XML – 16.05.2023

Documentele XML sunt realizate din unitati de stocare numite entitati, ce contin date parsate sau neparsate. Datele parsate sunt realizate din caractere, unele dintre ele formand date caracter iar altele ca marcaje. Marcajele codifica o descriere a schemei de stocare a documentului si structura logica. XML furnizeaza un mecanism pentru a impune constringeri asupra schemei de stocare si a structurii logice.

XML - eXtensible Markup Language- este un limbaj pentru crearea altor limbaje.
Are o structura bine definite.
Poate fi utilizat pentru a crea limbaje de marcare precum HTML, XHTML

Scopul limbajului XML:

- XML a fost elaborat pentru:
- Separarea sintaxei de semantica pentru a furniza un cadru comun de structurare a informatiei
- construirea de limbaje de mark-up pentru aplicatii din orice domeniu structurarea informatiei in viitor
- asigurarea independentei de platforma si suport pentru internationalizare

XML vs. HTML

Exemplu HTML

```
<html>
<body>
<h2> Student</h2>
<p>Informatii
</p>
</body>
</html>
```

HTML -

- Specifica modul de randare a documentului, si nu ce tip de informatie este continuta in document
- Este dificil pentru o masina sa extraga informatia continuta inpagina. Este relativ simplu pentru om

XML vs. HTML

Sa consideram acum urmatoarea reprezentare

```
<contact>
<name>Student</name>
<address>Bucuresti</address>
<web-page>http://www.utm.ro</web-page>
<email>test@s.utm.ro</email>
</contact>
```

In acest caz

- Informatia continuta este pusa in evidenta si nu modul de randare
- Continutul este separat de prezentare
- Informatia poate fi "inteleasa" atat de oameni cat si de masini

Concluzii:

- HTML este utilizat pentru a marca textul astfel incat el sa poata fi afisat
- XML este utilizat pentru marcarea datelor astfel incat ele sa poata fi procesate automat de calculator
- HTML descrie atat structura (ex. <p>, <h2>,) cat si modul de reprezentare (ex. <b>, <font>, <i>)
- XML descrie numai continutul sau "intelesul"
- HTML utilizeaza un set fix, neschimbat, de tag-uri
- In XML putem sa definim propriile tag-uri

Avantajele utilizarii limbajului XML

- Este un limbaj bazat standard open
- Este un limbaj extensibil
- Este transformabil
- Disponibilitatea unui numar mare de unelte de dezvoltare

XML a fost elaborat pentru:

- separarea **sintaxei** de **semantica** pentru a furniza un cadru comun de structurare a informatiei
- **construirea de limbaje de mark-up** pentru aplicatii din orice domeniu
- **structurarea informatiei** in viitor
- **asigurarea independentiei** de platforma si suport pentru **internationalizare**

Un document XML este un **arbore ordonat etichetat**:

- **date caracter** - noduri frunza ce contin datele
- noduri **elemente** etichetate cu
  - un nume (adesea numit si tipul elementului) si
  - o multime de atribute, fiecare din ele avand un nume si o valoare

  acestea pot contine unu sau mai multi copii.

Structura documentelor XML

Un document XML este format din:

- marcaje (tag-uri)
- date caracter

Un *marcaj (tag)* este un sir de caractere delimitat de caracterele "<" si ">". *Datele caracter* reprezinta continutul marcajelor.

