

XML prin SAX

Sabin-Corneliu Buraga

Articol aparut in PC Report vol.9, 01 (89), ianuarie 2000

Am prezentat in precedentul articol un model obiectual destinat procesarii documentelor XML, model denumit **DOM (Document Object Model)**. In continuare vom ilustra concret cum, prin intermediul DOM, putem interveni in structura unui document XML, modificandu-l in mod dinamic. Analiza documentelor XML se poate realiza independent de navigator sau chiar in cadrul navigatorului Web.

Interfata SAX

Pentru implementarea DOM se pot utiliza mai multe metode, una dintre cele mai facile fiind **SAX (Simple API for XML)**, o interfata simpla de programare destinata manipularii documentelor XML, inasa nu atat de completa precum DOM. Majoritatea celor care se ocupa de XML vad structura unui document in forma arborescenta, iar modelul DOM ofera din plin posibilitatea de a manipula informatiile in aceasta maniera. Din punctul de vedere al implementatorilor aceasta abordare are numeroase deficiente (maniera de stocare interna a arborelui de obiecte, parcurgerea lui etc.). Unul dintre beneficiile utilizarii interfetei SAX este ca arborele nu mai trebuie construit, dindu-i-se programatorului o alta cale de manipulare a documentului XML. In plus, SAX poate ajuta la convertirea datelor din formatul arborescent DOM in alt format mai comod, iar pentru parsare nu este necesar a se memora intreaga informatie XML ci numai partile dorite.

In mod uzual, SAX are implementari in C sau Java.

SAX in Java

Implementarile Java pentru SAX sint incapsulate in pachetul de clase `org.xml.sax` care poate fi obtinut gratuit prin Internet. Ca drivere de parsare SAX se pot mentiona **IBM XML for Java** si **XP** (James Clark). Alternativele sint **SAXDOM** sau **MSXML** (Microsoft), iar daca se doreste o procesare mai elaborata a unui document XML se poate utiliza un servlet numit **DOMit**, dezvoltat de IBM. Drept aplicatii bazate pe SAX putem enumera: **JUMBO** (vizualizator de documente XML), **FREE-DOM** (implementare libera a unor biblioteci Java de analiza XML), **SAXON** (interfata de nivel ridicat bazata pe SAX), **Alfred XML Parser** si altele. Desigur, exista parsere XML si pentru Perl, Python, Delphi etc. disponibile pentru diverse platforme.

Vom ilustra succint pachetul de distributie SAX `org.xml.sax` care se compune din 11 clase si interfete de baza, la care se alatura 3 clase suplimentare si 4 clase demonstrative, totul divizindu-se in 5 grupuri:

- a. *interfete implementate de analizorul (parserul) documentelor XML: `Parser` si `AttributeList` (obligatorii), plus `Locator` (optionala); analizorul XML se mai denumeste si *SAX Driver*.*
- b. *interfete implementate de aplicatia care doreste sa manipuleze documentele*

XML via driverul SAX: `DocumentHandler`, `ErrorHandler`, `DTDHandler`, `EntityResolver` pentru procesarea documentului, raportarea erorilor, analiza definitiei tipului de document (Document Type Definition) si rezolvarea entitatilor XML, respectiv. Toate aceste clase pot fi utilizate optional.

- c. *clase SAX standard* (care pot fi folosite atat de parsere cit si de aplicatii): `InputSource`, `SAXException`, `SAXParseException` si `HandlerBase`, toate acestea fiind implementate in intregime de SAX.
- d. *clase aditionale* specifice Java, complet implementate: `ParserFactory`, `AttributeListImpl` si `LocatorImpl`.
- e. *clase demonstrative* a capabilitatilor SAX, in fapt aplicatii Java: `SystemIdDemo`, `ByteStreamDemo`, `CharacterStreamDemo` si `DemoHandler`. Aceste clase nu fac parte din specificatiile de baza ale SAX si pot sa nu apara in implementari SAX in alte limbaje.

