



# Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroii

În cadrul acestui articol vom prezenta un set de funcții oferite de limbajul PHP pentru prelucrarea fișierelor XML și vom exemplifica acest set de funcții.

## PHP și XML

Limbajul *PHP* oferă un set de funcții și clase care sunt utile în cazul în care se dorește prelucrarea sau accesarea fișierelor *XML*. Setul de clase este realizat în conformitate cu specificațiile *DOM Level 1 (Document Model Object – Model obiectual de document)* care se referă atât la fișierele de tipul *XML*, cât și la fișierele de tipul *HTML*.

După cum probabil știți, un fișier *XML* are o structură similară unui fișier *HTML* și se deosebește de acesta prin faptul că secțiunile pot avea orice nume și orice attribute și fiecare secțiune are un identificator de sfârșit al secțiunii.

Un fișier *XML*, ca de altfel și un fișier *HTML*, pot fi privite ca o structură arborescentă de date.

Elementele următoare caracterizează această structură arborescentă:

- fiecare secțiune, mai puțin secțiunea rădăcină, are un părinte;
- fiecare secțiune, mai puțin secțiunea rădăcină, are unul sau mai mulți frați (un frate al unei secțiuni este o secțiune care are același părinte cu aceasta);
- fiecare secțiune poate avea 0 sau mai mulți fii;
- fiecare secțiune are un tip și fiecare tip de secțiune are diferite attribute;
- structura unui fișier *XML* poate fi controlată cu ajutorul specificațiilor *DTD (Document Type Definition – Definiția tipului documentului)*; aceste definiții se pot afla în fișiere externe sau chiar în cadrul fișierului *XML*.

În principiu, un document realizat în conformitate cu specificațiile *DOM* are două elemente componente, și anume restricțiile pe care trebuie să le respecte structura arborescentă și această structură. În acest moment se poate construi un nod care în cadrul specificațiilor *DOM* are tipul *DomDocument* și care să fie rădăcina întregii structuri arborescente descrisă de acest document. Acest nod are cel mult doi fii și anume un fiu este constituit de secțiunea prin-

cipală descrisă în document, iar cel de-al doilea fiu este constituit din restricțiile pe care trebuie să le respecte primul fiu. Cei doi fii ai nodului rădăcină sunt de tipul *DomElement* și *DomDocumentType*.

În cadrul specificațiilor *DOM*, orice nod (secțiune) al structurii arborescente este derivat din tipul *DomNode*, deci și nodul de tipul *DomDocument* este un tip derivat din *DomNode*.

În cadrul specificațiilor *DOM* există mai multe tipuri de noduri și funcționalitatea fiecărui tip de nod este dată de o clasă în cadrul limbajului *PHP*.

În continuare vom prezenta clasele *DomNode* și *DomDocument*.

## Clasa DomNode

Această clasă încapsulează proprietățile comune ale oricărui nod din structura arborescentă și, în concluzie, stă la baza tuturor tipurilor de noduri pe care arborele documentului le poate conține.

În continuare vom prezenta metodele și proprietățile acestei clase și menționăm că nici una dintre metode nu poate fi apelată static.

### Metoda DomNode->node\_name()

Această proprietate nu are parametri și returnează un șir de caractere care reprezintă numele unui nod din arbore. Există tipuri de noduri al căror nume nu este utilizat și a cărui modificare nu este luată în considerare sau este generată o eroare. De exemplu, pentru un nod de tipul *DomDocument*, numele este întotdeauna "#document".

### Metoda DomNode->node\_type()

Această metodă nu are parametri și returnează un număr întreg care reprezintă tipul unui nod din arbore. Pentru fiecare tip de nod este definită câte o constantă.

În tabelul 1 pot fi observate numele constantelor oferite de limbajul *PHP* pentru diferitele tipuri de noduri care apar în cadrul specificațiilor *DOM*.

### Metoda *DOMNode->node\_value()*

Această metodă returnează un șir de caractere care reprezintă valoarea (informațiile) asociată nodului curent. Există tipuri de noduri care nu au valoare.