Un fisier XML cuprinde urmatoarele sectiuni:

- Prolog
- Definitia tipului de document (optionala)
- Elementul radacina

Exemplu: Fisierul mail.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MAIL SYSTEM "MAIL.DTD">
<MAIL id="10" date="10-05-2021">
        <FROM>nume_prof@prof.utm.ro</FROM>
        <TO>student@s.utm.ro</TO>
        <SUBJECT>Hello</SUBJECT>
        <MESSAGE>
        Hello Student
        </MESSAGE>
</MAIL>
```

Prologul:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Este o instructiune de procesare. Ea informeaza ca urmeaza descrierea unui fisier XML ce respecta versiunea de specificatie 1.0 iar setul de caractere utilizat este encodat UTF-8.

Definitia Tipului de Document:

```
<!DOCTYPE MAIL SYSTEM "MAIL.DTD">
```

Precizeaza ca fisierul MAIL.DTD contine declaratia tipului de document (DTD-ul), document ce are ca radacina tag-ul MAIL. Acesta este un set de reguli ce defineste structura unui fisier XML.

Elementul radacina:

```
<MAIL id="10" date="10-05-2020">
    <FROM>profesor@prof.utm.ro</FROM>
    <TO>student@s.utm.ro</TO>
    <SUBJECT>Hello</SUBJECT>
    <MESSAGE>
    Hello Student
    </MESSAGE>

</MAIL>
```

Un document XML are un singur element radacina.

Un document XML poate contine urmatoarele tipuri de marcaje:

- Elemente
- Atribute
- Comentarii
- Entitati
- Sectiuni CDATA
- Instructiuni de procesare
- Declaratia tipului de document

Elementele:

<nume_tag> ..... </nume_tag>

- sunt blocurile de baza ale unui document XML
- Retin informatii sau definesc structura
- Fiecare element are un tag de inceput si unul de sfarsit
- Elementele pot fi imbricate

Elemente vide:

<nume_tag/>

Exemplu

```
<?xml version="1.0"?>
<BIBLIOTECA>
<CARTE>
<TITLU>XML </TITLU>
<AUTOR>Eminescu</AUTOR>
<EDITURA>Art Educational</EDITURA>
<AN_APARITIE>2019</AN_APARITIE>
</CARTE>
</BIBLIOTECA>
```

Se observa ca

- elementele `<TITLU>`, `<AUTOR>`, `<EDITURA>`, `<AN_APARITIE>` contin informatii
- `<BIBLIOTECA>`, `<CARTE>` sunt folosite doar pentru a defini structura datelor

Atributele:

```
<nume_tag numeAtr1="val1" ... numeAtrN="valN">
        . . .
</nume_tag>
```

- sunt localizate in tag-ul de start al unui element
- au rolul de a descrie elementele
- adesea contin metadate, ex: `id`

```
<?xml version="1.0"?>
<BIBLIOTECA>
<CARTE cota="12345">
<TITLU>XML</TITLU>
<AUTOR>Eminescu</AUTOR>
<EDITURA>Art Educational</EDITURA>
<AN_APARITIE>2019</AN_APARITIE>
</CARTE>
</BIBLIOTECA>
```

- cota="12345" este un atribut

Comentarii:

```
<!-- comentariu -->
```

- sunt secvente de caractere ce pot aparea oriunde in document ignorate la parsare
- Ele nu fac parte din datele caracter ale documentului

Exemplu