Interfata de baza a interfetei SAX este `org.xml.sax.Parser` care trebuie implementata de toate analizoarele XML, permitind aplicatiilor sa inregistreze metode de raspuns la diverse tipuri de evenimente si sa initializeze procesul de analiza a documentelor XML incarcate via URI sau printr-un flux de caractere.

Interfata `org.xml.sax.DocumentHandler` reprezinta interfata principala pe care trebuie s-o implementeze o aplicatie utilizind SAX. Daca aplicatia doreste sa fie informata asupra evenimentelor ce pot interveni in procesul de analiza, va implementa aceasta interfata si va folosi o instanta a ei prin intermediul metodei `setDocumentHandler` in cadrul parserului SAX. Parserul va utiliza instanta pentru a raporta diverse evenimente precum inceputul/sfirsitul unui element XML (`startElement`, `endElement`), a unei instructiuni de procesare etc. Ordinea evenimentelor va fi in fapt data de ordinea de aparitie fizica a diverselor elemente ale documentului XML care se doreste a fi analizat.

Pentru procesarea atributelor asociate elementelor XML se va folosi interfata `org.xml.sax.AttributeList`. Iata un exemplu de iterare a intregii liste de attribute asociate elementelor unui document XML:

```
public void startElement (String ename, AttributeList atts) {
    for (int i = 0; i < atts.getLength(); i++) {
        String name = atts.getName(i);
        String type = atts.getType(i);
        String value = atts.getValue(i);
        ...
    }
}
```

Daca elementul nu va avea attribute, atunci `getLength()` va returna 0. Functiile `getName()`, `getType()` si `getValue()` furnizeaza numele, tipul (CDATA, ID, IDREF, NMTOKEN, ENTITY etc.) si valoarea atributului, respectiv.

Clasa `org.xml.sax.Locator` este folosita pentru localizarea documentelor XML prin intermediul unui URI (Uniform Resource Identifier). Daca o aplicatie are nevoie de redirectarea URI-urilor, atunci poate implementa `EntityResolver`.

Ca *middleware* intre un driver (parser) SAX si un client (aplicatie) SAX poate fi folosit procesorul arhitectural **XAF** conceput de Megginson.

SAX in C

O implementare gratuita in limbajul C pentru SAX este cea reprezentata de **libxml**

(denumita si **gnome-xml**), o biblioteca de procesare a documentelor XML scrisa de Daniel Veillard de la Consorțiul Web, fiind inclusa si in mediul de dezvoltare GNOME (GNU Network Object Model Environment) din distributia de Linux RedHat 6. In fapt, **libxml** implementeaza intreg modelul DOM pentru procesarea documentelor XML.

Interfata cu programatorul se concretizeaza in existenta a doua functii C utilizate la analiza unui document XML (aflat in memorie sau incarcat de pe disc):

```
#include <parser.h>

xmlDocPtr xmlParseMemory(char *buffer, int size);
xmlDocPtr xmlParseFile(const char *filename);
```

Pentru crearea arborelui de obiecte-nod DOM, vom utiliza functii de manipulare arborescente al caror prototip se afla in **tree.h**. Un simplu exemplu, in care se adauga noduri-obiect prin **xmlNewChild()**, iar elementelor avind atribute li se asociaza acest valorile atributelor cu **xmlSetProp()**:

```
xmlDocPtr doc; /* documentul XML */
xmlNodePtr tree, /* arborele asociat */
          subtree; /* subarbore */

doc = xmlNewDoc("1.0"); /* creeaza o instanta de document */
doc->root = xmlNewDocNode(doc, NULL, "articol", NULL);
xmlSetProp(doc->root, "vers", "draft");
xmlSetProp(doc->root, "author", "Sabin-Corneliu Buraga");
tree = xmlNewChild(doc->root, NULL, "head", NULL);
subtree = xmlNewChild(tree, NULL, "title", "XML prin xml-lib");
tree = xmlNewChild(doc->root, NULL, "sect", NULL);
subtree = xmlNewChild(tree, NULL, "title", "Prezentare XML");
subtree = xmlNewChild(tree, NULL, "p", "Un exemplu");
```

