UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE AMPOSDRU
Fondul Social European POSDRU 2007-2013
Instrumente Structurale 2007-2013
MINISTERUL EDUCAŢIEI CERCETĂRII TINERETULUI ŞI SPORTULUI OIPOSDRU
MINISTERUL EDUCAŢIEI CERCETĂRII TINERETULUI ŞI SPORTULUI UMPFE

Proiect cofinanţat din Fondul Social European prin Programul Operaţional Sectorial Dezvoltarea Resurselor Umane 2007-2013
Investeşte în oameni!

Formarea profesională a cadrelor didactice
din învăţământul preuniversitar
pentru noi oportunităţi de dezvoltare în carieră

REALIZAREA PAGINILOR WEB

Traian C. IONESCU **Ana Magdalena ANGHEL**

Adriana OLTEANU **Radu Nicolae PIETRARU**

Program de conversie profesională la nivel postuniversitar
pentru cadrele didactice din învăţământul preuniversitar

Specializarea TEHNOLOGIA INFORMAŢIEI ŞI COMUNICĂRII
Forma de învăţământ ID - semestrul II

2011

TEHNOLOGIA INFORMAŢIEI ŞI COMUNICĂRII

Realizarea paginilor WEB

Traian C. IONESCU
Ana Magdalena ANGHEL
Radu Nicolae PIETRARU
Adriana OLTEANU

2011

Acest manual a fost elaborat în cadrul "Proiectului pentru Învăţământul Rural", proiect co-finanţat de către Banca Mondială, Guvernul României şi comunităţile locale şi este revizuit în cadrul proiectului "Formarea profesională a cadrelor didactice din învăţământul preuniversitar pentru noi oportunităţi de dezvoltare în carieră", proiect co-finanţat din Fondul Social European.

ISBN 973-0-04103-2

REALIZAREA PAGINILOR WEB
CUPRINS

REALIZAREA PAGINILOR WEB

INTRODUCERE

Stimate cursant,

Încă de la început doresc să îţi urez bun venit la studiul modulului, destinat deprinderii utilizării unor unelte pentru realizarea paginilor WEB. Acest modul se adresează în principal personalului didactic din aria învăţământului obligatoriu, ce activează în mediul rural în vederea obţinerii calificării necesare prin intermediul Programului de Educaţie la Distanţă. Sper că acest modul va fi util personalului didactic care dispune de cunoştinţe despre universul calculatoarelor cât şi oricărei persoane interesate în dobândirea de cunoştinţe specifice de introducere in acest domeniu.

Există totuşi anumite cunoştinţe specifice necesare parcurgerii acestui modul:

- Cunoaşterea modalităţii de organizare şi manipulare a informaţiei în format electronic (sistem de fişiere, directoare, copierea, mutarea şi deschiderea de fişiere electronice).
- Operaţii de bază utilizând sistemul de operare Microsoft Windows (pornirea unei sesiuni de lucru, deschiderea unei aplicaţii, comutarea între mai multe aplicaţii ce rulează simultan, închiderea în mod corespunzător a sistemului),

Obiectivele modulului:

După studiul acestui modul veţi fi suficient de pregătit pentru a fi capabil să:

- Descrieţi arhitectura World Wide Web
- Descrieţi structura de bază a unui document HTML
- Explicaţi sintaxa directivelor HTML
- Descrieţi modul de specificare a culorilor pentru elementele HTML
- Definiţi titluri şi paragrafe în cadrul unui document HTML
- Modificaţi tipurile de caractere, mărimea sau culoarea acestora într-un document HTML
- Creaţi liste ordonate, neordonate şi imbricate cu ajutorul directivelor HTML specializate
- Creaţi hyperlink-uri în documente HTML
- Folosiţi imagini ca hyperlink-uri
- Adaugaţi imagini la o pagină HTML
- Specificaţi modul de aliniere al imaginii în pagină şi al textului din jurul ei ; dimensiunea imaginii
- Creaţi un tabel HTML si să modificaţi proprietăţile acestuia
- Creaţi şi să adaugaţi elemente la un formular
- Ce este un script pe partea de client
- Folosiţi un scrip într-o pagină WEB
- Folosiţi un applet Java într-o pagină WEB
- Folosiţi un obiect Active X într- pagină WEB

- Folosiţi fişierele multimedia într-o pagină Web
- Folosiţi limbajul PHP pentru a realiza pagini Web dinamice
- Folosiţi limbajul PHP pentru a utiliza informaţii dintr-o bază de date
- Folosiţi un document XML într-o pagină HTML
- Folosiţi sintaxa pentru definirea unei foi de stil

Prezentul modul este în format tipărit fiind conceput pentru educaţia prin corespondenţă. Modulul este de lungime medie: 14 ore pentru SI (studiu individual), 7 ore pentru AT (activităţi tutoriale), 7 ore pentru TC (teme de casă) şi 28 de ore pentru AA (activităţi asistate). Activităţile tutoriale au ca scop stabilirea unui dialog între cursant şi tutore în vederea discutării rezultatelor obţinute în urma evaluării temelor de casă şi nu în ultimul rând pentru lămurirea eventualelor neclarităţi sau probleme întâlnite de cursant. Temele de casă constau în rezolvarea lucrărilor de verificare care vor fi trimise tutorelui. Cele 28 ore alocate activităţilor asistate vor fi destinate efectuării celor şapte lucrări de laborator prezente în Anexa 1. Timpul de învăţare poate varia în funcţie de cunoştinţele anterioare ale cursantului despre realizarea paginilor WEB şi de cantitatea de muncă dedicată subiectului în studiu, pe care cursantul este dispus să o aloce.

Manualul de faţă este organizat în 14 unităţi de învăţare, fiecare dintre aceste unităţi conţinând o parte de prezentare teoretică a subiectului tratat, o parte de exerciţii şi rezolvările acestora. Cele 14 unităţi de învăţare îsi propun să te înveţe ce este şi cum poti sa construieşti o pagină WEB. Prima unitate face o introducere în Internet, World Wide Web, şi prezintă paşii şi uneltele necesare pentru realizarea primei tale pagini de WEB. Începând cu unitatea de învăţare 2 până la unitatea de învăţare 8 manualul te va ghida pas cu pas în tainele limbajului HTML. În unitatea de învăţare 9 şi 10 sunt prezentate tehnologii folosite în World Wide Web pentru a realiza o pagină capabilă să interacţioneze cu utilizatorul. Unităţile de învăţare 11 şi 12 vă vor familiariza cu limbajul PHP şi cu utilizarea bazelor de date utilizând acest limbaj. La finalul manualului în unităţile de învăţare 13 şi 14 vom studia tehnici avansate pentru structurarea într-un mod cât mai eficient şi mai flexibil a informaţiilor ce trebuiesc prezentate în pagina Web.

Instrucţiuni de transmitere a lucrărilor de verificare:

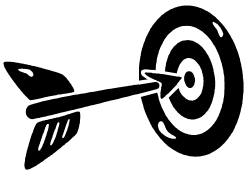
Modulul **Realizarea paginilor WEB** conţine patru lucrări de verificare (LV). Fiecare LV va fi transmisă spre corectare tutorelui, la care aţi fost alocat, într-un fişier separat, astfel incât să intre în posesia acesteia înainte sau cel târziu la data specificată de calendarul modului. Prima LV trebuie predată după ce încheiaţi studiul Unităţii de învăţare nr. 3, a doua după încheierea studiului Unitătii de învăţare nr. 6; a treia după încheierea studiului Unitătii de învăţare nr. 9; iar ultima, la sfârşitul modului.

Instructiuni de întocmire a lucrărilor de verificare:
LV-urile sunt o componentă importantă a modulului, care îngăduie o corectă evaluare a experienţei pe care o căpătaţi studiind modulul şi a capacităţii dvs. de a-i aplica ideile în practică. Rezolvarea problemelor propuse din lucrările de verificare este asemănatoare cu cea din exemplele din cadrul unităţii de învăţare respective. Ceea ce trebuie să faceţi este să prezentaţi cât mai concis ideile cele mai importante, încercând să nu depăşiţi limita de **3000 de cuvinte.** Suma maximă a punctelor care vi se acordă pentru tratarea subiectelor unei lucrări de verificare este 25.

Criteriile de evaluare si ponderile evaluării continuie şi finală:

Notele pe care le veţi obţine la sfărşitul acestui modul se calculează în funcţie de nota pe care o veţi primi la examen şi de media obţinută la LV-uri (evaluare pe parcurs). Cele două componente participă la nota finală cu ponderi de 60% (pentru LV), respectiv 40% (pentru evaluarea finală realizată prin examen). In notarea evaluării pe parcurs, notele celor patru LV-uri intră cu ponderi egale. De aceea, vă recomand insistent să predaţi **toate** cele patru LV-uri, deoarece pentru o lucrare pe care nu o realizaţi veţi primi nota 0.

Testele de autoevaluare (TA) reprezintă o formă de autoevaluare a cursantului şi face parte din tehnologia ID de parcurgere a materialului de studiu. Testele de autoevaluare sunt incluse în manual pentru a te ajuta să îţi testezi cunoştinţele şi felul în care ai înţeles materialul deja parcurs dintr-o unitate de învăţare. TA sunt concepute astfel încât să nu îţi consume mai mult de căteva minute. Răspunsurile la testele de autoevaluare se vor completa în spaţiile libere din chenar, acestea încadrându-se strict în spaţiul rezervat.