```
<?xml version="1.0"?>
<!-- Documentul retine cartile dintr-o biblioteca -->
<BIBLIOTECA>
<CARTE cota="12345">
<!-- titlul cartii -->
<TITLU>XML </TITLU>
<!-- Autorul Cartii -->
<AUTOR>Nume Autor</AUTOR>
<!-- Editura in care a aparut cartea -->
<EDITURA> Editura care a publicat cartea</EDITURA>
<!-- Anul de aparitie a cartii -->
<AN_APARITIE>2019</AN_APARITIE>
```

</CARTE>
</BIBLIOTECA>

Important

Tag-uri sunt *case sesitive*, adica se face distinctia intre litere mari si litere mici.

De exemplu, urmatoarele exemple de taguri sunt gresite:

<Student>NUME**</STUDENT>**
<StudenT>NUME**</student>**

Numele unui tag este o sucesiune de caractere alfa-numerice ce *incepe obligatoriu cu o litera*. Astfel <7nume> este eronat.

Referinte la entitati

Referintele la entitati sunt de fapt pointeri catre entitati. În XML, entitatile sunt unitati de text, unde o unitate poate fi orice, de la un singur caracter la un intreg document sau chiar o referinta la un alt document.

Sintaxa referintelor la entitati este:

***&nume_entitate*;**

'&', urmat de numele entitatii, urmat de ';'

Una dintre cele mai frecvente utilizari ale referintelor la entitati este atunci cand se doreste folosirea unor caractere care ar duce la aparitia unor confuzii pentru analizorul XML si deci care nu ar trebui sa apara in forma lor normala in text. În acest caz exista cinci entitati predefinite in XML:

Table 1. Entitati definite in XML

Entitate	Referinta la entitate
<	<
>	>
&	&
'	'
"	&quote;

În momentul in care analizorul XML intalneste referinta la o entitate in document, el o va substitui cu datele pe care aceasta le refera si va returna documentul cu inlocuirile facute.

Exemplu:

<TITLE>Tom **&** Jerry</TITLE>

dupa analizarea textului de catre analizorul XML, va rezulta:

Tom & Jerry

O alta utilizare frecventa a referintelor la entitati este in cazul in care avem in documentul XML fragmente de text care se repeta. Pentru a nu scrie aceste parti de text de mai multe ori vom defini o entitate care va avea ca valoare acea parte de text si de fiecare data cand fragmentul respectiv apare in document vom folosi referinta la entitate.

Prin folosirea referintelor la entitati se vor obtine documente mai scurte si se va scurta timpul de redactare.

Instructiuni de prelucrare

Instructiunile de prelucrare sunt un tip special de marcaj care contin informatii despre anumite aplicatii ce urmeaza a fi executate. Sintaxa generala a unei instructiuni de procesare ar fi urmatoarea:

<?aplicatie instructiune="valoare" ?>

Încep cu <?, urmat de numele aplicatiei si de specificarea unor elemente ce tin de acea aplicatie si se incheie cu ?>. Numele aplicatiei trebuie sa fie diferit de xml sau XML, sau alte moduri de scriere a acestui cuvant, deoarece cuvintele de acest tip sunt rezervate, urmand a fi standardizate intr-o versiune ulterioara.

Sectiuni CDATA

Sectiunile CDATA sunt utilizate pentru a include blocuri de text continand caractere care altfel ar fi recunoscute ca marcaje. Sectiunile CDATA incep cu sirul <![CDATA[si se termina cu sirul]]>.

Sectiunile CDATA sunt folosite in general atunci cand dorim ca datele incluse in interiorul lor sa nu fie interpretate de catre analizor, ci sa fie considerate date caracter.

```
<?xml version="1.0" encoding="UTF-8"?>
<exemplu>
Un exemplu de creare a unui tabel in HTML:
 <![CDATA[
      <table align=”center”>
           <tr>
              <td>Coloana 1</td>
              <td>Coloana 2</td>
           </tr>
      </table>
]]>;
</exemplu>
```

Folosind sectiunea CDATA, analizorul va ignora continutul acesteia si datele vor fi expuse utilizatorului exact in forma in care sunt, si datele nu vor fi interpretate drept marcaje, ci drept date caracter.

O restrictie de sintaxa este faptul ca in interiorul sectiunilor CDATA nu poate sa apara sirul ']]'. Înca un lucru de retinut este ca sectiunile CDATA nu pot fi incluse unele in altele.