Se va crea astfel arborele asociat documentului XML urmator:

```
<?xml version="1.0"?>
<articol vers="draft" author="Sabin-Corneliu Buraga">
  <head
    <title>XML prin xml-lib</title>
  </head>
  <sect>
    <title>Prezentare XML</title>
    <p>Un exemplu</p>
  </sect>
</articol>
```

Arborele creat mai sus poate fi salvat pe disc (o functie de salvare fiind **void xmlDocDump(FILE *f, xmlDocPtr doc)**), eventual compresat in format **gzip**. Aceasta maniera este folosita de aplicatia **gnumeric** care poate salva/incarca fisierele in format XML compresat cu GNUzip.

Un exemplu complet

Fie un document XML utilizat pentru stocarea datelor despre aplicatiile de laborator si despre studenti la disciplina "Programare Web":

```
<?xml version="1.0"?>
<projects year="1999/2000">
  <application name="WebTools in XML" id="2">
    <category>software</category>
    <status>
      <update>Mon, 25 Oct 1999 12:27:45</update>
      <version>0.23</version>
    </status>
    <authors>
      <person>
```

```
<name>Ionut Gavrilă</name>
<email>giangu@infoiasi.ro</email>
<wpage></wpage>
<address></address>
<year>master</year>
</person>
<person>
    ...
</person>
</authors>
</application>
    ...
</projects>
```

Funcția C care analizează informațiile despre persoanele implicate în realizarea unor proiecte este următoarea:

```
/* structura cu datele despre o persoana */
typedef struct person {
    char *name;
    char *email;
    char *year;
    char *address;
    char *wpage;
} person, *person_ptr;

/* functia care analizeaza documentul si returneaza informatii
despre un student */

person_ptr parse_person(xmlDocPtr doc, xmlNodePtr cnode) {

person_ptr ret = NULL;

/* aloca structura, daca se poate */
ret = (person_ptr) malloc(sizeof(person));
if (ret == NULL) {
    fprintf(stderr, "out of memory\n");
    return(NULL);
}
memset(ret, 0, sizeof(person));

/* Nu ne intereseaza care e nodul radacina al documentului
si vom viza doar elementele <name> si <email> */
cnode = cnode->childs;
while (cnode != NULL) { /* mai exista un nod? */
    if (!strcmp(cnode->name, "name")) /* am gasit o persoana */
        ret->name = xmlNodeListGetString(doc, cnode->childs, 1);
    if (!strcmp(cnode->name, "email")) /* memoram adresa e-mail */
        ret->email = xmlNodeListGetString(doc, cnode->childs, 1);
    cnode = cnode->next; /* trece la urmatorul element... */
}

return (ret);
}
```

Desigur, parcurgerea documentului se putea realiza, mai elegant, în mod recursiv, exploatând informațiile din arborele DOM asociat. Biblioteca `libxml` permite și asocierea și procesarea spațiilor de nume XML, diverse reguli de validare a documentelor, oferind suport pentru RDF, XPath și XPointer și altele.

