Sesiuni – PHP

Sesiunile, reprezinta o functionalitate prin care anumite informatii sunt mentinute de la o pagina la alta.

O sesiune dureaza atat timp cat utilizatorul acceseaza un site si se incheie odata cu inchiderea browserului.

Sesiuni PHP - reprezinta perioada de timp in care mai multe scripturi PHP, accesate la momente diferite de timp, pot stoca si folosi informatii comune.

O sesiune incepe atunci cand un script apeleaza functia session\_start si se termina cand utilizatorul inchide browserul.

Sesiunile sunt folosite pentru a transporta date de la o pagina la alta. Sesiunile nu sunt mai sigure ca si cookie-urile (deoarece unele servere nu sunt securizate destul de bine si ceilalti utilizatori cu care se imparte serverul poate vedea id-ul sesiunilor).

Pentru securizarea sesiunilor putem utiliza un hash…

Exemplu salvare si afisare date cu ajutorul sesiunilor

Ex\_sesiune.php

<?php

session\_start();//functia trebuie apelata in primele linii din fisier pentru a nu se trimite nici macar un caracter la browser.

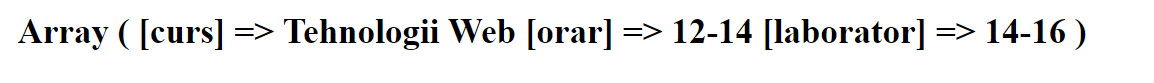
$\_SESSION['curs'] = 'Tehnologii Web';//

$\_SESSION['orar'] = '12-14';

$\_SESSION['laborator'] = '14-16';

echo '<b>'.print\_r($\_SESSION,true).'</b>';

?>



Exemplu 2 pentru a afla valoarea unei variabile din sesiune folosind $\_SESSION[index]

<?php

session\_start();

if(isset($\_SESSION['curs']) ){

// folosim $\_SESSION[index] pentru a afla valoarea unei variabile din sesiune

echo "Cursul <b>{$\_SESSION['curs']}</b> este de la ora <b>{$\_SESSION['orar']}</b> si laboratorul este la ora <b>{$\_SESSION['laborator']}! ";

} else{ echo 'Trebuie sa accesezi mai intai ex\_sesiune.php!';

}

?>

functia **session\_destroy() permite sergerea unei sesiuni;** Stergerea unei variabile din sesiune se realizeaza folosind unset()

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  session\_start();  unset($\_SESSION[curs]);  ?> |

<https://www.php.net/manual/ro/function.session-destroy.php>

**Ce este XML**

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>

<email1>...</email2>

<email2>...</email2>

</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>, <em>) 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

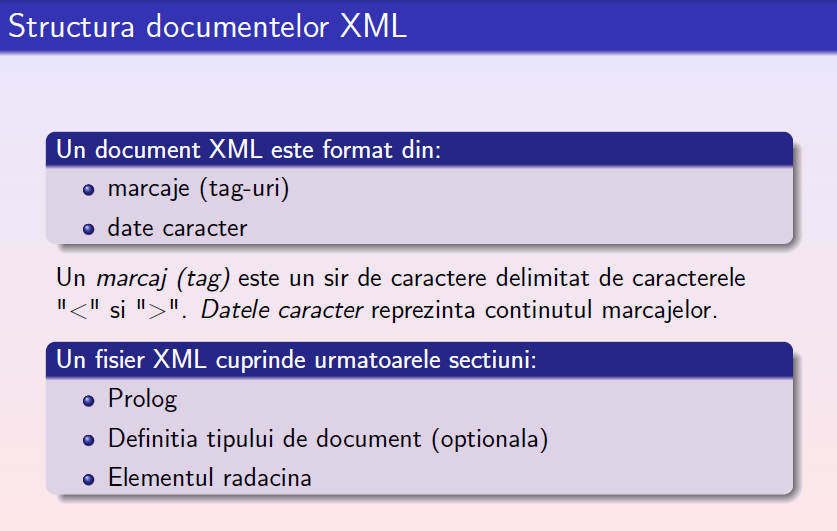
XML a fost elaborat pentru:

* 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.