```
<?xml version="1.0" encoding="UTF-8"?>
<exemplu>
Un exemplu de creare a unui tabel in HTML:
 <![CDATA[
        <table align=”center”>
                <tr>
                    <td>Coloana 1</td>
                    <td>Coloana 2</td>
                </tr>
        </table>
]]>;
</exemplu>
```

Parsarea XML-urilor este o parte integranta a dezvoltarii web,

In php exista o biblioteca cunoscuta sub numele de **simpleXML**, ale carei facilitati sunt mai mult decat suficiente pentru a acoperi procesarea de baza a XML-urilor.

PHP - XML - SimpleXML

SimpleXML a aparut in PHP 5. Lucreaza ca si DOM, cu obiecte, preia tot documentul XML sub forma unui arbore ierarhic in memorie, dar spre deosebire de acesta, e mai flexibil si foloseste mai putina memorie deoarece elementele sunt stocate direct ca variabile PHP (de tip string si array) si astfel pot fi imediat utilizate. Foloseste un minim necesar de cod si are o forma intuitiva a datelor.

In SimpleXML se lucreaza mult cu functii pentru Array, majoritatea datelor sunt stocate in variabile de tip array.

Este util cand se doreste citirea catorva date dintr-un document XML, a carui structura o cunoasteti, si scrierea altora inapoi.

Citire document XML cu SimpleXML

ex1.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<carti>
 <titlu titlul="Poezii" id="1">
  <autor nume="Mihai Eminescu" />
  <text>Luceafarul</text>
  <text>Ce te legeni...</text>
  <pret suma="10.1" />
 </titlu>
 <titlu titlul="Padurea spanzuratilor" id="2">
  <autor nume="Liviu Rebreanu" />
  <text>Text 1</text>
  <text>Text 2</text>
  <pret suma="0.0" />
 </titlu>
</carti>
```

Urmatorul script preia si afisaza date din acest document XML

afisare1.php

```php
<?php
$obj = simplexml_load_file("ex1.xml");
$titlu = $obj->titlu;
$text = $obj->titlu[0]->text;
// preia intr-un array elementele "autor" si "pret"
for($i=0; $i<count($titlu); $i++) {
  $autor[] = $titlu[$i]->autor;
  $pret[] = $titlu[$i]->pret;
}
echo $autor[0]['nume']. '<br >';
echo $pret[1]['suma']. '<br> Afisare text...<br />';

for($i=0; $i<count($text); $i++) {
  echo $text[$i]. ' <br /> ';
}
?>
```