La începutul fiecărei Unităţii de învăţare vor fi detaliate obiectivele propuse, această secţiune fiind indicată de imaginea alăturată (o săgeată). Modulul nu integrează alte materiale suplimentare de studiu individual, dar recomanda la finalul fiecărei Unităţii de învăţare un decupaj minimal din bibliografia manualului, decupaj necesar pentru aprofundarea şi inţelegerea completă a noţiunilor expuse pe durata Unităţii de învăţare.

Pe tot parcursul prezentărilor teoretice, importanţa anumitor paragrafe va fi semnalizată in partea stangă a textului prin imaginea unei goarne.

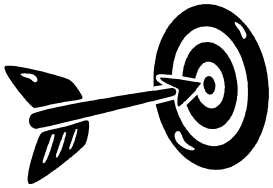
Paragrafele care conţin testele de autoevaluare vor fi semnalizate prin folosirea imaginii unei pene şi vor fi incadrate într-un chenar.

În cazul în care nu veţi reuşi să rezolvaţi problemele propuse trebuie recitite zonele de text care apar înainte de lucrarea de verificare. În speranţa că nu vor exista probleme vă urăm:
Spor la treabă!

Unitatea de învăţare Nr. 1

INTERNET ŞI WORLD WIDE WEB

Obiectivele Unităţii de învăţare Nr. 1:



Principalele obiective ale Unităţii de învăţare 1 sunt:

- Însuşirea unor noţiuni de bază din domeniul reţelelor de calculatoare
- Însuşirea unor noţiuni de bază din domeniul Internet
- Descrierea arhitecturii World Wide Web
- Însuşirea paşilor necesari într-un ciclu de creare-vizualizare rezultat, pentru o pagină Web

1.1 Introducere în Internet

Înainte de a putea înţelege ce înseamnă şi cum funcţionează World Wide Web, va trebui să clarificăm anumite noţiuni, definiţii, tehnologii pe care se bazează.

Ce este o reţea de calculatoare?

O **Reţea de calculatoare** este o colecţie de calculatoare (zeci sau sute) interconectate între ele prin cabluri speciale cu scopul de a putea interschimba sau folosi în comun anumite resurse (fişiere, imprimante, etc).

Pentru a comunica între ele calculatoarele folosesc un set de reguli care definesc noţiunea de **protocol de comunicaţie.**

Ce este Internetul?

Reţelele de calculatoare locale pot fi la rândul lor interconectate , formând reţele globale de calculatoare, adică inter-reţele. Cea mai mare inter-reţea cu access public este reţeaua **Internet.**

Definiţia de mai sus este foarte generală şi mai necesită câteva completări:

- Internetul este un mijloc de comunicare – este un mediu foarte eficient de expunere a ideilor unei audienţe foarte mari.
- Internetul este o resursă de informare – este un imens depozit de informaţii. Oricând ai nevoie de o informaţie legată de orice domeniu vei putea găsi undeva publicat pe Internet o lucrare care să te ajute.
- Internetul este o comunitate – face posibilă şi foarte eficientă comunicarea între oameni cu aceleaşi preocupăr.

Care sunt serviciile Internet?

Pentru toate facilităţile enumerate mai sus Internetul oferă mai multe metode de access – **servicii Internet** :

- World Wide Web – serviciul ce perminte accesul la informaţia stocată pe un calculator aflat oriunde în lume,

- E-Mail – este un serviciu de mesagerie electronică. Permite schimbul de mesaje între utilizatorii Internet,
- FTP – este prescurtarea de la File Transfer Protocol care înseamnă în limba română protocol pentru transferul fişierelor,
- Telnet – serviciul ce permite accesul la resursele altui calculator din Internet.

TCP/IP protocolul folosit în Internet

Protocolul de comunicaţie folosit în Internet pentru oricare din serviciile de mai sus este **TCP/IP (**şi pentru altele). Acesta defineşte:

- modul în care calculatoarele ar trebui să fie conectate în Internet,
- modul în care se stabileşte o legătură de la un calculator la altul din Internet,
- modul în care sunt transmise date între calculatoarele din Internet.

Ce este o adresă IP?

Pentru a se putea conecta şi a fi identificat în reţea un calculator trebuie să aibă o adresă de reţea unică În Internet această adresă se numeşte **adresa IP.** O adresă de IP este formată din 4 numere cuprinse între 0 şi 255 separate prin caracterul ".".

66.249.85.99 este un exemplu de adresă IP

Deoareace acest format de adrese este greu de reţinut de către om există posbilitatea asocierii unei forme mai prietenoase de adresare fiecărei adrese IP. Această formă de adresare poartă denumirea de **Nume de Domeniu**.

De exemplu pentru adresa IP de mai sus este asociat următorul nume de domeniu:

www.google.com

Test de autoevaluare

1.1 Ce este un protocol de comunicaţie?

1.2 Cum este identificat un calculator în Internet?

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 11.

1.2 Arhitectura World Wide Web

Ce este World Wide Web?

World Wide Web sau pe scurt Web este un sistem de calculatoare în Internet care poate efectua schimb de fişiere într-un anumit format, denumit **HTML** care suportă existenţa de legături către alte documente sau către fişiere grafice sau audio.

Din ce este format Web-ul?

Acest schimb de fişiere se face folosind tehnologia *client-server* care presupune existenţa următoarelor entităţi:

- **Pagini Web –** Fişiere cu un anumit format ce permite organizarea asociativă a informaţiilor – HTML,
- **Web Site –** Este o mulţime organizată de pagini Web
- **Server** – Un calculator contectat la Internet pe care sunt stocate paginile Web şi pe care rulează un program - **Web server** - care poate servi aceste pagini Web la cerere unui alt calculator din Internet,
- **Client** - Un calculator conectat la Internet pe care rulează un program - **Web Browser -** ce permite comunicaţia cu HTTP server la care face cerere pentru a primii o resursă Web specificată de un **URL** introdus de utilizator pe care este apoi capabil să il interpreteze şi să îl afişeze.
- **URL -** Fiecare pagină sau resursă WEB are asociată o adresă unică în Internet cunoscută sub acronimul de **URL** (*Uniform Resource Locator*).

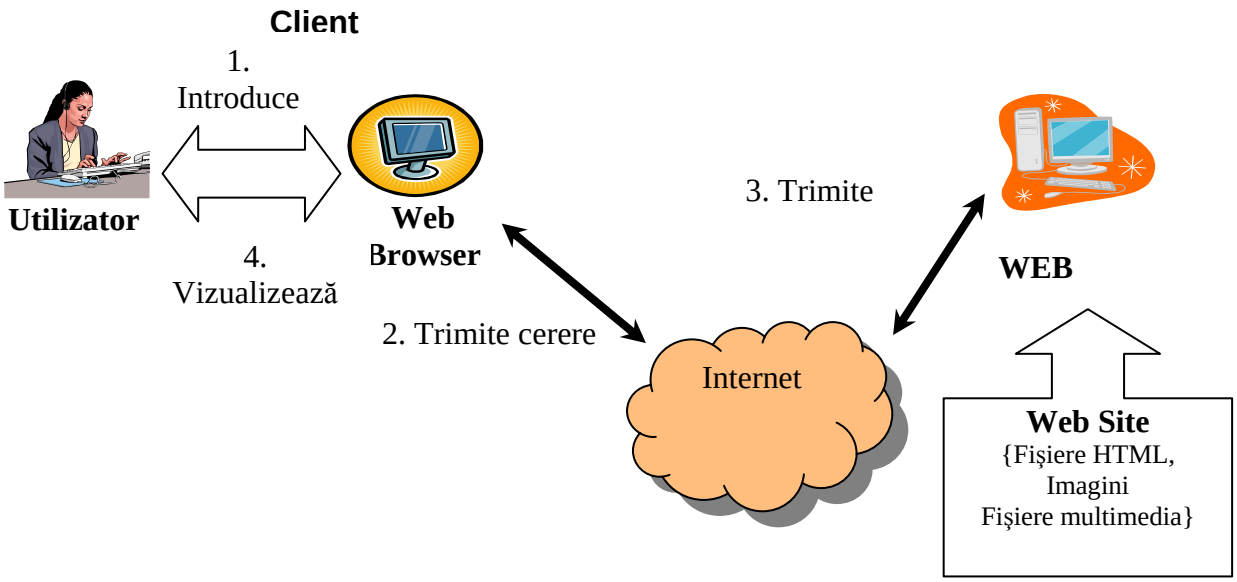


Figura 1.1 Arhitectura World Wide Web

Sensul de circulaţie al datelor în Web

În Figura 1.1 este prezentat "locul" fiecărei entităţi descrise mai sus ca fiind parte din World Wide Web, şi de asemenea ordinea etapelor pentru accesul la resursele Web:

1. Utilizatorul introduce URL-ul care identifică resursa Web dorită în browserul care rulează pe calculatorul client,

2. Clientul conectat la Internet formulează o cerere de resursă către Web-Serverul identificat de URL,
3. Web Serverul primeşte, analizează cererea şi întoarce ca răspuns la client resursa cerută dacă aceasta există. In caz contrar întoarce un răspuns ce conţine un mesaj de eroare.
4. Web Browserul de pe calculatorul client primeşte răspunsul de la Web-server şi îl afişează.

Test de autoevaluare

1.3 Un Web Browser poate rula pe acelaşi calculator cu un Web Server?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 11.

1.3 Editarea şi vizualizarea unei paginii WEB

Un document HTML implementează o pagină Web. Documentele HTML sunt simple fişiere text ceea ce permite ca editarea lor să fie posibilă prin folosirea oricărui editor de texte.