**Exemplu: Fisierul mail.xml**

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

<!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>

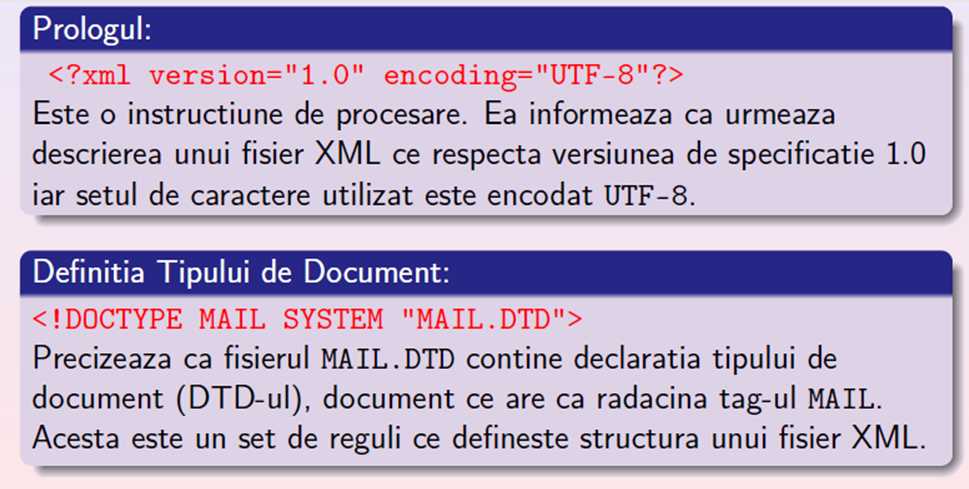
<SUBJECT>Hello</SUBJECT>

<MESSAGE>

Hello Student

</MESSAGE>

</MAIL>



**Elementul radacina:**

<MAIL id="10" date="10-05-2020">

<FROM>profesor@prof.utm.ro</FROM>

<TO>student@s.utm.ro</TO>

<SUBJECT>Hello</SUBJECT>

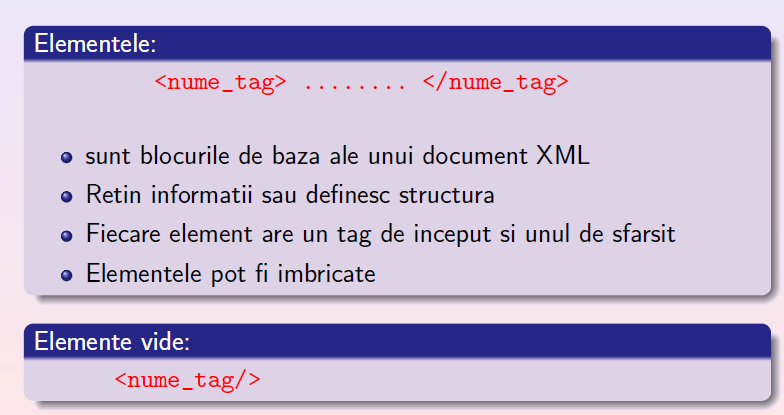
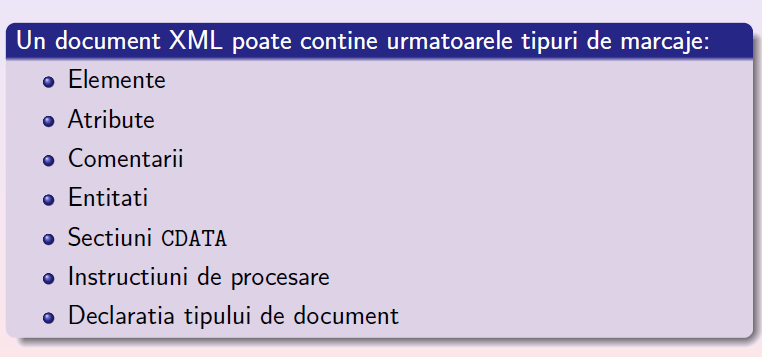
<MESSAGE>

Hello Student

</MESSAGE>

</MAIL>

Un document XML are un singur element radacina.