### Metoda *DOMNode->add\_namespace()*

Această metodă setează pentru nodul curent domeniul de utilizare (*namespace*) și primește doi parametri de tipul șir de caractere. Primul parametru reprezintă identificatorul resursei constituite de acest nod (*URI - Uniform Resource Identifier*), iar cel de-al doilea parametru reprezintă prefixul spațiului de utilizare.

### Metoda *DOMNode->set\_namespace()*

Această metodă modifică domeniul de utilizare a nodului curent și primește doi parametri de tipul șir de caractere. Parametrii au aceeași semnificație ca la metoda anterioară, iar cel de-al doilea parametru este opțional.

### Metoda *DOMNode->prefix()*

Metoda *prefix()* nu are parametri și returnează un șir de caractere care reprezintă prefixul spațiului de utilizare al nodului curent.

### Metoda *DOMNode->parent\_node()*

Această metodă returnează un obiect derivat din *DOMNode* care reprezintă părintele nodului curent. În structura arborescentă asociată unui document există un singur nod care nu are părinte, și anume nodul de tipul *DomDocument* care este părintele întregului arbore.

### Metoda *DOMNode->parent()*

Această metodă nu primește nici un parametru și returnează un obiect derivat din *DOMNode* care reprezintă părintele nodului curent. În structura arborescentă asociată unui document există un singur tip nod care nu are părinte, și anume *DomDocument* care este părintele întregului arbore.

Constantă	Valoare	Clasă asociată
<i>XML_ELEMENT_NODE</i>	1	<i>DomElement</i>
<i>XML_ATTRIBUTE_NODE</i>	2	<i>DomAttribute</i>
<i>XML_TEXT_NODE</i>	3	<i>DomText</i>
<i>XML_CDATA_SECTION_NODE</i>	4	<i>DomCDATA</i>
<i>XML_ENTITY_REF_NODE</i>	5	<i>DomEntityReference</i>
<i>XML_ENTITY_NODE</i>	6	<i>DomEntity</i>
<i>XML_PI_NODE</i>	7	<i>DomProcessingInstruction</i>
<i>XML_COMMENT_NODE</i>	8	<i>DomComment</i>
<i>XML_DOCUMENT_NODE</i>	9	<i>DomDocument</i>
<i>XML_DOCUMENT_TYPE_NODE</i>	10	<i>DomDocumentType</i>
<i>XML_DOCUMENT_FRAG_NODE</i>	11	<i>DomDocumentFragment</i>
<i>XML_NOTATION_NODE</i>	12	<i>DomNotation</i>

Tabelul 1: Constantele asociate diferitelor tipuri de noduri

### Metoda *Dom...->owner\_document()*

Metoda *owner\_document()* nu are parametri și returnează un obiect de tipul *DomDocument* care reprezintă părintele structurii căreia îi aparține nodul curent.

### Metoda *DOMNode->clone\_node()*

Această funcție are un singur parametru de tip logic și realizează duplicare conținutului nodului curent. Dacă parametru are valoarea **TRUE**, atunci este duplicat întregul subarbore care are ca părinte nodul curent.

În cazul în care parametru are valoarea **FALSE** sau lipsește, se duplică numai tipul nodului și atributele.

Metoda *clone\_node()* returnează un obiect de tipul celui curent.

### Metoda *DOMNode->has\_child\_nodes()*

Această metodă nu are parametri și returnează o valoare logică indicând faptul că nodul curent are sau nu fii.

Atributele unui nod sunt accesate folosind alte metode pe care le vom prezenta în cadrul acestui articol.

Există tipuri de noduri care sunt frunze ale structurii arborescente, deci care nu pot avea fii.

### Metoda *DOMNode->first\_child()*

Metoda *first\_child()* nu are parametri și returnează un obiect care reprezintă primul fiu al nodului curent, dacă există, și valoarea logică **FALSE** în caz contrar.

### Metoda *DOMNode->last\_child()*

Metoda *last\_child()* nu are parametri și returnează un obiect care reprezintă ultimul fiu din lista de fii a nodului curent, dacă un astfel de fiu există, și valoarea logică **FALSE** în caz contrar.