Cum editez o pagină Web?

În prezent există o serie de editoare specializate pentru editarea paginilor HTML. Acestea permit realizarea de documente HTML rapid şi uşor doar prin apăsarea a câtorva butoane fără ca utilizatorul să aibă cunoştinţe de HTML. Această facilitate este foarte bună pentru utilizatorii începători, însă este foarte importantă cunoaşterea temeinică şi înţelegerea limbajului. deoarece dezavantajul principal al acestor unelte este că generează adesea cod redundant şi uneori chiar incorect. În aceste cazuri fiind necesară intervenţia directă in codul generat pentru a corecta aceste neajunsuri.
Exemple de astfel de editoare: Microsoft FrontPage, Macromedia DreamWaver, Eclipse, Bluefish etc.

În acest modul se va folosi editorul de text standard care este livrat cu sistemul de operare Windows: **Notepad**. Acesta precum banuieşti nu are nici o facilitate specială pentru HTML ci este un simplu editor de text.

Dacă nu foloseşti sistemul de operare Windows poţi folosi în mod asemănător orice editor de text disponibil pe sistemul tău:

1. Pentru a porni programul Notepad pe un sistem Windows XP:

- Click pe butonul Start din colţul stânga jos al ecranului
- Click pe meniul All Programs -> Accessories
- Localizaţi iconiţa NotePad din acest meniu şi faceţi click pe ea

2. Odată ce ai pornit aplicaţia poţi începe să scrii documentul HTML. Încearcă să-l scrii pe cel de mai jos:

```
<HTML>
 <HEAD>
  <TITLE>Prima mea pagina</TITLE>
 </HEAD>
 <BODY>
   <H1>Prima mea pagina Web</H1>
    Curand am sa devin un
    <STRONG>expert</STRONG> HTML
   <P>
   Pagina realizata de:
   <CITE>numele tau aici</CITE>
 </BODY>
</HTML>
```

3. În momentul în care documentul este complet şi doreşti să-ti salvezi munca următoarele operaţii sunt necesare:

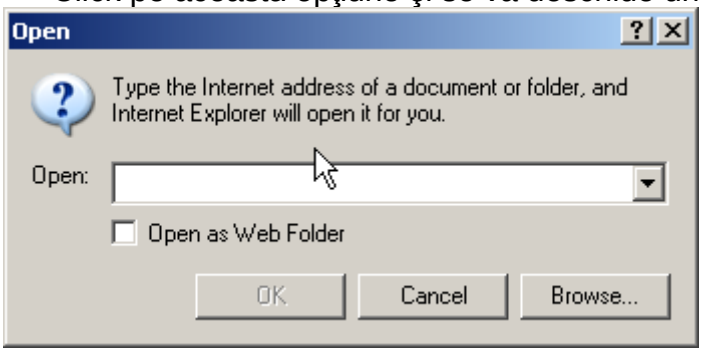
- Localizează în fereastra programului NotePad meniul “File”
- De aici selectează opţiunea “Save As…”
- În fereastra dialog care se deschide introdu numele sub care vrei să salvezi fişierul urmat de extensia .htm sau .html. Spre exemplu alege numele *primapagină.html.*

Atenţie este important să specifici extensia deoarece altfel programul NotePad va adăuga automat extensia .txt la numele fişierului.

Cum vizualizez pagina Web creată de mine?

4. Pentru a vizualiza fişierul .html proaspăt creat de tine într-un browser, următoarele operaţii sunt necesare:

- Porneşte programul Microsoft Internet Explorer
- În meniul “File” localizează opţiunea “Open…”
- Click pe această opţiune şi se va deschide următorul dialog:



- Click pe butonul “Browse…” şi navighează până în folderul în care ai salvat fişierul primapagina.html, selectează şi. apasă butonul Open

Felicitări! Ai creat prima pagină HTML. Rezultatul ar trebui să fie asemănător cu imaginea de mai jos :

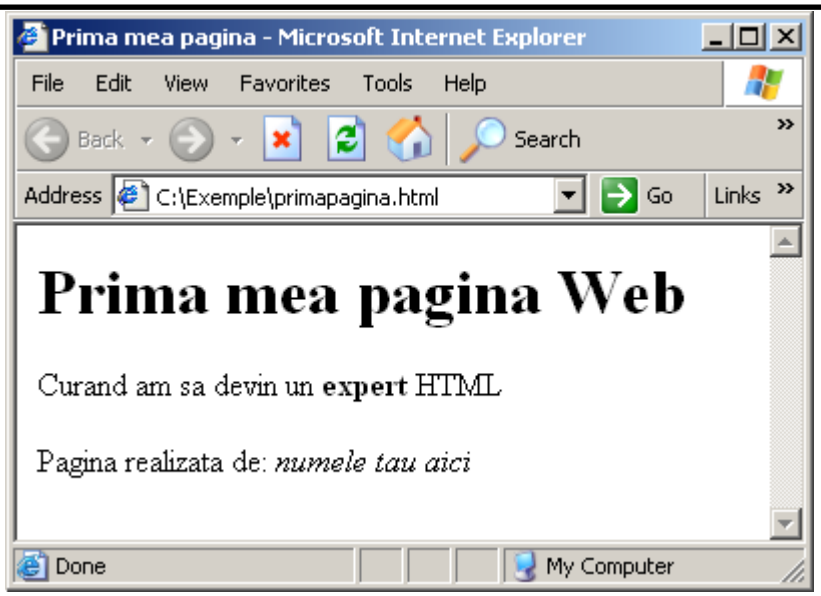


Figura 1.2 – Primul meu document HTML

Dacă rezultatul nu este asemănător cu cel din figura de mai sus înseamnă că nu ai introdus corect codul HTML sau unul dintre paşi nu au fost executat corect.
Mai încearcă odată!

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 1.1.
Un protocol de comunicaţie este un set de reguli folosite de calculatoarele într-o reţea ca să comunice între ele. A se revedea secţiunea 1.1.

Întrebarea 1.2.
Un calculator este identificat în internet cu ajutorul unei **adrese de IP** unică în Internet. Adresele de IP au formă numerică şi sunt greu de reţinut de către oameni. Din acest motiv o adresă IP poate avea asociat un nume de domeniu cu ajutorul căruia se poate identifica de asemenea un calculator în Internet. A se revedea secţiunea 1.1.

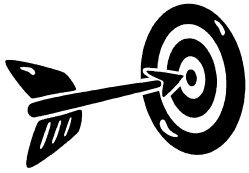
Întrebarea 1.3.
Bineînteles că da! Nu este cea mai des întâlnită situaţie însă este posibil. A se revedea secţiunea 1.2.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.21-40
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.1-7
3. Sabin Buraga – Proiectarea siturilor Web. Design şi funcţionalitate, Ediţia a II-a, Editura Polirom 2002, pg. 13-22

Unitatea de învăţare Nr. 2
LIMBAJUL HTML

Obiectivele Unităţii de învăţare Nr. 2:



Principalele obiective ale Unităţii de învăţare nr. 2 sunt:

- Definirea noţiunii HTML
- Descrierea structurii de bază a unui document
- Explicarea sintaxei directivelor HTML
- Descrierea modului de specificare a culorilor pentru elementele HTML

2.1 Introducere în HTML

HTML este limbajul folosit pentru a structura informaţia în documentele World Wide Web. Numele său reprezintă iniţialele următoarelor cuvinte in limba engleză: "***H****yper****T****ext* ***M****arkup* ***L****anguage*".

`Definitie`

Limbajul HTML este compus din instrucţiuni de afişare, care vor fi folosite de către browser pentru a determina cum anume să afişeze informaţia utilă din document. Aceste instrucţiuni de afişare poată denumirea de **directive HTML** sau, **tags** în limba engleză.

`Definitie`

Un **element** este o componentă fundamentală din strucutra unui document. O pereche de directive delimitează un element HTML, iar informaţia cuprinsă între directiva de început şi cea de sfârşit delimitează conţinutul elementului. Un element poate conţine text simplu sau alte elemente.

2.2 Sintaxa Directivelor HTML

În cazult documentelor HTML mai mult de jumătate din textul sursă nu este afişat de către browser. Mai precis ceea ce lipseşte este textul cuprins înăuntrul caracterelor pereche "<" şi ">", aceasta datorită faptului că în limbajul HTML ceea ce este cuprins între aceste caractere sunt interpretate ca fiind **directive HTML** (tags).

O directivă HTML constă dintr-un **nume** care, opţional, este urmat de o **listă de atribute** ale directivei HTML, toate acestea fiind plasate între perechea de caractere "<" şi ">". Atributele unei directive HTML pot lua diferite valori şi permit autorului documentului să-i modifice comportamentul. Iată un exemplu:

Dacă directivei <BODY> i se adaugă atributul "BGCOLOR" cu o valoare dorită atunci browserul va afişa pagina respectivă pe un fundal de culoarea specificată de valoarea atributului. Pentru a afişa pagina pe fond roşu vom scrie:

```
<BODY BGCOLOR=”red”>
```

De obicei numele directivei este intuitiv fiind reprezentat de un cuvânt sugestiv pentru funcţia pe care o are.

Directive HTML pereche

Directivele HTML apar de regulă în perechi, de exemplu <html> </html> sau <body></body>, prima directivă fiind de început iar cea de-a doua de sfarşit. Un text aflat între aceste directive va respecta funcţionalitatea directivei respective.