Exemplu

<?xml version="1.0"?>

<BIBLIOTECA>

<CARTE>

<TITLU>XML </TITLU>

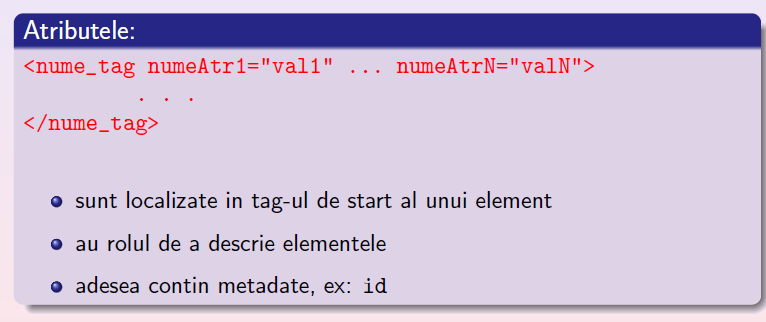
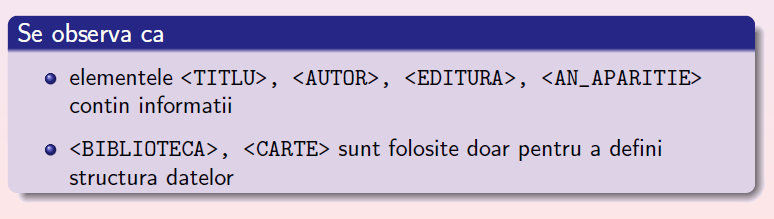
<AUTOR>Eminescu</AUTOR>

<EDITURA>Art Educational</EDITURA>

<AN\_APARITIE>2019</AN\_APARITIE>

</CARTE>

</BIBLIOTECA>



<?xml version="1.0"?>

<BIBLIOTECA>

<CARTE cota="12345">

<TITLU>XML</TITLU>

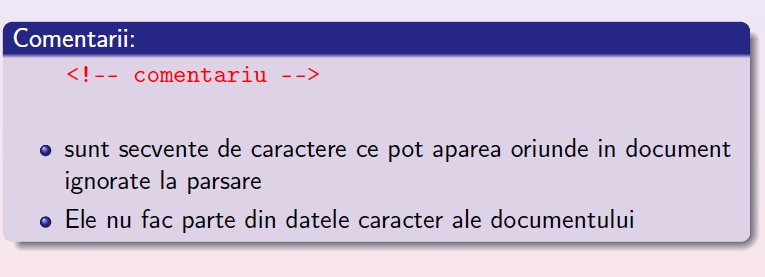
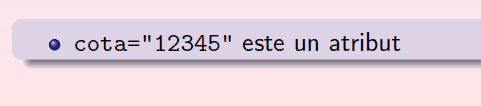
<AUTOR>Eminescu</AUTOR>

<EDITURA>Art Educational</EDITURA>

<AN\_APARITIE>2019</AN\_APARITIE>

</CARTE>

</BIBLIOTECA>



Exemplu

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** |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ' | &apos; |
| " | &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 **&amp;** 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:

**<?aplicatieinstructiune="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

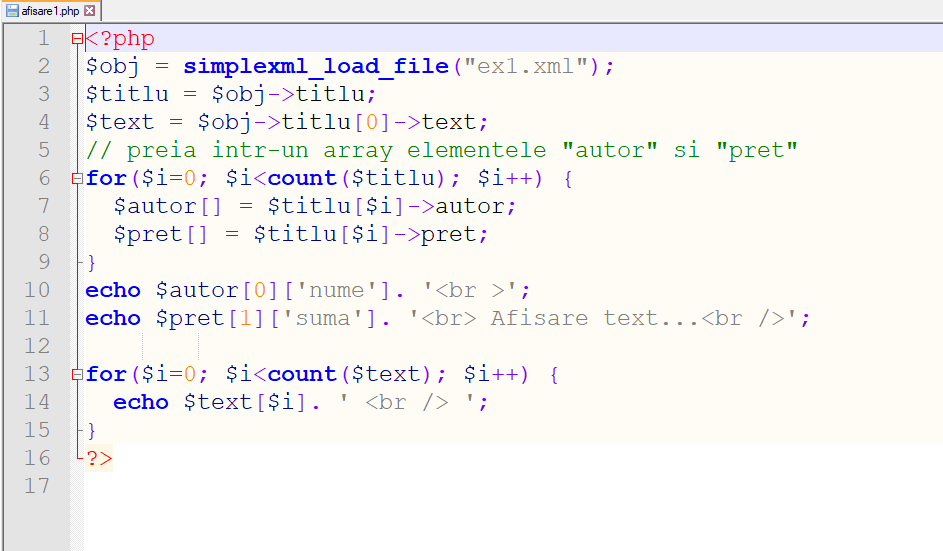
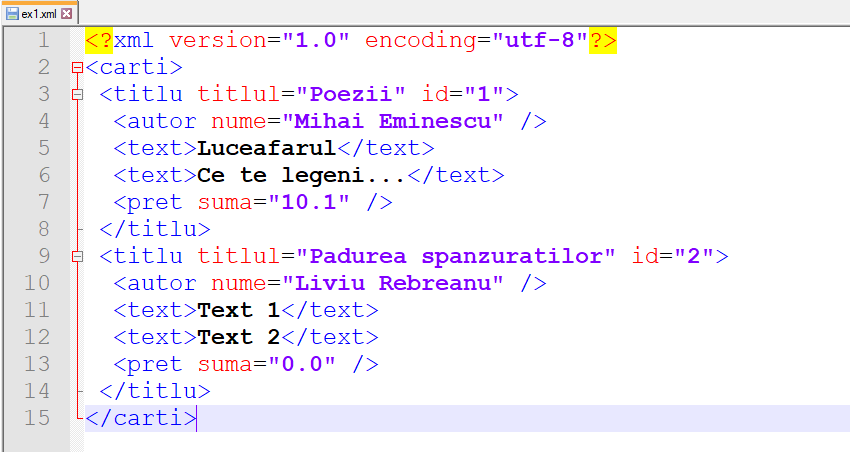
**SimpleXML** a aparut in PHP 5. Lucreaza ca si DOM, cu obiecte, preia tot documentul XML sub forma unui arbore ierarhic in memorie, dar spre deosebire de acesta, e mai flexibil si foloseste mai putina memorie deoarece elementele sunt stocate direct ca variabile PHP (de tip string si array) si astfel 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 altor informatii/date.