### Metoda *DOMNode->child\_nodes()*

Metoda *child\_nodes()* nu are parametri și returnează un tablou care conține toți fiii nodului curent.

În cazul în care nodul curent nu are fii, metoda returnează valoarea logică **FALSE**.

### Metoda *DOMNode->append\_child()*

Această metodă primește ca parametru un obiect care are un tip derivat din *DOMNode* și îl adaugă la sfârșitul listei de fii.

### Metoda *DOMNode->insert\_before()*

Această metodă inserează un obiect în lista de fii.

Metoda primește două obiecte ca parametru.

Primul parametru reprezintă obiectul care va fi inserat, iar al doilea reprezintă o referință spre obiectul înaintea căruia se va face inserarea.



**Metoda DomNode->replace\_child()**

Această metodă se folosește pentru a schimba valoarea unui fiu cu un alt obiect.

Metoda are doi parametri de tip obiect. Primul parametru reprezintă o referință spre obiectul care este fiu al nodului curent, iar cel de-al doilea parametru reprezintă obiectul cu care fiul va fi înlocuit.

**Metoda DomNode->remove\_child()**

Metoda `remove_child()` se folosește pentru a elimina un nod al fiului curent.

Metoda primește ca parametru o referință spre fiul care va fi eliminat din listă.

**Metoda DomNode->has\_attributes()**

Această metodă nu are parametri și returnează o valoare logică indicând faptul că nodul curent are sau nu atribute.

Există tipuri de noduri care nu au atribute.

**Metoda DomNode->attributes()**

Metoda `child_nodes()` nu are parametri și returnează un tablou care conține obiecte de tipul *DomAttribute* care reprezintă atributele nodului curent. În cazul în care nodul curent nu are atribute, metoda returnează valoarea logică **FALSE**.

**Metoda DomNode->previous\_sibling()**

Această metodă nu are parametri și returnează un obiect care reprezintă nodul care se află în lista de fii a părintelui înaintea nodului curent.

Dacă nodul curent este primul nod din lista de fii a părintelui său, atunci funcția returnează valoarea logică **FALSE**.

**Metoda DomNode->next\_sibling()**

Această metodă nu are parametri și returnează un obiect care reprezintă nodul care se află în lista de fii a părintelui după nodul curent.

Dacă nodul curent este ultimul din lista de fii a părintelui său, atunci funcția returnează valoarea logică **FALSE**.

Metodele pe care le-am prezentat anterior care se referă la fiii și frații unui nod al arborelui unui document pot fi utilizate pentru a naviga în interiorul structurii arborescente.

**Clasa DomDocument**

Această clasă este derivată din clasa *DomNode* și conține metode necesare pentru a crea diferite tipuri de noduri care pot fi atașate ulterior structurii arborescente, pentru a procesa restricțiile pe care trebuie să le respecte arborele unui document.

Limbajul *PHP* oferă funcții pentru a crea o structură arborescentă vidă sau o structură arborescentă a unui document stocat într-un fișier pe disc sau într-un șir de caractere. În continuare vă prezentăm aceste funcții, urmând

ca după aceea să prezentăm metodele clasei *DomDocument*.

**Funcția domxml\_new\_doc()**

Această funcție creează un document *XML* nou și returnează un obiect de tipul *DomDocument*. Funcția primește un singur parametru de tipul șir de caractere care reprezintă versiunea tipului de fișier *XML* și o valoare posibilă pentru acesta este "1.0".

**Funcția domxml\_open\_file()**

Această funcție creează arborele unui document *XML* pe baza informațiilor stocate într-un fișier și returnează un obiect de tipul *DomDocument*.

**Funcția domxml\_open\_mem()**

Această funcție creează arborele unui document *XML* pe baza informațiilor stocate într-un șir de caractere și returnează un obiect de tipul *DomDocument*.