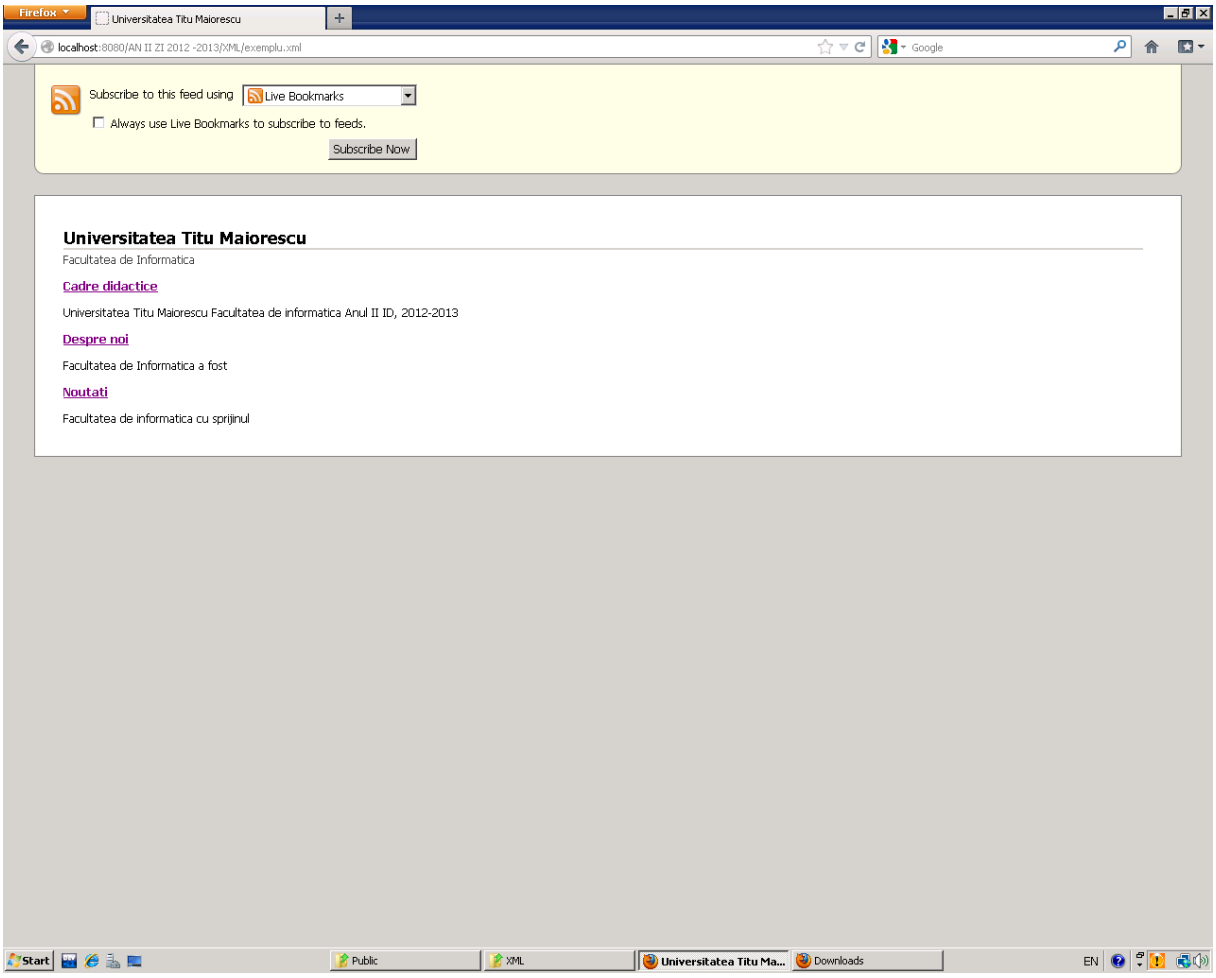
Modificare document XML cu SimpleXML

modificare1.php

```php
<?php
$obj = simplexml_load_file("ex1.xml");
$titlu = $obj->titlu;
 // Preia toate elementele cu nume "titlu" (intr-o variabila tip array)
// Parcurge matricea cu $titlu si adauga, cu "addChild()" inca un element stoc cu val 1
for($i=0; $i<count($titlu); $i++) {
  $titlu[$i]->addChild('stoc', 1);
}
// Modifica valoarea atributului 'nume' din elementul "autor" al primul element "titlu"
$titlu[0]->autor['nume'] = 'Nume Modificat';

// Folosind sistemul de lucru cu array, creaza inca un element "autor" )
$titlu[0]->autor[1] = 'Alt autor';
$titlu[0]->autor[1]['atribut'] = 'Atribut_adaugat';
// Adauga datele XML intr-un sir
$xml_doc = $obj->asXML();
echo htmlentities($xml_doc);
?>
```

Exemplu cu RSS



exemplu.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
    <channel>
    <title>Universitatea Titu Maiorescu</title>
    <description>Facultatea de Informatica</description>
     <link>http://www.utm.ro</link>
     <language>ro</language>
      <item>
      <title><![CDATA[ Informatii Secretariat ]]> </title>
      <description><![CDATA[ Universitatea Titu Maiorescu Facultatea de informatica Anul II ZI, 2020-2021]]>
    </description>
       <link><![CDATA[  https://www.utm.ro/facultatea-de-informatica/informatii-de-secretariat/ll]>
    </link>
     </item>
       <item>
        <title><![CDATA[  Noutati ]]> </title>
        <description><![CDATA[  Facultatea de Informatica a fost ]]> </description>
         <link><![CDATA[  https://www.utm.ro/facultatea-de-informatica/noutati/ll]>
        </link>
        </item>
        <item>
          <title><![CDATA[  Avizier ]]></title>
          <description><![CDATA[  Facultatea de informatica cu sprijinul ]]>
        </description>
           <link><![CDATA[  https://www.utm.ro/facultatea-de-informatica/avizier-studii-zi/ll]>
        </link>
           </item>
        </channel>
    </rss>
```

test.php