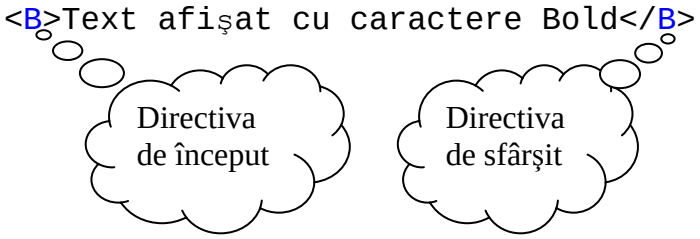


Figura 2.1 – Mod de folosire al directivelor HTML pereche.

Exemplu de mai sus prezintă modul în care se pot folosi directivele HTML pentru afişarea textului cu caractere de tip Bold.

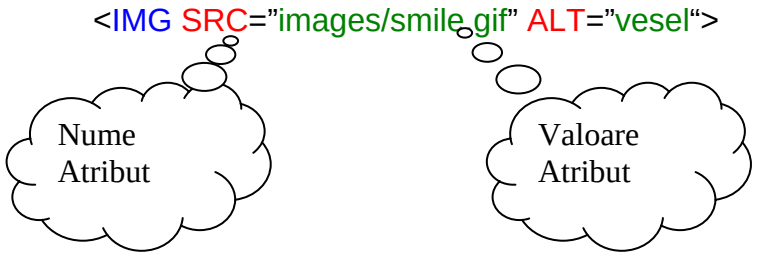
O directivă de sfârşit are acelaşi nume cu directiva corespunzătoare de început, dar este precedată de caracterul “/”. Tagurile de sfârşit nu conţin niciodată atribute.

Directive HTML fară pereche

O parte din directivele HTML nu au directive pereche de sfârşit. Acestea poartă denumirea de *stand-alone*. Un exemplu de directivă stand-alone este <img>. Aceasta face ca browserul să plaseze o imagine (specificată de valoarea atributelor directivei) în pagină.

Atributele

Atributele sunt adăugate directivelor HTML pentru a extinde sau pentru a modifica comportamentul acestora. Atributele apar întotdeauna în perechi de forma nume/valoare. Poţi adăuga mai multe atribute la acelaşi tag, separând fiecare tag printr-unul sau mai multe spaţii. Ordinea de apariţie nu este relevantă. Valorile atributelor au lungimea limitată la 1024 caractere.



Valoarea unui atribut trebuie scrisă de regulă între caracterele ‘ “ ’ şi ‘ “ ‘, iar în cazurile speciale în care însuşi valoarea atributului conţine

caracterul ghilimele se folosesc apostroafe, ca în exemplu de mai jos:

<MAP NAME='nume cu caracterul " '>

Dacă valoarea este un singur număr sau cuvânt şi conţine numai litere (a-z) sau cifre (0-9) sau caracterele "." şi "-", nu este obligatorie încadrarea între ghilimele sau apostroafe a valoarii atributului, ea putând fi plasată imediat după semnul "=".

Totuşi atunci când nu eşti sigur că regula de mai sus este respectată este o idee bună folosirea încadrării între ghilimele a valorii atributelor.

Mai jos sunt prezentate câteva exemple de directive care conţin atribute. Fii atent la faptul că prima directivă conţine atât atribute a căror valoare este încadrată de ghilimele cât şi atribute care nu necesită această încadrare:

```
<IMG SRC="img/myimg.gif" ALIGN=right WIDTH=45
HEIGHT=60>
<BODY BGCOLOR="#000000">
<FONT FACE="Arial, Helvetica" SIZE=4>
```

În afară de textele care conţin caractere obişnuite, HTML oferă posibilitatea de a insera şi afişa caractere care în mod normal nu ar putea fi incluse în document sau care au un scop predefinit în limbajul HTML – cum ar fi, spre exemplu, caracterul "<" care este folosit în limbajul HTML pentru a semnala începutul unei directive.

Test de autoevaluare

2.1 Ce este un element HTML?

2.2 Ce este o directivă HTML?

2.3 Ce rol au atributele unui element?

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 22.

Caractere speciale

În limbajul HTML, caracterul ampersand (“&”) instruieşte browserul să insereze un caracter special în funcţie de codul sau identificatorul ce îl urmează. O astfel de construcţie poartă denumirea de “*character entities*” şi este formată din trei părţi:

- Caracterul ampersand &
- Un nume predefinit al caracterului sau semnul # urmat de codul caracterului
- Caracterul “;” (punct şi virgulă)

Spre exemplu pentru a afişa caracterul “<” se va scrie în text următoarea construcţie: <. Similar > introduce caracterul “>” iar & introduce caracterul ampersand.

Numele predefinite pentru caracterele speciale trebuiesc întotdeauna scrise cu litere mici.

Se pot de asemenea introduce caractere speciale pentru care nu există identificatori predefiniţi, folosindu-se semnul “#” urmat de codul ASCII al caracterului. Spre exemplu următoarea construcţia < are acelaşi efect ca < , adică introducerea caracterlui “<”.

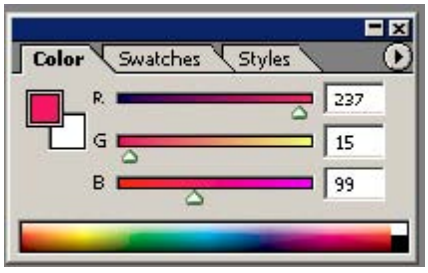
Specificarea culorilor în HTML

Culoarea anumitor elemente din cadrul documentului HTML se poate specifica folosindu-se anumite directive HTML sau atribute ale acestora. Există 2 metode de specificare a culorilor în cadrul paginilor WEB:

- prin valorea codului RGB
- folosind numele culorii

Cel mai adesea folosită şi mai flexibilă modalitate este folosirea codului RGB. Fiecare culoare RGB conţine 3 valori corespunzătoare concentraţiei fiecărei componente roşu (**R**ED), verde (**G**REEN) sau albastru (**B**LUE) din cadrul culorii alese. Aceste valori se încadrează între 0 si 255. Specificarea concentraţiei maxime (R=255, G=255, B=255) în fiecare componentă rezultă în culoarea albă, iar specificarea valorii 0 pentru toate componentele rezultă în culoarea neagră.

Majoritatea utilitarelor de grafică au dialog care permite în mod grafic alegerea culorii dorite furnizând pentru aceasta codul RGB corespunzător. În figura de mai jos este prezentat un astfel de dialog:



Se observă în imaginea de mai sus că în funcţie de culoarea aleasă se generează concentraţia corespunzătoare fiecărei componente de culori în cazul nostru (R=237, G=15, B=99).

Odată identificate aceste concentraţii pentru culoarea dorită vor trebui transformate în valori hexazecimale echivalente pentru a le putea folosi ca valoare pentru un atribut în cadrul unei directive HTML. Aceasta este sintaxa:

"#RRGGBB"

Cu ajutorul acestor valori se poate specifica orice culoare din spaţiu de culori "true color".

Cea de-a doua metodă este a identifica culorile după nume. Există un set de 140 de culori predefinite.

Comentarii HTML

Cometariile sunt folosite în cadrul unui document HTML pentru a documenta codul HTML scris in pagina respectivă. Comentariile nu sunt afişate de către browser. În procesul de creare sau de mentenanţă a paginilor de Web aceste comentarii se pot dovedii foarte utile oferind persoanei care crează sau modifică pagina informaţii suplimentare despre ce anume s-a dorit a fi realizat în acea pagină şi eventual informaţii suplimentare despre modul în care se realizează acel obiectiv.

Pentru a definii un comentariu HTML se foloseşte o directivă specială cu următoarea sintaxă:

```
<!--Textul comentariului -->
```

sau

```
<!--Textul comentariului
    pe mai multe randuri -->
```

Trebuie să existe obligatoriu un spaţiu după <!-- şi un spaţiu înainte de --> , în rest se poate folosi aproape orice caracter sau combinaţie de caractere în interiorul cometariului fără a influenţa modul în care pagina va fi afişată de către browser.

Imbricarea directivelor HTML

O directivă HTML pote fi folosită în interiorul unei alte directive HTML cu scopul de a putea aplica efectul ambelor directive asupra unui anumit element. Spre exemplu dacă se doreşte ca un anumit cuvânt din text să fie şi în format italic şi bold în acelaşi timp pentru a-l scoate în evidenţă se foloseşte această metodă de imbricare a directivelor după cum este arătat în exemplul de mai jos:

```
…
Numele meu este: <B><I>Ana</I><B>.
…
```

Va avea ca rezultat:

Numele meu este: ***Ana***

Totuşi această imbricare trebuie facută având mereu grjă ca ordinea de închidere a tagurilor să fie inversă cu ordinea de deschidere – **ultimul tag deschis să fie primul inchis.**

Informaţii ignorate de browser

Există anumite informaţii, caractere sau chiar directive pe care browserele nu le afişează. Acestea sunt:

- Sfaşitul de linie (CR/LF)
  Caracterele de sfârşit de linie folosite în interiorul documentului HTML sunt ignorate şi browserul va organiza textul in funcţie de dimensiunea ferestrei. Trecerea la linie nouă se va face explicit prin folosirea directivelor <P> sau
- Taburile sau spaţiile multiple
  Dacă browserul întâlneşte un caracter Tab sau mai multe caractere “spaţiu liber” consecutive îl va afişa ca un singur caracter spaţiu liber. Pentru a introduce spaţii suplimentare se poate folosi caracterul special: .
  Spre exemplu:

  ```
  Text,    cu    multe    spatii
  ```

  Va fi afişat ca:
  Text, cu multe spatii

- Directivele <p> care apar de mai multe ori.
  Dacă directiva <P> este folosită de mai multe ori consecutiv fără conţinut, browserul va interpreta această construcţie ca un singur paragraf şi îl va afişa ca atare.
- Comentariile
  Browserele nu vor afişa textul cuprins între caracterele <!-- şi -->
- Directivele necunoscute
  Browserele ignoră directivele pe care nu le suportă sau pe cele incorect specificate. În funcţie de tipul browserului comportamentul în cazul acestor directive este fie de a nu afişa nimic când sunt întâlnite, fie de a afişa conţinutul directivei ca simplu text.