**Metoda DomDocument->create\_attribute()**

Metoda `create_attribute()` creează și returnează un obiect de tipul *DomAttribute* care reprezintă un atribut care poate fi asociat unui nod din structura arborescentă.

Această metodă primește ca parametru două șiruri de caractere. Primul parametru reprezintă numele atributului care va fi creat, iar cel de-al doilea reprezintă valoarea acestuia.

**Metoda DomDocument->create\_cdata\_section()**

Această metodă creează și returnează un obiect de tipul *DomCDATA* care reprezintă un nod a cărui valoare este de tip șir de caractere și al cărui nume este "#cdata-section".

Metoda `create_cdata_section()` are un singur parametru de tip șir de caractere care va constitui valoarea nodului care va fi creat.

Acest nod poate fi atașat structurii arborescente prin intermediul metodelor `append_child`, `insert_before` sau `replace_child` apelată pentru un nod din arbore.

În cadrul unui fișier *XML* acest tip de nod este reprezentat astfel:

```
<![CDATA[valoare]]>
```

unde *valoare* reprezintă valoarea nodului de tipul *DomCDATA*.

**Metoda DomDocument->create\_text\_node()**

Această metodă creează și returnează un obiect de tipul *DomText* care reprezintă un nod a cărui valoare este de tip șir de caractere și al cărui nume este "#text".

Metoda `create_text_node()` are un singur parametru de tip șir de caractere care va constitui valoarea nodului care va fi creat.

Tipul de nod *DomText* este derivat de tipul *DomCDATA* și diferă prin acesta numai prin modul în care sunt reprezentate datele într-un fișier.

**Metoda DomDocument->create\_comment()**

Metoda `create_comment()` creează și returnează un obiect de tipul *DomComment* care reprezintă un comentariu în structura arborescentă și primește ca parametru un șir de caractere care reprezintă textul care va fi asociat nodului care va creat.

Tipul de nod *DomComment* este derivat de tipul *DomCDATA* și diferă prin acesta numai prin modul în care sunt reprezentate datele într-un fișier.

În cadrul unui fișier XML acest tip de nod este reprezentat astfel:

```
<!-- valoare -->
```

unde *valoare* reprezintă valoarea nodului de tipul *DomComment*.

**Metoda DomDocument->create\_element()**

Această metodă creează și returnează un obiect de tipul *DomElement*. Metoda `create_element()` are un singur parametru de tip șir de caractere care reprezintă numele nodului de tipul *DomElement* care va fi creat.

Acest tip de nod nu are valoare dar poate avea atribute și poate avea fii și într-un document XML este reprezentat astfel:

```
<nume atr1="val1" atr2="val2" ... atrn="valn" />
sau
<nume atr1="val1" atr2="val2" ... atrn="valn">
    reprezentare fii
</nume>
```

unde *nume* reprezintă numele nodului de tipul *DomElement*, *atr1*, *atr2*, ..., *atrn* reprezintă atributele nodului și *val1*, *val2*, ..., *valn* reprezintă valorile asociate atributelor.

Dacă un nod de tipul *DomElement* nu are fii este folosită prima sintaxă și în caz contrar cea de-a doua.

**Metoda DomDocument->create\_element\_ns()**

Această metodă creează și returnează un obiect de tipul *DomElement* căruia i se va asocia un nume, un identificator de resursă și un domeniu de utilizare.

Metoda `create_element_ns()` are trei parametri de tip șir de caractere care reprezintă identificatorul resursei constituită de acest obiect, numele nodului, respectiv prefixul domeniului de utilizare al nodului.

**Metoda DomDocument->create\_entity\_reference()**

Metoda `create_entity_reference()` creează și returnează un obiect de tipul *DomEntityReference*. Această metodă primește ca parametru un șir de caractere care reprezintă valoarea asociată unui nod de tipul *DomEntityReference*.

Un astfel de nod nu are nume, ci doar valoare, nu poate avea atribute și nici fii, deci putem considera că acest tip de nod poate fi doar o frunză a arborelui unui document.