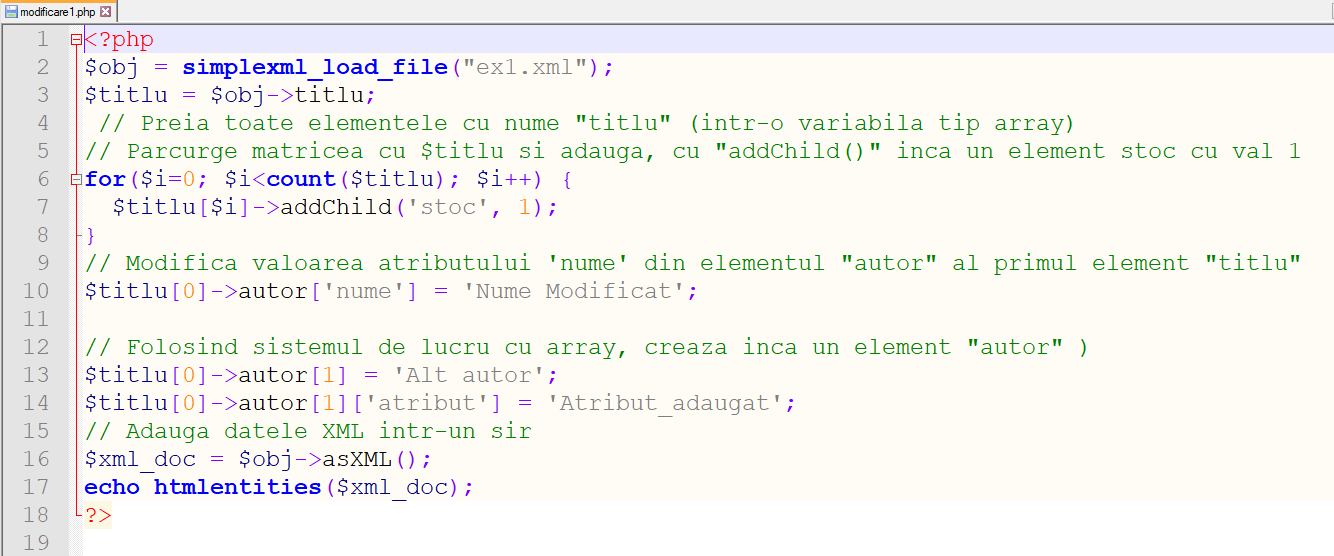
Citire document XML cu SimpleXML

Urmatorul script preia si