Test de autoevaluare

2.4 Ce sunt caracterele speciale şi cum pot fi acestea folosite in cadrul unui document HTML?

2.5 Identificaţi care din codurile de mai jos reprezintă codul RGB care specifică culoarea roşie :

a> ☐ (255, 0, 0) sau #FF0000
b> ☐ (0, 0, 0) sa #000000
c> ☐ (0, 255, 0) sau #00FF00
d> ☐ (255, 255, 0) sau #FFFF00

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 22.

2.3 Structura de bază a unui document HTML

Documentele realizate cu ajutorul limbajului HTML au extensia htm sau html şi sunt simple fişiere de tip ASCII (text). Un document HTML conţine text util (conţinutul efectiv al paginii) şi directive HTML care sunt folosite pentru a defini structura, modul de afişare şi comportamentul conţinutului.

Structura unui document HTML

Fiecare document HTML începe cu directiva <HTML> care specifică browserului faptul că informaţia care urmează în fişier este în format HTML. Ultima directivă este </HTML> şi marchează sfarşitul documentului HTML.

După această primă directivă urmează un bloc marcat de directivele <HEAD> şi </HEAD> care reprezintă antetul documentului, informaţiile din cadrul acestui bloc nefiind afişate de către browser.

Următoarea secţiune, marcată de directivele <BODY> şi </body>, conţine informaţia afişată de către browser. Această secţiune include informaţia utilă, şi directive HTML care specifică browserului modul în care acesta o va afişa. Directivele HTML pot de asemenea să facă referire la diverse fişiere externe cum ar fi fişiere de imagine sau alte documente HTML.

Structura de bază unui document HTML este următoarea:

```
<HTML>
 <HEAD>
  Informaţii generale despre documentul HTML
 </HEAD>
 <BODY>
  Corpul principal al paginii
 </BODY>
</HTML>
```

Directiva HEAD

Antetul unui document HTML conţine informaţii generale despre conţinutul şi strucura documentului. Directiva <HEAD> nu conţine atribute ci serveşte drept cadru pentru alte directive: <BASE>, <ISINDEX>, <LINK>, <META>, <NEXTID>, <OBJECT>, <SCRIPT>, <STYLE> şi <TITLE>. Le vom descrie pe scurt pe cele mai des folosite.

Directiva <TITLE> - reprezită titlul documentului. Textul definit în interiorul ei va apărea ca titlu pentru fereastra de browser care afişează pagina şi de asemenea va fi folosit atunci când pagina este adaugată la în meniul “Favorites” sau “Bookmarks”. De asemenea acest text va fi folosit de către motoarele de căutare atunci când îşi adaugă pagina în baza de date. Pentru toate aceste motive este important ca acest text să fie cât mai sugestiv.

Directiva <BASE> - stabileşte calea de bază pe care serverul de WEB o va folosi pentru toate legăturile definite în interiorul documentului. Despre aceasta şi despre legături vom vorbi în unitatea de învăţare 5

Directiva <SCRIPT> - conţine cod Java Script sau VB Script despre care vom vorbi în unitate de învăţare 9 (

Directiva <STYLE> - conţine informaţii despre stilurile folosite de către tabela de stiluri (CSS) despre care vom vorbi în unitate de învăţare 12.

Directiva BODY

Corpul documentului HTML este definit de către directiva pereche <BODY></BODY>. Conţinutul lui poate fi un singur paragraf, o imagine sau o combinaţie complexă de imagini, tabele, obiecte multimedia, text.

Elementul BODY are un rol foarte important în ceea ce priveşte imaginea de ansamblu a pagini HTML, deoarece permite definirea unor parametrii globali cum ar fi: culoarea sau imaginea de fundal a paginii sau culoarea textului şi a legaturilor din pagină. Aceşti parametrii globali se definesc cu ajutorul atributelor directivei <BODY>.

Atributul BGCOLOR permite stabilirea culorii de fundal a paginii. În exemplul de mai jos se defineşte culoarea albastru ca fundal:

```
<BODY BGCOLOR=“blue”></BODY>
```

Atributul TEXT este folosit pentru a definii culoarea textului normal din cadrul documentului. Culoare implicită a textului este negru. În exemplul de mai jos este setată culoarea de fundal alb şi culoarea textului roşu.

```
<BODY BGCOLOR=“#FFFFFF” TEXT=”#FF0000”>
```

La alegerea culorii textului trebuie avut în vedere ca acesta să poată fi distins uşor de fundal.

Atributele LINK, ALINK, VLINK permit controlul culorii legăturilor în funcţie de starea acestora după cum este descris mai jos:

- LINK – stabileşte culoarea cu care vor fi afişate iniţial legăturile şi implicit este Albastru
- VLINK – stabileşte culoarea cu care vor fi afişate legaturile care au mai fost vizitate implicit este Purpuriu
- ALINK – stabileşte culoarea cu care este afişată legătura deja vizitată.

În exemplul de mai jos se va stabili următoarea schemă de culori pentr pagină: culoarea de fundal alb, culoarea textului va fi roşu, culoarea legăturilor albastru, culoarea legăturilor vizitate va fi magenta, iar a legăturilor active va fi verde

```
<BODY     BGCOLOR=“#FFFFFF” TEXT=”#FF0000”
          LINK=”#0000FF” VLINK=”#FF00FF”
          ALINK=”#00FF00”>
</BODY>
```

Atributul BACKGROUND – cu ajutorul acestui atribut al elementului BODY se poate seta ca fundal pentru documentul HTML o imagine. Imaginea specificată ca valoare a atributului trebuie să fie în format .gif sau .jpg, şi va fi poziţionată astfel încât să acopere întreaga arie a paginii. Dacă o singură imagine nu este destul de mare pentru a satisface această condiţie atunci aceasta va fi replicată şi spaţiul rămas neocupat va fi umplut la dreapta şi în jos cu aceste replici până când fereastra browserului este complet ocupată de imagine.

```
<BODY     BGCOLOR=“#FFFFFF”
          BACKGROUND=”logo.jpg”>
```

În exemplul de mai sus s-a specificat imaginea care va fi folosită ca fundal şi de asemenea culoarea de fundal utilizată de browser până în momentul încărcării imaginii.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 2.1.
Un element este o componentă fundamentală din structura unui document. O pereche de directive delimitează un element HTML, iar informaţia cuprinsă între directiva de început şi cea de sfârşit delimitează conţinutul elementului. Revedeţi secţiunea 2.1.

Întrebarea 2.2.
Directivele HTML sunt instrucţiuni folosite de către browser pentru a determina cum anume să afişeze informaţia utilă din document. Revedeţi secţiunea 2.1.

Întrebarea 2.3.
Atributele sunt adăugate directivelor HTML pentru a extinde sau pentru a modifica comportamentul acestora. Revedeţi secţiunea 2.2.

Întrebarea 2.4.
În afară de textul normal HTML oferă posibilitate afişării unor caractere care în mod normal nu sunt aifşate de către browser deoarece acestea sunt caractere cheie folosite pentru identificarea unor construcţii HTML. Spre exemplu: caracterele “<“ sau “>”. Acestea se pot specifica în cadrul unui document HTML cu ajutorul caracterelor speciale definite printr-o construcţie de tipul: caracterul “&” + cod caracter + caracter “;”. Revedeţi secţiunea 2.2.

Întrebarea 2.5.
Codul RGB este : (255, 0, 0). Varianta corectă de răspuns : a). Revedeţi secţiunea 2.2.

Indicaţii la problemele propuse

Problemele propuse în lucrările de verificare se fac dupa modelele de exemple prezentate în unitatea de învăţare sau dupa tipicul acestora.

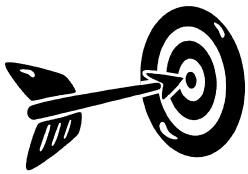
Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.53-70
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.17-20, pg.28-29, pg.48-49

Unitatea de învăţare Nr. 3
FORMATAREA TEXTELOR IN LIMBAJUL HTML

Obiectivele Unităţii de învăţare Nr. 3:

Principalele obiective ale Unităţii de învăţare Nr. 3 sunt:



După parcurgerea acestei Unităţii de învăţare veţi fi capabili:

- Să definiţi titluri şi paragrafe în cadrul unui document HTML
- Să modificaţi tipurile de caractere, mărimea sau culoarea acestora într-un document HTML
- Să folosiţi directivele HTML pentru formatarea logică a textului
- Să folosiţi directivele HTML pentru formatarea fizică a textului

3.1 Limbajul HTML si formatarea textelor

Pentru ca informaţiile dintr-un document să fie asimilate cât mai uşor de către cititorii săi şi pentru ca prezentarea lor să aibă succes maxim este foarte important ca textul să fie organizat într-o formă cât mai atractivă. HTML oferă mijloace eficiente pentru a structura şi a înfrumuseţa un text.