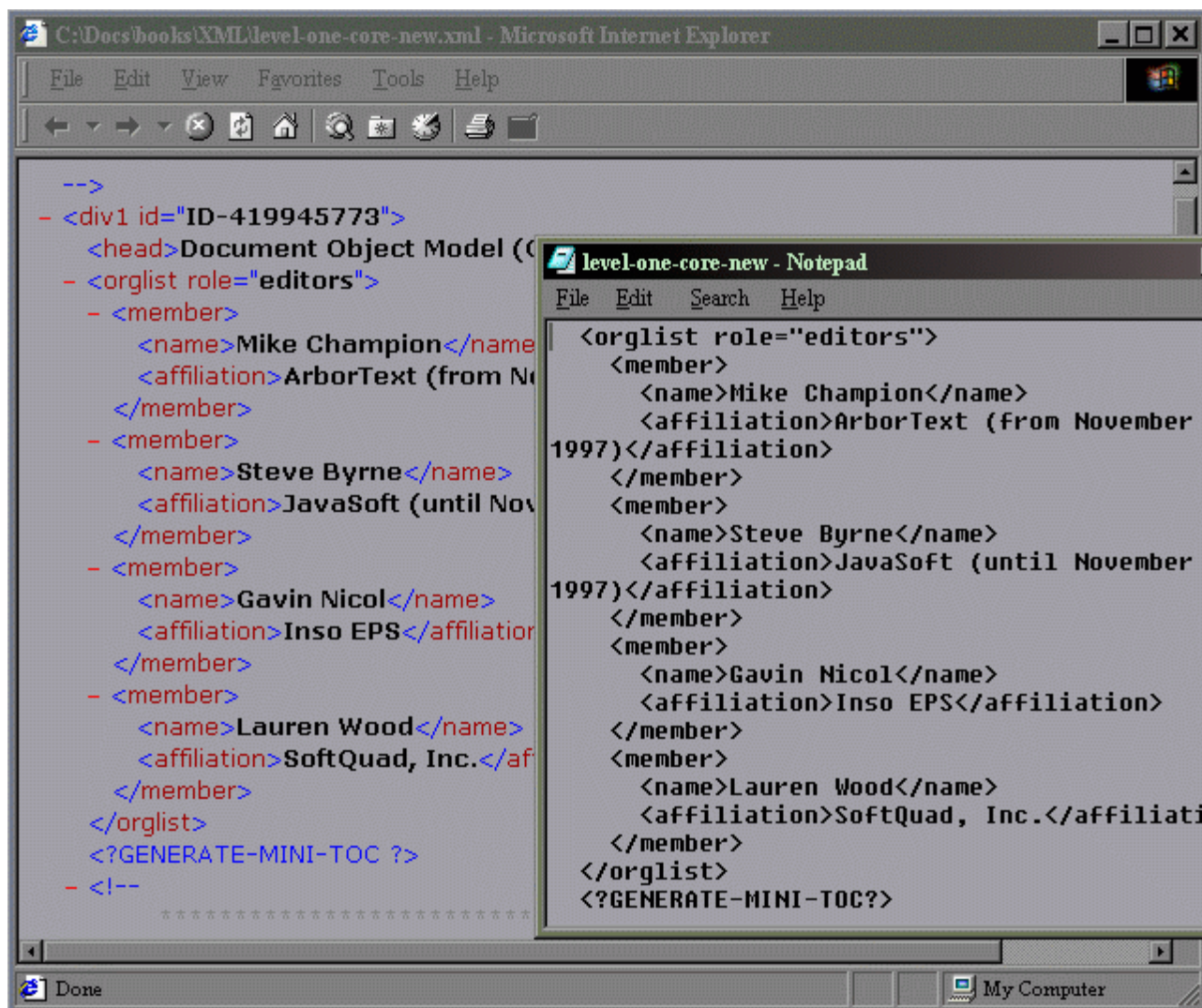
Manipularea documentelor XML prin navigatorul Web

Am văzut până acum cum putem procesa documentele XML, via interfețelor Java sau C, independent de Web, putând concepe aplicații de sine-stătătoare. Mai atractiv ar fi probabil un mijloc dinamic de modificare și de manipulare a unui document XML în cadrul navigatorului Web, prin limbajul JavaScript. Odată cu apariția versiunii 5 a browserului Internet Explorer acest lucru se

poate realiza lejer. Se asteapta si replica de la Netscape, o serie din versiunile de test de la mozilla.org fiind deja disponibile. Netscape Gecko reprezinta noua generatie de navigatoare incluzind suport pentru XML si DOM.

Desigur, implementarea DOM din IE 5 este departe de a fi perfecta si nici macar nu se conformeaza 100% recomandarilor comitetului DOM din cadrul Consorțiului Web, replica Microsoft la aceasta rezumindu-se la urmatorul proverb: *"O camila este un cal proiectat de un comitet"* (Anonim).