afisaza date din acest document XML

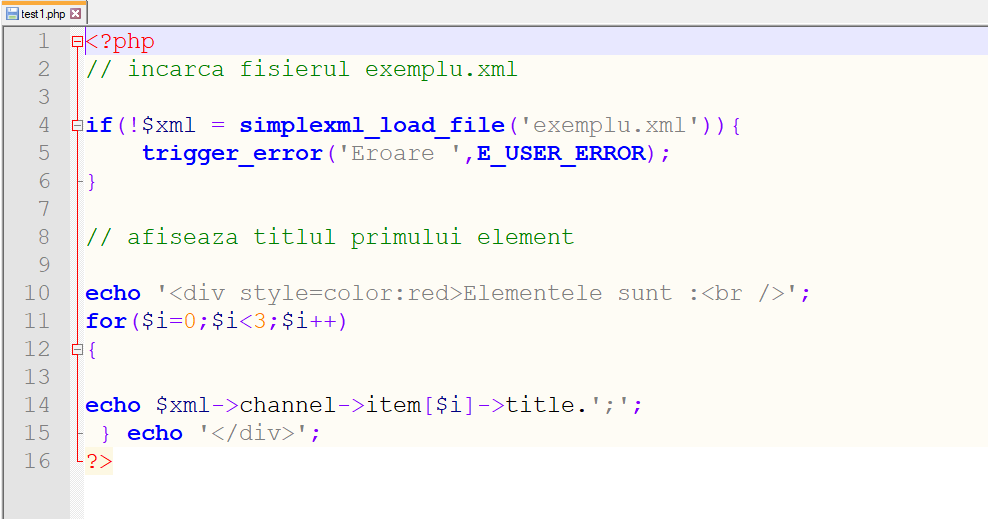
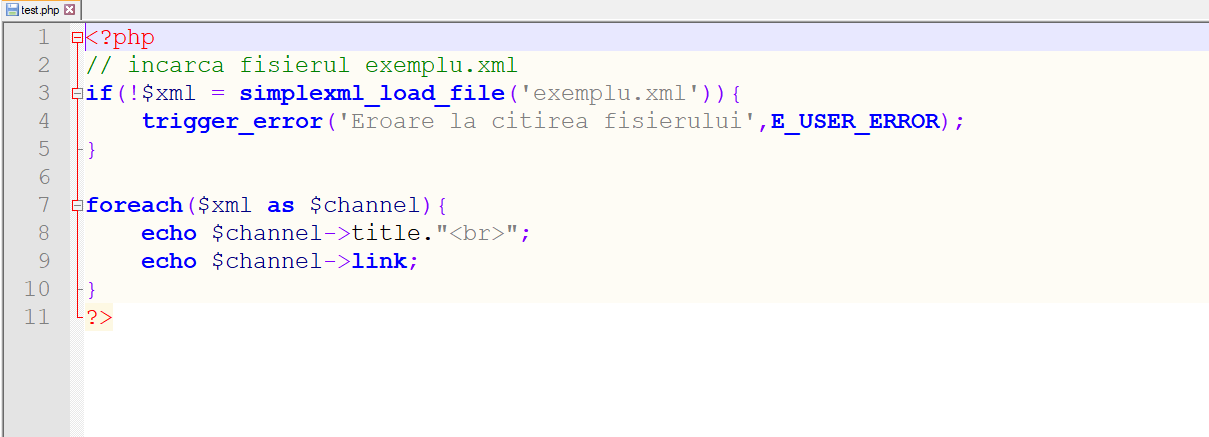
