California, 1998-07: 197-209.

15 Wrtten I H, Neal R M, Cleary J G. Arithmetic Coding for Data Compression[J]. Commun. of ACM, 1987, 30(6): 520-540.