De unde putem sti ca IE 5 intelege documente XML? Pur si simplu putem incerca sa incarcam un document XML si sa vedem daca il putem vizualiza. Iata cum arata un fisier XML in IE 5 (observati culorile pentru elementele XML in comparatie cu vizualizarea aceluiasi document in Notepad):



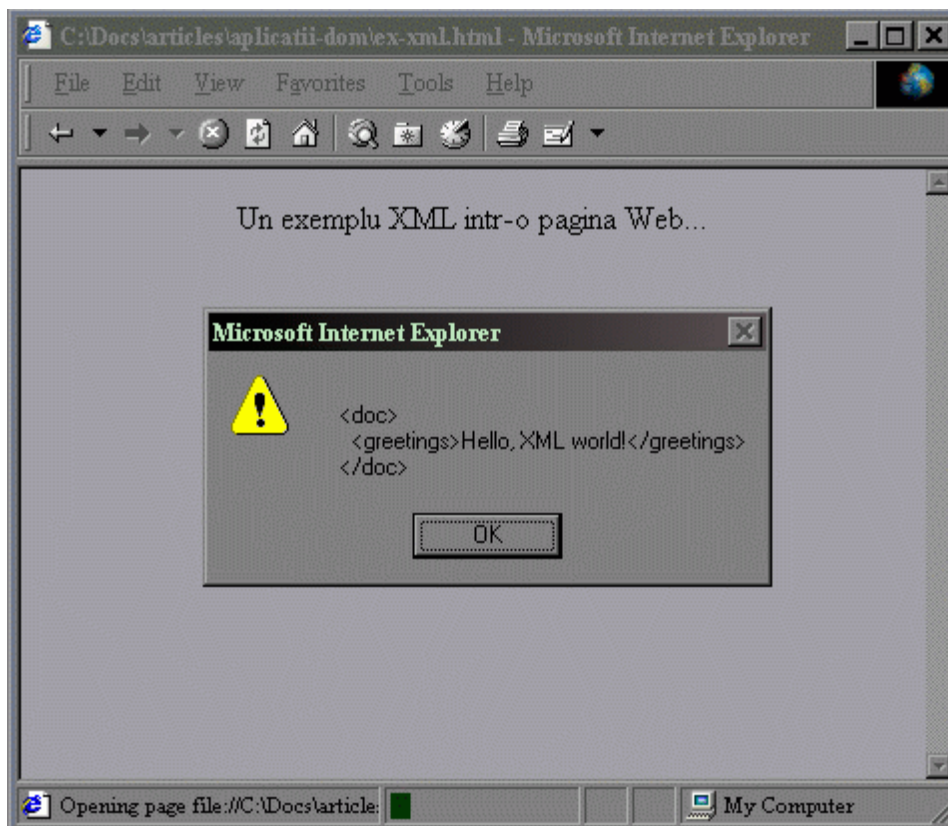
Putem incapsula informatiile XML in cadrul unei pagini HTML astfel:

```
<html>
<body>
<p align="center">Un exemplu XML intr-o pagina Web...</p>
<xml id="fragxml">
  <doc>
    <greetings>Hello, XML world!</greetings>
  </doc>
</xml>

<script language="JavaScript">
```

```
alert (fragxml.innerHTML);  
</script>  
  
</body>  
</html>
```

Marcatorul `<xml>` este un element HTML valid conform specificatiilor Consortiului Web. Identificatorul `innerHTML` reprezinta un fragment de document definit in varianta DOM pentru HTML. Pe ecran va apare:



Folosind metodele definite de interfetele DOM putem concepe un exemplu mai "complex". Dorim sa sortam o lista de carti, in functie de autorii lor. Aceasta lista o vom considera inclusa intr-un tabel HTML (putind la fel de bine fiind stocata si pe disc, intr-un document XML extern). Codul JavaScript inclus in pagina Web si folosit pentru ordonare este urmatorul:

```
function insertionSort(t, iRowStart, iRowEnd, fReverse)  
{  
    var iRowInsertRow, iRowWalkRow;  
    for (iRowInsert = iRowStart + 1; iRowInsert <= iRowEnd; iRowInsert++)  
    {  
        textRowInsert = t.children[iRowInsert].innerText;  
        for (iRowWalk = iRowStart; iRowWalk <= iRowInsert; iRowWalk++)  
        {  
            textRowCurrent = t.children[iRowWalk].innerText;  
            if (((!fReverse && textRowInsert <= textRowCurrent)  
                || (fReverse && textRowInsert >= textRowCurrent))  
                && (iRowInsert != iRowWalk))  
            {  
                eRowInsert = t.children[iRowInsert];  
                eRowWalk = t.children[iRowWalk];  
                t.insertBefore(eRowInsert, eRowWalk);  
                iRowWalk = iRowInsert; // gata!  
            }  
        }  
    }  
}
```