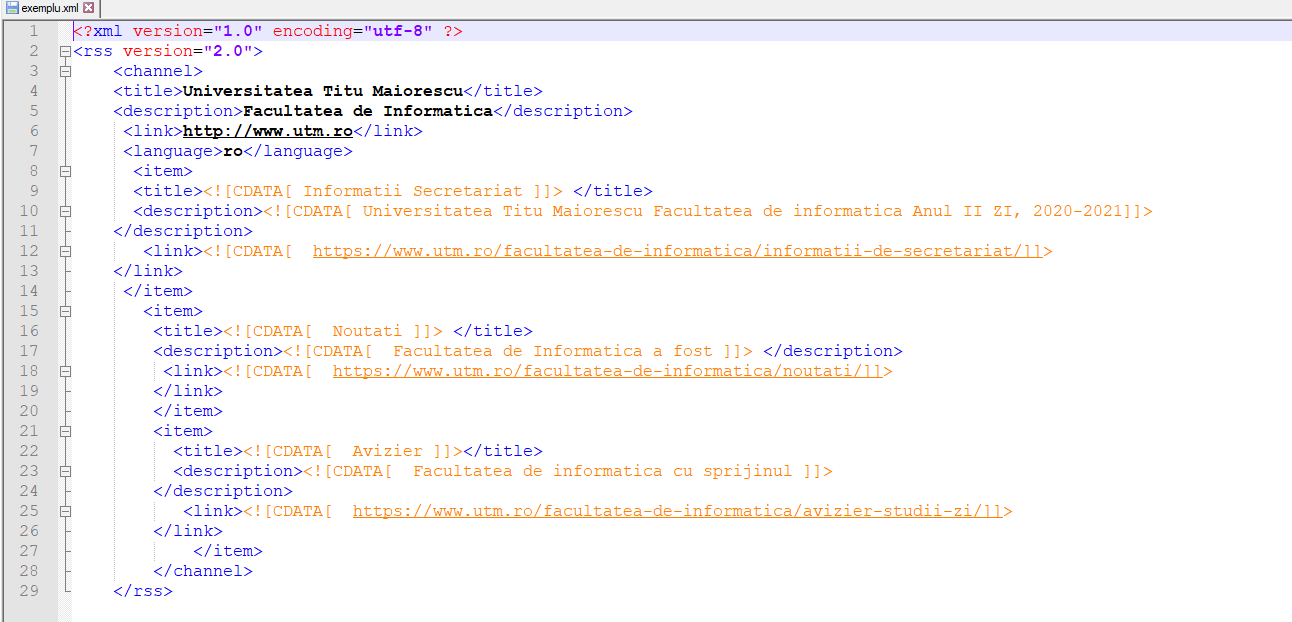
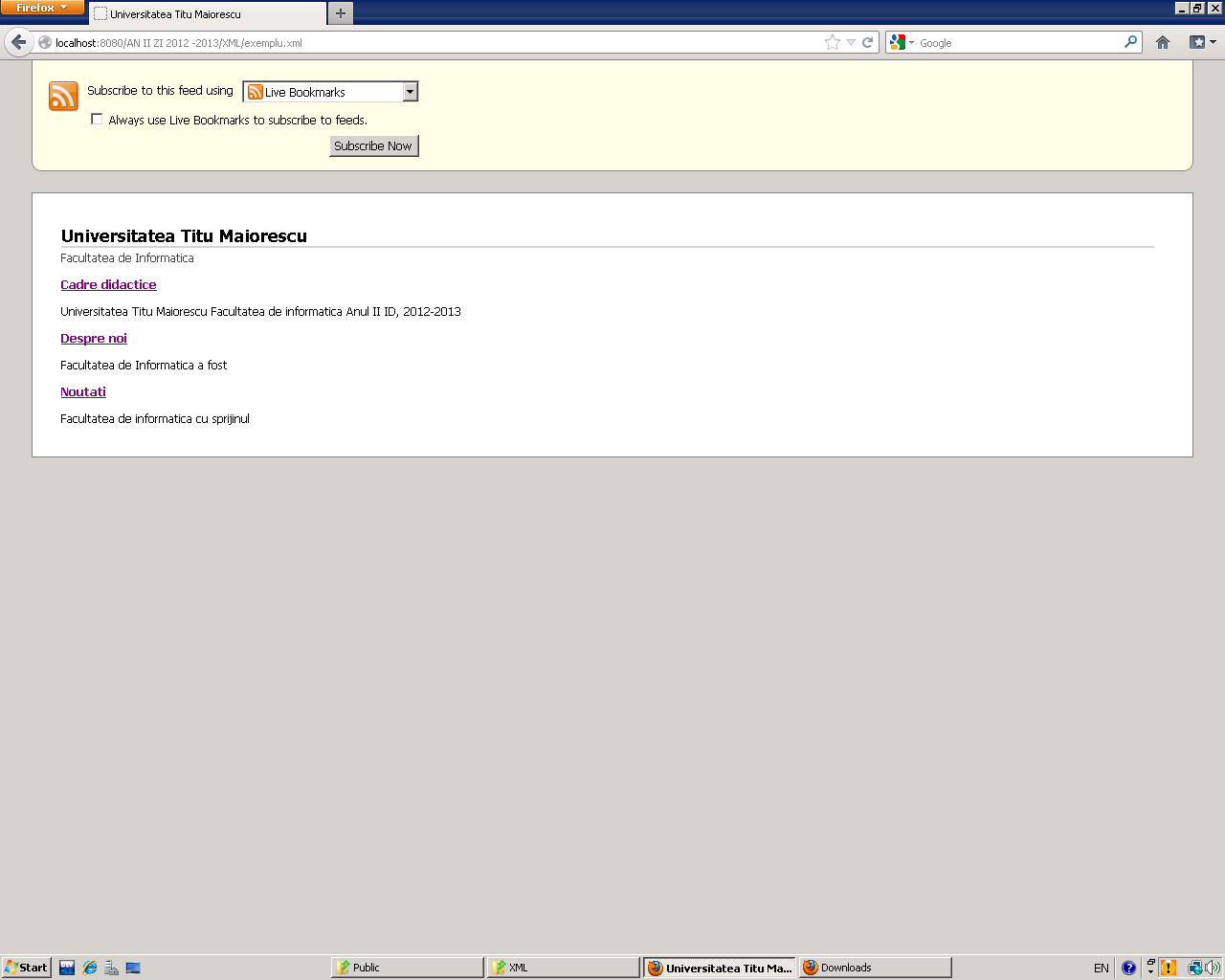