În cadrul unui fișier XML un nod de tipul *DomEntityReference* este reprezentat astfel:

```
&valoare;
```

unde *valoare* reprezintă valoarea nodului.

**Metoda DomDocument->create\_processing\_instruction()**

Metoda `create_processing_instruction()` creează și returnează un obiect de tipul *DomProcessingInstruction*.

Un nod de tipul *DomProcessingInstruction* reprezintă o instrucțiune de procesare care are o țintă și date.

Această metodă are doi parametri de tip șir de caractere care reprezintă ținta instrucțiunii de procesare, respectiv datele.

În cadrul unui fișier XML, un nod de tipul *DomProcessingInstruction* nu poate avea fii și nici atribute.

Acest tip de nod este reprezentat într-un fișier XML, astfel:

```
<?țintă date?>
```

unde *țintă* reprezintă ținta instrucțiunii de procesare, iar *date* reprezintă datele instrucțiunii de procesare.

**Metoda DomDocument->document\_element()**

Un nod de tipul *DomDocument* poate avea ca fiu un singur element de tipul *DomElement*, și pentru a ușura munca programatorului există o metodă care returnează o referință la acest element și anume `document_element()` care nu are nici un parametru.

**Metoda DomDocument->doctype()**

Această metodă returnează un obiect de tipul *DomDocumentType* care reprezintă tipul unui document XML, adică restricțiile pe care trebuie să le respecte arborele documentului dacă astfel de restricții sunt definite.

**Metoda DomDocument->dump\_file()**

Metoda `dump_file()` nu este prevăzută în standardul DOM, dar ea este utilizată pentru a salva arborele unui document sub forma unui fișier XML.

Această metodă are trei parametri. Primul parametru este de tip șir de caractere și reprezintă numele fișierului în care se va salva structura arborescentă.

Al doilea parametru este opțional, este de tip logic și indică faptul că se va folosi sau nu compresie pentru a salva arborele documentului, iar al treilea parametru este opțional, este de tip logic și indică faptul că se va folosi sau nu formatarea documentului în momentul salvării.

Metoda returnează o valoare logică **TRUE** dacă s-a reușit salvarea arborelui în fișierul specificat ca parametru, și valoarea logică **FALSE** în caz contrar.

**Metoda DomDocument->dump\_mem()**

Metoda `dump_mem()` nu este prevăzută în standardul DOM, dar ea este utilizată pentru a salva arborele unui document sub forma unui șir de caractere.



Această metodă are doi parametri opționali, de tip logic. Acești parametri au aceeași semnificație cu ultimii doi parametri ai metodei prezentate anterior.

Metoda `dump_mem()` returnează un șir de caractere care reprezintă documentul *XML*.

### Clasa *DomCDATA* și clasele derivate

Nodurile de tipul *DomCDATA*, *DomText* și *DomComment* nu pot avea atribute și nici fii, astfel ele sunt frunze ale arborelui unui document.

Clasele *DomText* și *DomComment*, derivate din clasa *DomCDATA*, nu au metode proprii, iar clasa *DomCDATA*, pe lângă metodele moștenite de la clasa *DomNode*, introduce metoda `length()` care returnează dimensiunea datelor conținute.

### Clasa *DomElement*

Într-un document *XML*, în afară de restricțiile care trebuie să le respecte arborele documentului, există un singur tip de nod care poate fi nod intern. Este vorba despre *DomElement*. Tot *DomElement* este tipul de nod care poate avea atribute.

În continuare vă prezentăm metodele pe care clasa *DomElement* le introduce pe lângă cele moștenite de la clasa *DomNode*.

#### Metoda *DomElement->has\_attribute()*

Această metodă are un singur parametru de tip șir de caractere și returnează valoarea logică **TRUE** dacă nodul curent are setat atributul cu numele primit ca parametru și valoarea logică **FALSE** în caz contrar.

#### Metoda *DomElement->get\_attribute\_node()*

Această metodă primește ca parametru un șir de caractere și returnează atributul nodului curent cu numele primit ca parametru, sub forma unui obiect de tipul *DomAttribute*. Dacă nodul curent nu are acest atribut setat, metoda returnează valoarea logică **FALSE**.