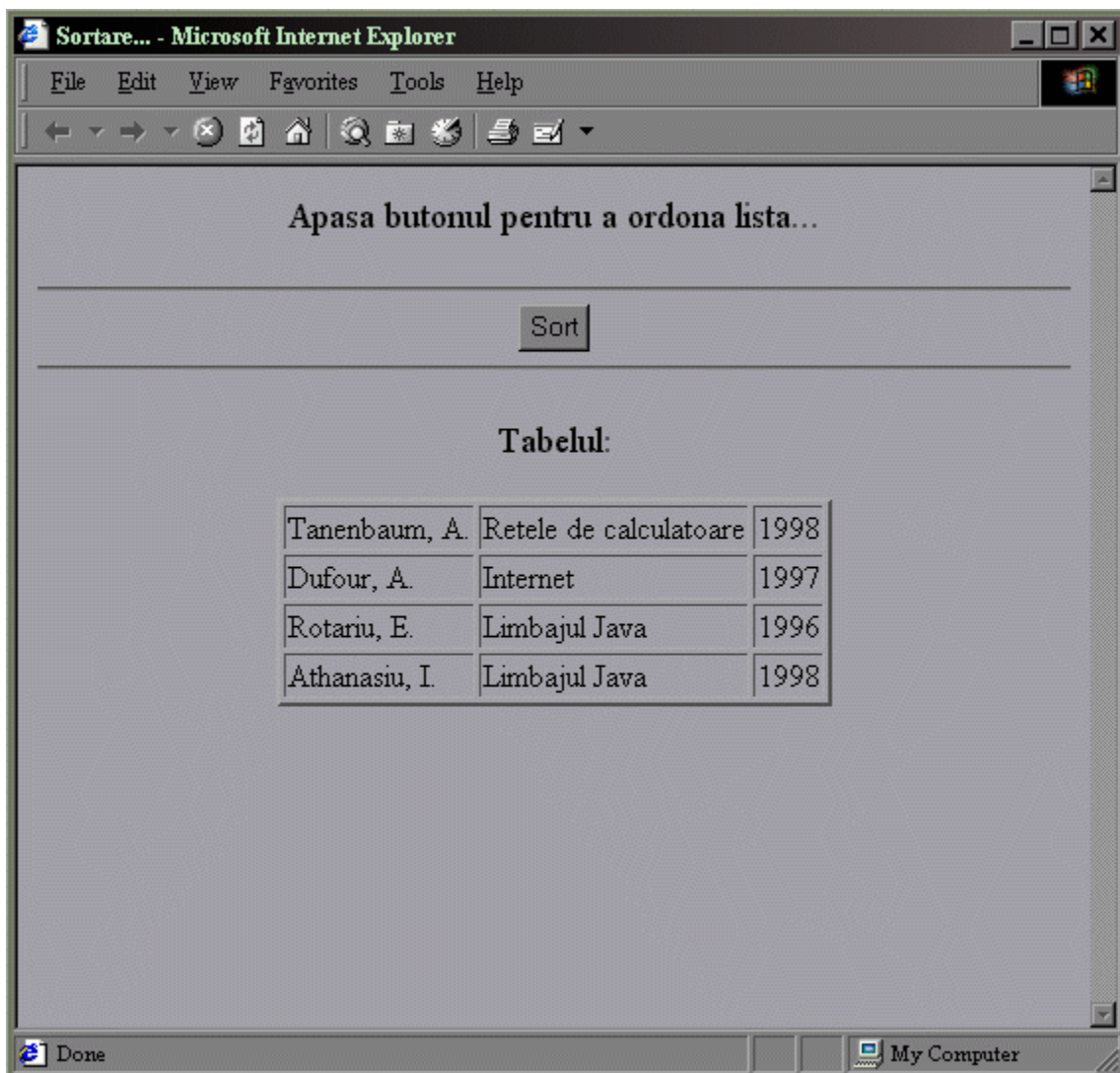
Dupa cum se observa modificarile de pozitie ale elementelor se realizeaza la nivelul arborelui de noduri-obiect asociate elementelor. Apelarea functiei `insertionSort` o vom realiza prin:

```
<input type="button" value="Sort"  
  onclick="insertionSort (TBooks.children[0], 0,  
    TBooks.rows.length - 1, false)">
```