Sa se afiseze intreg continutul fisierului ex1.xml

### Modificare document XML cu SimpleXML



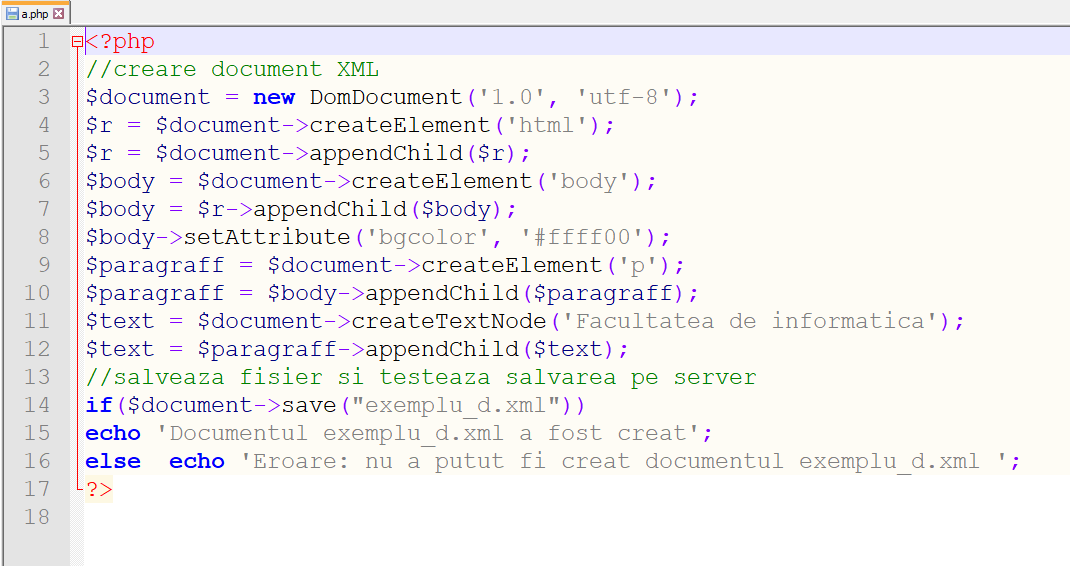
# Exemplu cu RSS



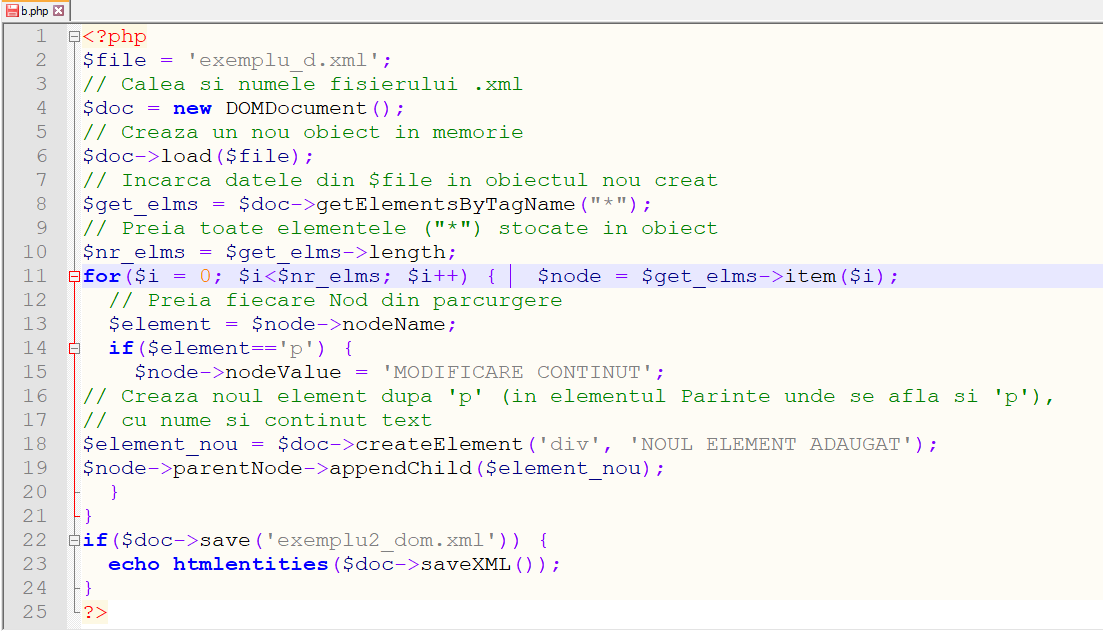
# Functii SimpleXML

* **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



Citire fisier XML, si crearea unui nou fisier modificat



Tema:

* Sa se testeze toate aplicatiile din curs.
* Adaugare fisier intr-un tabel dintr-o baza de date.