#### Metoda *DomElement->get\_attribute()*

Această metodă primește ca parametru un șir de caractere și returnează un șir de caractere care reprezintă valoarea atributului nodului curent cu numele primit ca parametru. Dacă nodul curent nu are acest atribut setat, metoda returnează valoarea logică **FALSE**.

#### Metoda *DomElement->set\_attribute()*

Metoda `set_attribute()` se folosește pentru a atașa nodului curent un atribut.

Metoda primește doi parametri de tipul șir de caractere. Primul parametru reprezintă numele atributului, iar al doilea reprezintă valoarea pe care acesta o va avea.

Dacă nodul curent nu are setat atributul cu numele primit ca parametru, atunci va fi creat un nod de tipul *DomAttribute* și va fi adăugat în lista de atribute a nodului curent. În caz contrar va fi schimbată valoarea pe care o are

nodul de tipul *DomAttribute* corespunzător numelui primit ca parametru.

#### Metoda *DomElement->remove\_attribute()*

Metoda `remove_attribute()` se folosește pentru a șterge un atribut din lista de atribute a nodului curent. Metoda primește ca parametru numele atributului care va fi șters.

În cazul în care ștergerea s-a efectuat cu succes, metoda returnează valoarea logică **TRUE**, altfel returnează valoarea logică **FALSE**.

#### Metoda *DomElement->tagname()*

Metoda `tagname()` returnează numele nodului curent și are același efect cu apelul metodei `node_name()` moștenită de la clasa de bază *DomNode*.

#### Metoda *DomElement->get\_elements\_by\_tagname()*

Această metodă returnează un tablou unidimensional care conține obiectele de tipul *DomElement* care sunt noduri ale subarborelui al cărui părinte este nodul curent și al căror nume este același cu singurul parametru de tip șir de caractere pe care metoda îl primește ca parametru.

În cazul în care subarborele nu conține un astfel de nod atunci metoda `get_elements_by_tagname()` returnează valoarea logică **FALSE**.

### Clasa *DomAttribute*

Această clasă reprezintă orice atribut pe care un nod din structura arborescentă îl poate avea. Orice nod al arborelui unui document reține două liste de fii. Prima listă conține toate atributele, iar cea de-a doua listă conține restul fiilor.

În continuare vă prezentăm metodele pe care această clasă le introduce pe lângă cele moștenite de la clasa *DomNode*.

#### Metoda *DomAttribute->name()*

Metoda `name()` returnează numele atributului curent și are același efect cu apelul metodei `node_name()` moștenită de la clasa de bază *DomNode*.

#### Metoda *DomAttribute->value()*

Metoda `value()` returnează numele atributului curent și are același efect cu apelul metodei `value()` moștenită de la clasa de bază *DomNode*.

#### Metoda *DomAttribute->specified()*

În cazul în care restricțiile pe care trebuie să le respecte arborele unui document *XML* permit ca un anumit atribut să nu fie specificat, și pentru nodul de tip *DomElement* căruia îi aparține obiectul curent, atributului nu este specificat atunci această metodă returnează valoarea logică **FALSE**. În caz contrar metoda returnează valoarea logică **TRUE**.

### Exemple de prelucrare XML

În continuare vă prezentăm câteva exemple de script-uri *PHP* care utilizează fișiere *XML*. Pentru a realiza aceste exemple am folosit fișierul *XML* din figura 1.