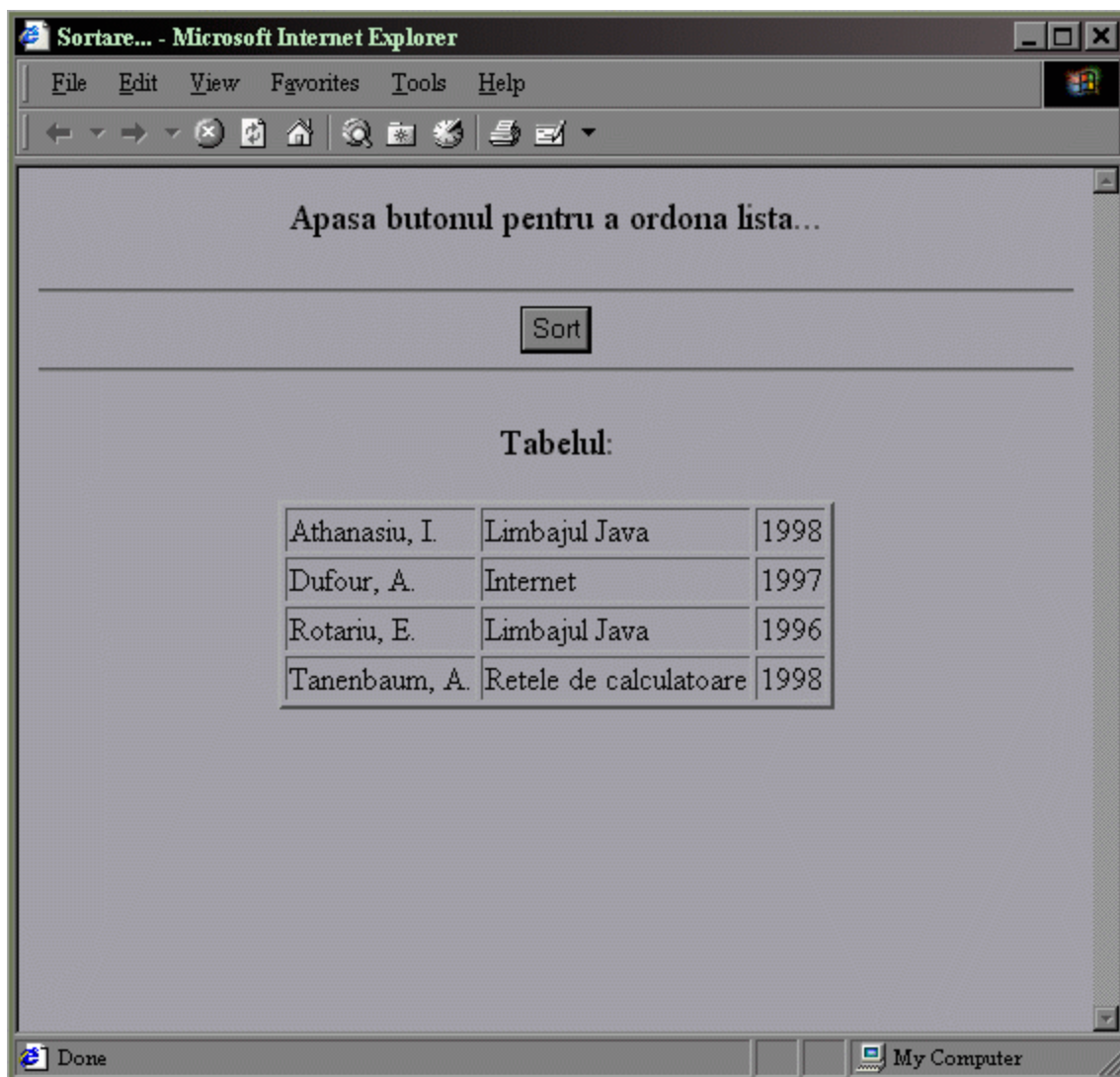
`Tbooks` este identificatorul tabelului cu lista cartilor. Acesta ar putea fi de exemplu:

```
<table id="TBooks" border="2">  
  <tr><td>Tanenbaum, A.</td><td>Rețele de calculatoare</td><td>1998</td></tr>  
  <tr><td>Dufour, A.</td><td>Internet</td><td>1997</td></tr>  
  <tr><td>Rotariu, E.</td><td>Limbajul Java</td><td>1996</td></tr>  
  <tr><td>AthanasIU, I.</td><td>Limbajul Java</td><td>1998</td></tr>  
</table>
```