Ca autor al unui document HTML ai două opţiuni în ceea ce priveşte textul pe care doreşti să îl afişezi. Prima este să îl scrii aşa cum este şi a doua să în incluzi între anumite directive HTML. Spre exemplu dacă vrei să afişezi textul “La multi ani!” poţi pur şi simplu să îl tastezi în cadrul documentului şi va fi afişat fără nici o problemă. Însă dacă vrei ca acest text să fie afişat într-un anumit mod – spre exemplu vrei ca textul să fie scris cu caractere roşii de dimensiunea mai mare decât cea normală a textului - va trebui să specifici explicit browserului modul în care vrei ca textul tău să fie afişat cu ajutorul directivelor HTML pentru formatarea a textelor.

3.2 Titluri şi paragrafe HTML

Directiva <H*n*></H>

În cadrul oricărui document este necesară definirea unui titlu principal şi a mai multor titluri pentru fiecare din subsecţiunile documentului. Limbajul HTML oferă o modalitate uşoară de definire a titlurilor de diferite dimensiuni. Titlurile se definesc folosindu-se directiva <H*n*>, unde *n* este un număr cuprins între 1 si 6. Valoarea 1 indică realizarea unui titlu cu cea mai mare dimensiune în timp ce 6 va creea un titlu cu cea mai mică dimensiune.

Directivele pentru titlu au un atribut optional care poate schimba modul de aliniere. Acest atribut este “align” şi poate luat una din valorile: “left” - stânga, “right” – dreapta sau “center” – centru. Folosirea valori “left” pentru atributul align este însă redundantă deoarece alinierea implicită folosită pentru cazurile în care nu este specificată este “left”.

Pentru a vedea modul în care este poziţionat textul din interiorul directivelor de titlu HTML şi pentru a compara diferitele dimensiuni ale acestora vom considera codul HTML de mai jos:

```
<HTML>
 <HEAD>
  <TITLE>
    Exemplu titluri
  </TITLE>
 </HEAD>
 <BODY>
   <H1 ALIGN="CENTER">
      Titlul H1 pozitionat central
   </H1>
   <H2>
      Titlul de tip H2.
   </H2>
   <P>
     Acesta nu este un titlu ci este text normal
   </P>
   <H3>
     Titlul de tip H3.
   </H3>
   <H4 ALIGN="RIGHT">
      Titlul H4 aliniat la dreapta
   </H4>
   <H5>
       Titlul de tip H5.
   </H5>
   <H6>
       Titlul de tip H6.
   </H6>
 </BODY>
</HTML>
```

Rezultatul va fi în browser de forma:

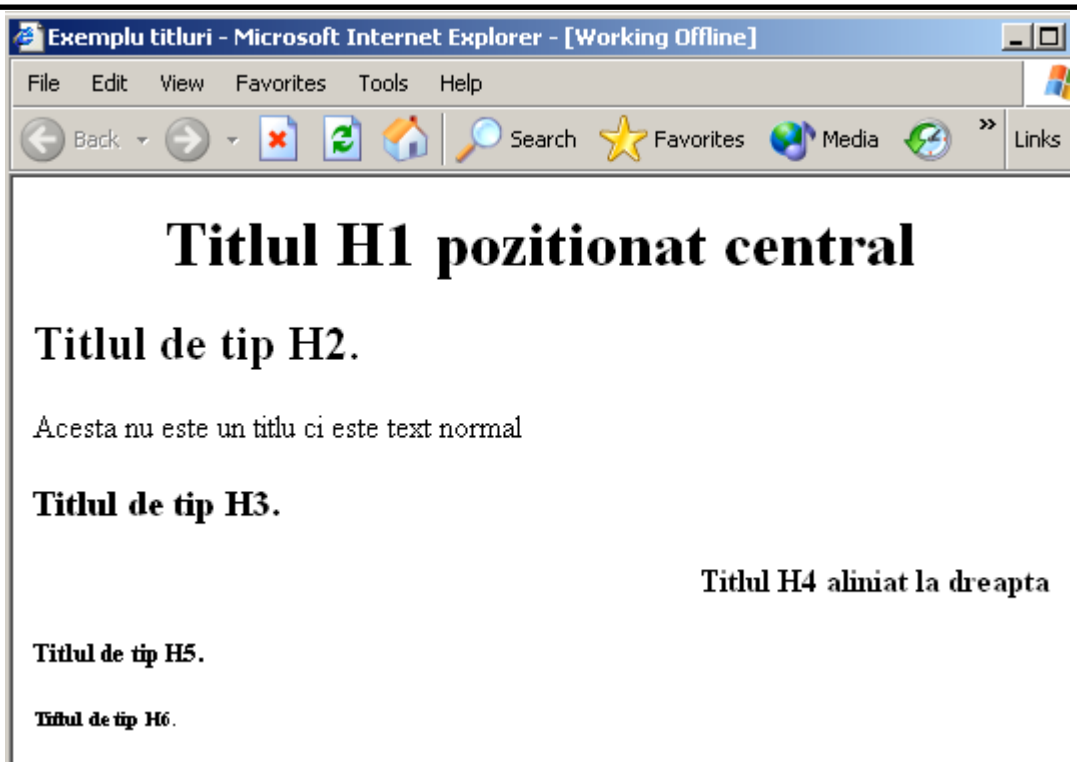


Figura 3. 1 Exemple titluri HTML

- Nu uita să foloseşti directiva de sfârşit pentru titlu deoarece browserul nu va include implicit una.
- Un document HTML bun ar trebui să folosească titlurile în ordinea ierarhică. H1 ar trebui folosit pentru primul nivel de titlu, H2 pentru al doilea s.a.m.d
- Se va evita omiterea unui nivel ierarhic la un moment dat, adică H3 nu ar trebui să apară imediat după H1, ci se va folosi H2 în loc.

Directiva <P></P>

O metoda eficientă de a face o pagină cât mai uşor de citit este despărţirea acesteia în paragrafe. Spre deosebire însă de documentele scrise cu majoritatea procesoarelor de text existente, caracterele de linie nouă nu sunt luate în considerare. În fapt, orice tip de spaţiere –linie nouă, taburi sau spaţii – vor fi transformate într-un singur spaţiu liber în momentul afişării documentului de către browser.

Pentru a indica începerea sau terminarea unui paragraf şi deci trecerea la o linie nouă se foloseşte directiva HTML <P></P>.

Directiva de încheiere </P> poate fi omisă. Aceasta deoarece majoritatea browserelor la întâlnirea unei noi directive <P> consideră automat că paragraful anterior s-a terminat.

Elementele de tip paragraf permit adăugarea de text la un document iar dimensiunea liniei va fi ajustată în funcţie de dimensiunea ferestrei browserului. Această ajustare se face automat de către browser în momentul vizualizării paginii.

- Inserarea mai multor elemente <P> fără conţinut succesiv va avea ca rezultat în browser afişarea unei singure linii libere în locul lor.
- Pentru a introduce mai multe rânduri libere consecutiv se poate folosi directiva
 descrisă mai jos.
- Deoareace textul este reformatat de câte ori utilizatorul îşi dimensionează fereastra se va evita introducerea manuală a rândurilor libere.

**Directiva
**

Adeseori este necesară trecerea forţată la o linie nouă fără a termina însă paragraful curent. Deoarece caracterele de linie nouă sunt ignorate în HTML această trecere se face folosind directiva HTML
. Elementele
 nu au directivă de sfârşit obligatorie deoarece elementul marchează o poziţie şi nu are nici un conţinut care să trebuiască delimitat.

În exemplul de mai jos vom urmări modul de folosire a directivelor HTML <P> şi
 precum şi diferenţele de poziţionare în cazul folosirii fiecăreia:

```
<HTML>
 <HEAD>
  <TITLE>
    Exemplu Paragraf si Break line
   </TITLE>
 </HEAD>
 <BODY>
   <H1>Titlul</H1>
   <P> Textul paragrafului 1 </P>
   <P> Textul paragrafului 2
   <BR>
      Linie noua in cadrul paragrafului 2
   <BR> A doua linie noua in cadrul  paragrafului 2
   </P>
   <P> Textul paragrafului 3 </P>
 </BODY>
</HTML>
```

Va avea ca rezultat în browser:

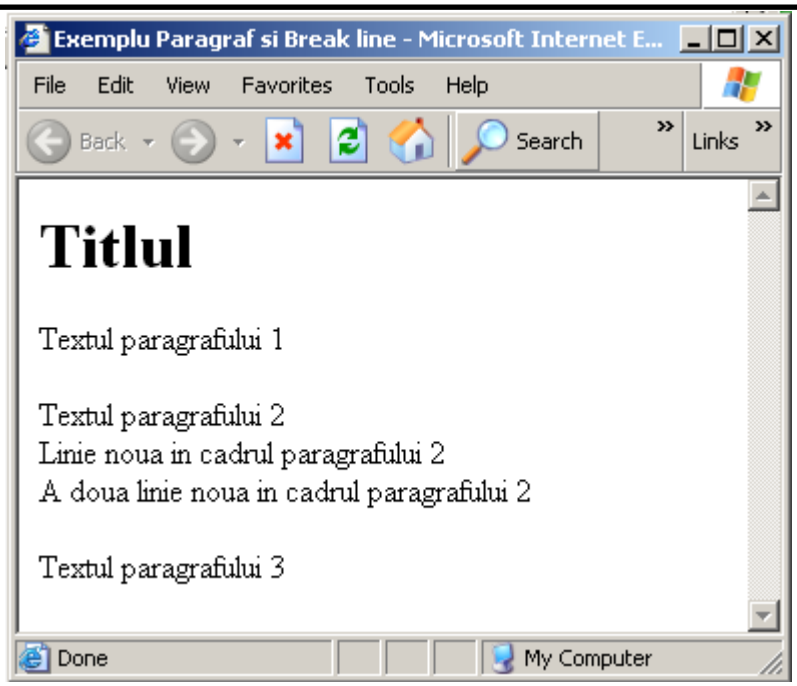


**Figura 3.2. Folosirea directivelor <P> şi
**

Se observă faptul că în cazul directivei <P> se inserează automat o linie liberă înainte şi după terminarea paragrafului, iar în cazul folosirii directivei
 nu este introdus nici un spaţiu suplimentar.