```php
<?php
// incarca fisierul exemplu.xml
if(!$xml = simplexml_load_file('exemplu.xml')){
    trigger_error('Eroare la citirea fisierului',E_USER_ERROR);
}

foreach($xml as $channel){
    echo $channel->title."<br>";
    echo $channel->link;
}
?>
```

test1.php

```php
<?php
// incarca fisierul exemplu.xml

if(!$xml = simplexml_load_file('exemplu.xml')){
    trigger_error('Eroare ',E_USER_ERROR);
}

// afiseaza titlul primului element

echo '<div style=color:red>Elementele sunt :<br />';
for($i=0;$i<3;$i++)
{

echo $xml->channel->item[$i]->title.';';
 } echo '</div>';
?>
```

Functii SimpleXML

- **simplexml_load_file("fisier.xml")** - Incarca in memorie, ca obiect, datele din "fisier.xml"
- **simplexml_load_string("sir_xml")** - Incarca in memorie, ca obiect, datele din sirul "sir_xml"
- **simplexml_import_dom("dom_node")** - Transforma (preia) un obiect DOM (sau Nod dintr-un obiect DOM) in obiect SimpleXML
- **addChild("nume", "continut")** - Adauga un element copil in cel curent (la sfarsit daca are si altele) cu numele "nume" si continutul text "continut" (acesta e optional).
- **addAttribute("nume", "valoare")** - Adauga un atribut intr-un element
- **children()** - Preia intr-o matrice toti copii dintr-un nod (element).
- **attributes()** - Preia intr-o matrice toate atributele dintr-un element.
- **count()** - Returneaza numarul de copii dintr-un element.
- **getName()** - Returneaza numele unui element

- **asXML("fisier.xml")** - Transforma datele obiectului SimpleXML intr-un sir, iar daca parametrul "fisier.xml" e specificat (*e optional*) scrie sirul in acel fisier.

Crearea unui fisier XML

a.php

```php
<?php
//creare document XML
$document = new DomDocument('1.0', 'utf-8');
$r = $document->createElement('html');
$r = $document->appendChild($r);
$body = $document->createElement('body');
$body = $r->appendChild($body);
$body->setAttribute('bgcolor', '#ffff00');
$paragraff = $document->createElement('p');
$paragraff = $body->appendChild($paragraff);
$text = $document->createTextNode('Facultatea de informatica');
$text = $paragraff->appendChild($text);
//salveaza fisier si testeaza salvarea pe server
if($document->save("exemplu_d.xml"))
echo 'Documentul exemplu_d.xml a fost creat';
else  echo 'Eroare: nu a putut fi creat documentul exemplu_d.xml ';
?>

```

Citire fisier XML, si crearea unui nou fisier modificat

b.php

```php
<?php
$file = 'exemplu_d.xml';
// Calea si numele fisierului .xml
$doc = new DOMDocument();
// Creaza un nou obiect in memorie
$doc->load($file);
// Incarca datele din $file in obiectul nou creat
$get_elms = $doc->getElementsByTagName("*");
// Preia toate elementele ("*") stocate in obiect
$nr_elms = $get_elms->length;
for($i = 0; $i<$nr_elms; $i++) {   $node = $get_elms->item($i);
  // Preia fiecare Nod din parcurgere
  $element = $node->nodeName;
  if($element=='p') {
    $node->nodeValue = 'MODIFICARE CONTINUT';
// Creaza noul element dupa 'p' (in elementul Parinte unde se afla si 'p'),
// cu nume si continut text
$element_nou = $doc->createElement('div', 'NOUL ELEMENT ADAUGAT');
$node->parentNode->appendChild($element_nou);
  }
}
if($doc->save('exemplu2_dom.xml')) {
  echo htmlentities($doc->saveXML());
}
?>
```