Inainte de sortare, pagina Web arata astfel:



Dupa ordonarea numerelor de autori vom observa cele de mai jos. Ultima coloana a unui rind de tabel n-a fost sortata, ci doar primele doua. Lasam cititorului sarcina de a vedea de ce.



Folosind metodele definite de interfetele DOM se pot imagina diverse aplicatii de manipulare a documentelor XML si HTML in cadrul paginilor WWW, in maniera dinamica, on-line. Astfel, se poate concepe un joc de tip puzzle, modificarea pieselor realizindu-se prin schimbarea ordinii nodurilor in arborele DOM asociat documentului.

Concluzii

Dupa cum am vazut mai sus, modelul DOM si interfata mai simpla SAX se pot utiliza atit in maniera independenta de Web, cit si in cadrul navigatoarelor WWW actuale, pentru manipularea facila si comoda a documentelor XML.

Resurse

1. Dan Connolly et al. - "The Evolution of Web Documents: The Ascent of XML", World Wide Web Journal Special Issue on XML, Volume 2, Number 4, 1997
2. James Clark: <http://www.jclark.com/>
3. JavaSoft: <http://java.sun.com/>
4. MSDOM: <http://www.microsoft.com/XML/>
5. Rebecca Norlander - "Internet Explorer 5: All Power to the Document Object Model", Site Builder Magazine, March 1999
6. IBM Alphaworks: <http://www.alphaworks.ibm.com/tech/>
7. James Henstridge - "Using the SAX Interface of LibXML", XMLDev, Oct.1999
8. "Client-Side JavaScript 1.3 Guide", Netscape Communications, May 1999

9. *SAXDOM*: <http://www.docuverse.com/personal/saxdom.html>
10. *XML.com*: <http://www.xml.com/>
11. *SAX*: <http://www.megginson.com/SAX/>
12. *Mozilla.org*: <http://www.mozilla.org/>
13. *Robin Cover* - "*The SGML/XML homepage*" (Oct.1999):
<http://www.oasis-open.org/cover/xml.html>
14. *S.Koch* - "*Voodoo's Introduction to JavaScript*", Mannheim University, 1997
15. *libxml*: <ftp://ftp.gnome.org/pub/GNOME/sources/libxml/>
16. *Raph Levien* - "*DOMination*": <http://www.levien.com/gnome/domination.html>
17. *A DOM-Based Sliding Puzzle*: <http://www.webreference.com/js/column45/>
18. *XAF*: <http://www.megginson.com/Software/XAF/>

Sabin-Corneliu Buraga este doctorand in Computer Science la Universitatea "A.I.Cuza" din Iasi; poate fi contactat la adresa: busaco@infoiasi.ro.