Directiva <HR>

Un alt mijloc de a separa diferitele secţiuni ale unui document este folosirea directivei <HR>. La întâlnirea acestei directive browserul trece automat la linie nouă şi desenează o linie orizontală. Caracteristicile linie orizontale pot fi controlate de către atributele directivei.

Atributul **WIDTH.** Specifică lungimea liniei fie ca dimensiune absolută în pixeli, fie ca procent din lungimea ferestrei browserului.
Atributul **ALIGN.** Poate lua una din valorile *left*, *center* sau *right* şi specifică modul de aliniere a liniei. Acest atribut nu are semnificaţie fără existenţa atributului WIDTH.
Atributul **NOSHADE.** Implicit linia este desenată în relief având umbră. Prezenţa acestui atribut indică faptul că linia va fi desenată fără umbră.

Atributul **SIZE**. Specifică grosimea – numărul de pixeli - cu care va fi desenată linia.

În exemplu de mai jos vom specifica o desenare a unei linii aliniate la dreapta desenată fără umbră şi cu dimensiunea 20% din fereastra browserului şi grosimea de 10 pixeli:

```
<HR
     NOSHADE
     ALIGH=”RIGHT”
     SIZE=”10”
     WIDTH=”20%”>
```

Test de autoevaluare

3.1. Directiva
 se foloseşte :

a> ☐ pentru a forţa trecerea la o linie nouă
b> ☐ fortează începerea unui paragraf nou
c> ☐ pentru introducerea mai multor linii libere succesive
d> ☐ introducerea unei linii orizontale

3.2. Cum formatează textul elementele de tip paragraf?

3.3. Cum se desenează în HTML o linie orizontală centrată cu dimensiunea jumătate din lungimea ferestrei browserului

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 37.

3.3 Definirea caracteristicilor fontului

Directiva <BASEFONT>

Directiva **<**BASEFONT**>** defineşte dimensiunea implicită a fontului pentru întreg textul folosit în interiorul elementului <BODY>. Titlurile H1-H6 nu sunt afectate de această directivă. Directiva BASEFONT are un singur atribut: **SIZE** care defineşte dimensiunea implicită a fontului folosit în cadrul documentului. Valoarea acestui atribut este un intreg între 1 şi 7 Dacă nu este specificată acest atribut are valoarea 3.
Pentru a schimba dimensiunea implicită a fontului la 5 scriem:

```
<BASEFONT SIZE=”5”>
```

- Schimbarea fontului de bază ar trebui să fie făcută cu grijă şi având un motiv clar deoarece utilizatorul are posibilitatea oricum să definească dimensiunea preferată a textului din opţiunile browserului.
- Aceast element nu are directivă de încheiere
- Directiva BASEFONT afectează textul din cadrul elementului BODY dar nu şi titlurile. Alegerea unei dimensiuni prea mari pentru fontul de bază poate duce la situaţia în care fontul

pentru titlu are dimensiuni mai mici decât fontul textului propriuzis producând astfel confuzie.

- Directiva BASEFONT afectează întreg textul ce urmează după ea şi până la încheierea elementului BODY.
- Nu este recomandat folosirea ei pentru schimbarea dimensiunii fontului unui bloc de text a caracterelor individuale, pentru aceasta fiind disponibile alte directive precum FONT, BIG sau altele despre care vom discuta mai taziu în această unitate de învăţare

Directiva <FONT>

Pentru a stabili caracteristici ale fontului precum: dimensiune, culoare, tip pentru un bloc de text se foloseşte directiva **<FONT>**. Schimbarea caracteristicilor fontului pentru o parte din caractere sau pentru anumite cuvinte este o metodă bună pentru a scoate în evidenţă ceva important sau pentru a creea efecte interesante. Directiva de sfârşit este obligatorie.

Atributul **SIZE.** determină noua dimensiune a fontului pentru textul elementului. Valoarea atributului este un intreg între 1 şi 7 specificând astfel dimensiunea absolută a fontului. Prin prin adăugarea prefixului “+” sau “-“ este specificată dimensiunea relativă faţă de fontul de bază.

Pentru specificarea dimensiunii în valoarea absolută vom scrie:

```
<FONT SIZE=”7”>A </FONT>fost odata ca niciodata…
```

care va avea care rezultat în browser:

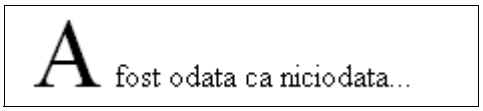


Figura 3. 3. Exemplu specificare dimensiune font în valoare absolută

- În exemplu de mai jos vom incerca să creem un efect de persepctivă pentru exclamaţia “URAAA” folosind atributul SIZE pentru directiva font şi specificând dimensiunea în format relativ:

```
<P>
   U
   <FONT SIZE="+1">R
    <FONT SIZE="+2">A
     <FONT SIZE="+3">A
      <FONT SIZE="+4">A
      </FONT>
     </FONT>
    </FONT>
   </FONT>
```

Va avea ca rezultat în browser:

Figura 3. 3. Exemplu specificare dimensiune font în valoare relativă

În cazul folosirii intercalate a directivelor <FONT> cu valoarea atributului SIZE specificată în valoare relativă, efectul nu este cumulativ ci referinţa este dimensiunea fontului de bază.

Atributul **COLOR.** Defineşte culoarea textului din interiorul elementului FONT. Specificarea culorii se face folosindu-se codul RGB[1] sau numele predefinit al culorii dorite.

Pentru a afişa un text cu culoarea galbenă şi dimensiunea absolută 5 vom scrie:

```
…
<P>
 Ultimul cuvant are culoarea
 <FONT COLOR="#0000FF" SIZE="5">
   Albastra
 <FONT>.
</P>
…
```

Şi rezultatul în browser va fi:

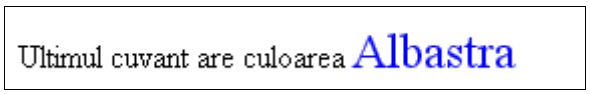


Figura 3. 4. Exemplu specificare culoare font

Atributul **FACE.** Permite schimbarea tipului de font folosit pentru afişarea textului. Dacă fontul nu este suportat de către browser va fi folosit tipul implicit de font. Se pot specifica mai multe tipuri de font despărţie prin virgulă şi un spaţiu.

<FONT FACE=”arial, courier, garamond”>

În acest caz daca primul tip de font nu este suportat de către browser se va încerca cu cel de al doilea s.a.m.d.

În exemplu prezentăm o listă de fonturi uzuale suportate de către majoritatea browserelor:

[1] Vezi unitatatea de învăţare numărul 2

Figura 3. 5. Tipuri de font uzuale

- Foloseşte directiva FONT pentru a schimba caracteristicile fontului pentru un număr oarecare de cuvinte sau paragrafe, iar pentru schimbarea intregii pagini fosolseşte BASEFONT.
- Este indicat să eviţi folosirea schimbării dimensiunii fontului folosind valori extreme deoarece aceasta poate face documentul greu de citit.

Test de autoevaluare

3.4 În ce cazuri este recomandată a se folosi directiva <BASEFONT> în locul directivei <FONT>

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 37.

3.4 Formatarea fizică a textelor

Directivele pentru formatare fizică forţează browserul să afişeze textul elementului respectiv într-un anumit format dorit de creatorul documentului HTML respectiv neţinând cont de setările din browser ale utilizatorului respectiv.

Dacă una din aceste directive nu este suportată de către browser ea va fi ignorata deoarece browserul nu are nici o altă alternativă de afişare. În cele ce urmează este prezentată o listă cu aceste directive şi este pe scurt descrisă funcţionalitatea fiecăreia.

Directivă	Funcţionaliate
<B>	Afişează textul îngroşat
<BIG>	Afişează textul cu o unitate mai mare decât dimensiunea fontului de bază
<SMALL>	Afişează textul cu o unitate mai mic decât dimensiunea fontului de bază
<I>	Afişează textul înclinat.
<U>	Afişează textul subliniat
<SUB>	Afişează textul sub formă de indice inferior
<SUP>	Afişează textul sub formă de indice superior
<TT>	Afişează textul cu caractere teleprinter
<BLINK>	Afişează textul pâlpâind

Toate directivele de formatare fizică de mai sus necesită specificarea explicită a directivei pereche de sfârşit.

3.5 Formatarea logică a textelor

Formatarea logică presupune, spre deosebire de cea fizică, faptul că fiecare browser va formata textul afectat de directivele logice în funcţie de posibilităţile platformei pe care rulează. Prin urmare formatarea propriuzisă nu va fi neapărat la fel de la un browser la altul, însă efectul va fi acelaşi. Spre exemplu pentru unele browsere textul din cadrul directivei <EM> va apărea îngroşat în timp ce pentru altele va apărea italic.Creatorul paginii se va concentra în acest caz mai mult pe definirea seminificaţiei textului decât asupra modului în care acesta va fi formatat în browser.