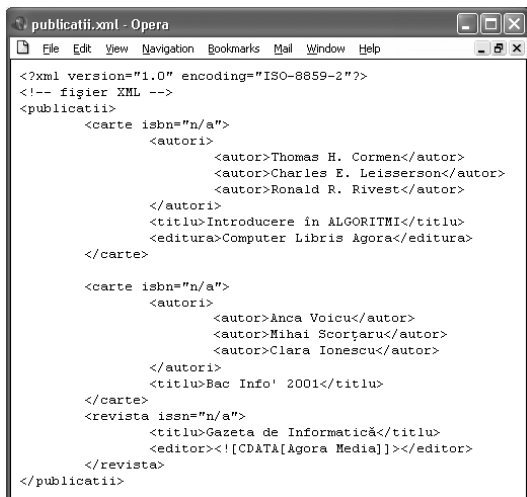


Figura 1: Fișierul publicatii.xml

### Exemplul 1

Script-ul PHP următor ilustrează modul de utilizare a metodei `get_elements_by_tagname()` și a metodei `get_attribute_node()` aparținând clasei `DomElement`, iar în figura 2 se poate observa rezultatul execuției acestuia.

```
<HTML>
<HEAD>
  <TITLE>
    Exemplu get_elements_by_tagname()
  </TITLE>
</HEAD>
<BODY>
  <?PHP
function main() {
  $dom = domxml_open_file("publicatii.xml");
  if (!$dom) {
    echo "Eroare la accesarea fișierului";
    return;
  }
  $root = $dom->document_element();
  $carti = $root->get_elements_by_tagname(
    "carte");
  foreach($carti as $carte) {
    $titlu = $carte->get_elements_by_
      tagname("titlu");
    echo "<B>". $titlu[0]->get_content().
      "</B><BR>";
    $isbn = $carte->get_attribute_node(
      "isbn");
    echo "ISBN: ". $isbn->node_value().
      "<BR>";
    $autori = $carte->get_elements_by_
      tagname("autor");
    if (sizeof($autori) != 0) {
      echo "Autori: ";
      for ($i=0; $i<sizeof($autori)-1; $i++)
        echo $autori[$i]->get_content(). ", ";
      echo $autori[$i]->get_content(). "<BR>";
    }
  }
}
```

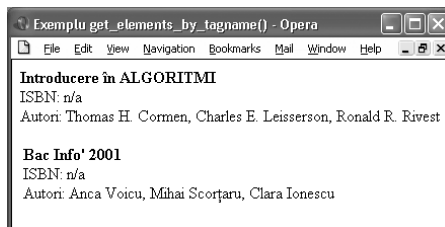


Figura 2: Exemplu `get_elements_by_tagname()`

```
}
}
}
main();
?>
</BODY>
</HTML>
```

### Exemplul 2

Script-ul PHP următor ilustrează modul de utilizare a metodei `create_` aparținând clasei `DomDocument`, a metodei `append_child()` aparținând clasei `DomNode` și a metodei `set_attribute()` aparținând clasei `DomElement`.

În figura 3 se poate observa rezultatul execuției script-ului.

```
<HTML>
<HEAD>
  <TITLE>
    Exemplu prelucrare fișier XML
  </TITLE>
</HEAD>
<BODY>
  <?PHP
  $dom = domxml_open_file("publicatii.xml");
  if (!$dom) {
    echo "Eroare la accesarea fișierului";
    return;
  }
  $root = $dom->document_element();
  $revista = $dom->create_element("revista");
  $revista->set_attribute("issn", "n/a");
  $titlu = $dom->create_element("titlu");
  $titluc = $dom->create_text_node(
    "PC Magazine");
  $titlu->append_child($titluc);
  $editor = $dom->create_element("editor");
  $editorc = $dom->create_text_node(
    "Agora Media");
  $editor->append_child($editorc);
  $revista->append_child($titlu);
  $revista->append_child($editor);
  $root->append_child($revista);
  $s = $dom->dump_mem(false);
  echo "<PRE><CODE>". htmlspecialchars($s).
    "</CODE></PRE>";
  ?>
```



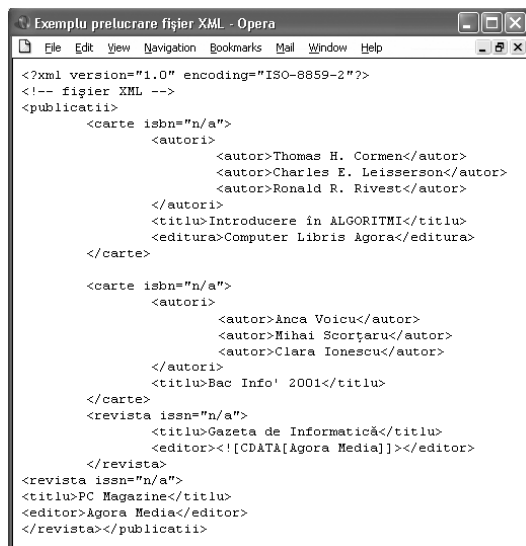


Figura 3: Exemplu de prelucrare fișier xml

&lt;/BODY&gt;

&lt;/HTML&gt;

### Exemplul 3

Script-ul PHP următor ilustrează modul de navigare prin structura arborescentă asociată unui document XML. Pentru fiecare nod am afișat numele, valoarea și, unde a fost cazul, atributele. În figura 4 se poate observa rezultatul execuției script-ului.

```
<HTML>
<HEAD>
  <TITLE>
    Parcurgere XML recursiv
  </TITLE>
</HEAD>
<BODY>
  <?PHP
function main() {
  $dom = domxml_open_file("publicatii.xml");
  if (!$dom) {
    echo "Eroare la accesarea fișierului";
    return;
  }
  echo "<B><PRE><CODE>";
  printRecursive($dom, 0);
  echo "</CODE></PRE></B>";
}

function printPadding($level) {
  for($i; $i<$level; $i++)
    echo "--";
  echo "@- ";
}

function printBlanks($level) {
  for($i; $i<=$level; $i++)
    echo "&nbsp;&nbsp;&nbsp;";
}
```

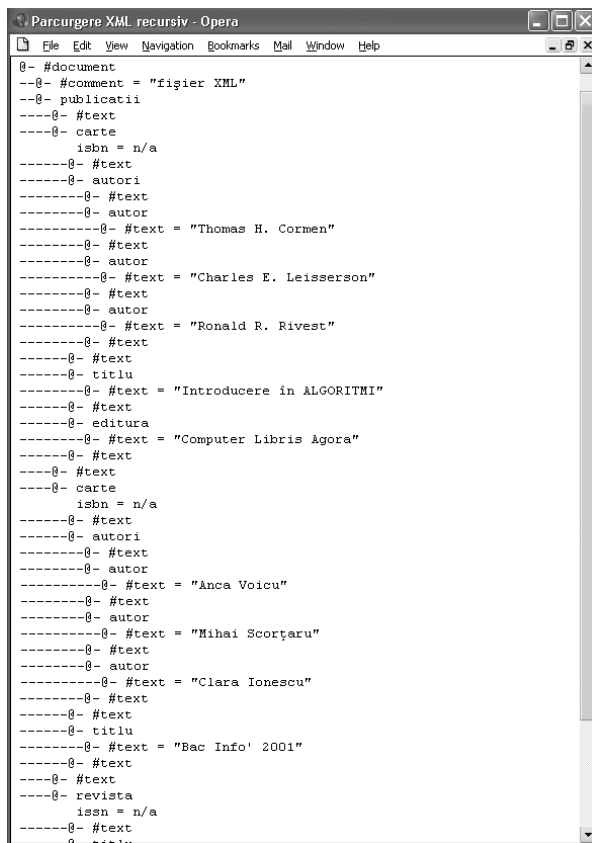


Figura 4: Exemplu de parcurgere recursivă a unui fișier XML

```
echo "&nbsp;&nbsp;&nbsp;";
}

function printRecursive($node, $level) {
  printPadding($level);
  echo $node->node_name();
  $s = trim($node->node_value());
  if (($s!="") && (isset($s)))
    echo " = \"".$s."\"";
  echo "<BR>";
  if ($node->has_attributes()) {
    printBlanks($level);
    $attrs = $node->attributes();
    foreach($attrs as $attr)
      echo $attr->name() . " = " . $attr->value();
  }
  echo "<BR>";
  $cs = $node->child_nodes();
  foreach($cs as $c)
    printRecursive($c, $level+1);
}

main();
?>
</BODY>
</HTML>
```