Să discutăm despre fiecare element în parte:

<ACRONYM> indică faptul că textul inclus este un acronim, adică un cuvânt format din iniţialele unor cuvinte care fac parte dintr-o expresie sau un nume.

<CITE> indică faptul că testul inclus este un citat bibliografic. Prin convenţie acest text este afişat înclinat.

<CODE> este folosit pentru a afişa exemple de cod sursă. Textul acestui element este afişat cu font de tip teleprinter precum este fontul “courier”.

<EM> este folosit pentru a afişa un text ce trebuie scos în evidenţă, sau un termen nou introdus in document. Majoritatea browserelor vor afişa textul acestui element în format îngroşat sau înclinat.

<STRONG> efectul este asemănător cu cazul folosirii directivei <EM> dar mai puternic.

Toate directivele de formatare logică de mai sus necesită specificarea explicită a directivei pereche de sfârşit.

3.6 Preformatarea textului cu ajutorul directivei <PRE>

Aşa cum am mai arătat în unitatea de învăţare precedentă în momentul în care un document HTML este afişat de către browser, spaţiile aflate între două cuvinte adiacente, în caz ca sunt mai multe, vor fi transformate automat într-un singur spaţiu. Există, bineînţeles metode de a definii spaţii suplimentare între cuvinte dacă acest lucru este necesar, iar una din aceste metode este folosirea directivei HTML: <PRE>.

Directiva <PRE> împreună cu directiva sa pereche de sfârşit, creează un spaţiu în interiorul căruia textul va fi afişat de către browser exact în formatul din codul sursă HTML, păstrându-se numărul de spaţii libere sau numărul de linii libere succesive. Lungimea liniei nu mai este ajustată în acest caz în funcţie de dimensiunea ferestrei browserului.

Textul din interiorul elementului <PRE> este afişat folosindu-se un font cu lungimea constantă a caracterului (e.g. courier).

Conţinutul elementului <PRE> poate include orice directivă de formatare fizică sau logică, imagini sau legături[2]. Directivele care implică terminarea unui paragraf – adică <P> sau <Hn> - nu sunt recomandate spre a fi folosite în interiorul elementului deoarece rezultatul folosirii acestora în acest caz nu este consistentă pentru toate tipurile de browser.

Directiva <PRE> are un atribut opţional - **WIDTH** care determină numărul de caractere conţinute într-o linie a blocului preformatat. Dacă o linie are lungimea mai mare de cea specificată de acest atribut nu înseamnă ca aceasta va fi automat ajustată la această dimensiune ci mai degrabă va fi extinsă în afara regiunii vizibile a ferestrei browserului.

Folosirea acestor elemente este în general utilă pentru realizarea de tabele sau atunci când se doreşte păstrarea integrităţii unor coloane sau rânduri. Pentru aceasta HTML pune la dispoziţie însă şi alte directive speciale despre care vom vorbi în lecţiile următoare. Însă avantajul folosirii acestei tehnici în locul directivelor specializate este că nu toate browserele suportă utilizarea acestora.

[2] Vom discuta despre acestea in unitatea de invaţare 5 şi 6

Lucrare de verificare a Unităţilor de învăţare Nr. 1, 2 şi 3

1. Creaţi un document HTML care să conţină doar structura de bază a unui document HTML şi care să aibă

- culoare de fundal: galbenă
- Să afişeze textul: “Am făcut prima Lucrare!” cu culoarea roşie

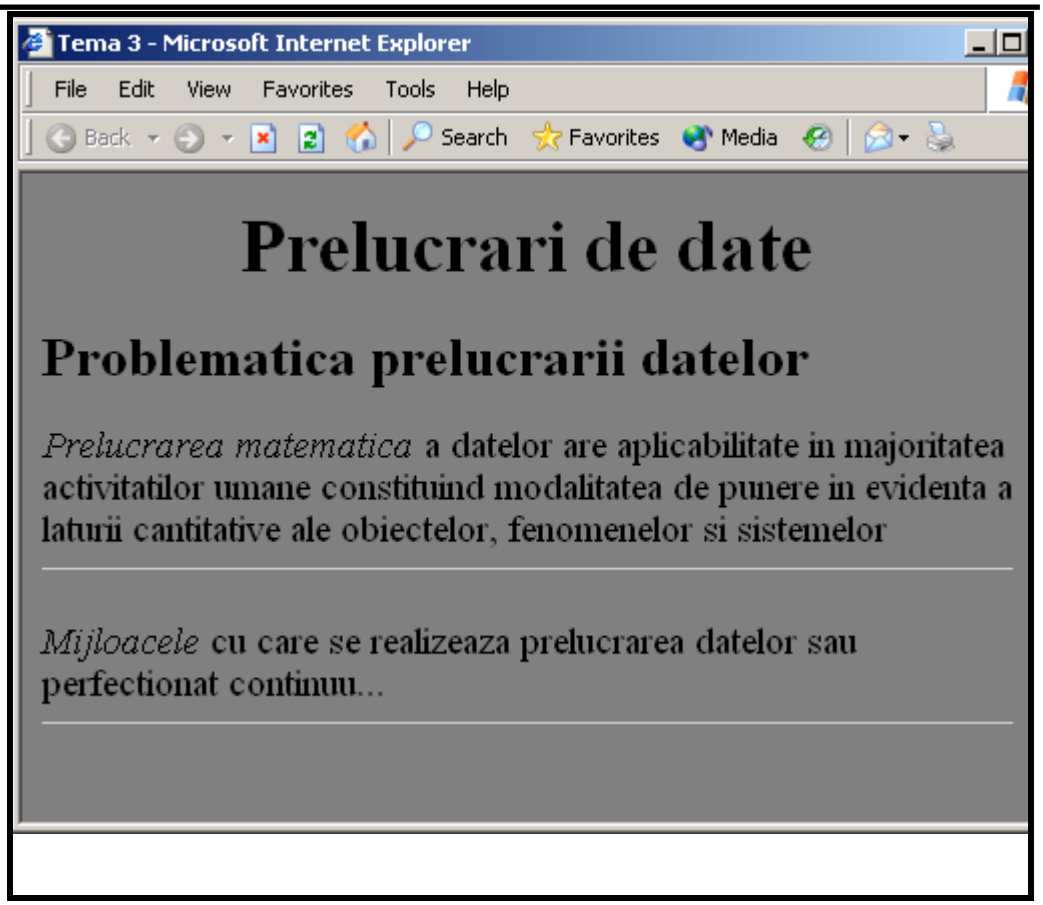
Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare.
Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 2 din bibliografia unităţii de învăţare.

Nr. de puncte **15** (7 puncte definire culoare fundal, 8 puncte afişarea textului cerut folosid culoarea roşie)

2. Scrieti documentul HTML pentru o pagină WEB ca cea din figura de mai jos. Aceasta trebuie sa conţină: 1 titlu de tip H1 poziţionat central, 1 titlu de tip H2 aliniat la stânga, corpul textului va fi format din 2 paragrafe în care primul cuvânt din text are format italic.

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare.
Ca ajutor suplimentar utilizati reperele bibliografice 2 şi 3 din bibliografia unităţii de învăţare.

Nr de puncte **10** (5p - folosirea directivelor specifice pentru titluri şi paragrafe, 2p - împărţirea în 2 paragrafe a textului (la fel ca in figură), 3p - formatarea tip italic a primului cuvânt din fiecare paragraf.


Tema 3 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Prelucrari de date
Problematica prelucrarii datelor
Prelucrarea matematica a datelor are aplicabilitate in majoritatea activitatilor umane constituind modalitatea de punere in evidenta a laturii cantitative ale obiectelor, fenomenelor si sistemelor
Mijloacele cu care se realizeaza prelucrarea datelor sau perfectionat continuu...

Răspunsuri şi comentarii la întrebările din testele de autoevaluare

Întrebarea 3.1.
Directiva
 se foloseşte pentru a forţa trecerea la o linie nouă. De asemenea se poate folosi în cazul în care se doreşte introducerea mai multor linii libere succesive. Vezi secţiunea 3.2.

Întrebarea 3.2.
Elementul de tip paragraf va formata textul conţinut astfel încât dimesiunea liniei să nu depăşească lungimea ferestrei. La începutul şi sfârşitul unui paragraf se va adaugă automat o linie liberă. Vezi secţiunea 3.2.

Întrebarea 3.3.
Directiva HTML pentru a desena această linie se scrie în felul următor:

```
<HR
     ALIGH="CENTER"
     WIDTH="50%">
```

Vezi secţiunea 3.2.

Întrebarea 3.4.
Directiva <BASEFONT> se foloseşte în cazul în care se doreşte schimbarea caracteristicilor fontului pentru întreg textul documnetului. Directiva FONT se foloseşte pentru a schimba caracteristicile pentru un număr oareacare de cuvinte din text. Vezi secţiunea 3.4.

Indicaţii la problemele propuse

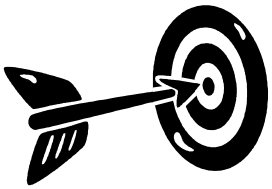
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.71-82
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.20-27
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000

Unitatea de învăţare Nr. 4
LISTE ÎN LIMBAJUL HTML

Obiectivele Unităţii de învăţare Nr. 4:



După parcurgerea acestei Unităţii de învăţare vei fi capabil:

- Să creezi liste neordonate cu ajutorul directivelor HTML specializate
- Să creezi liste ordonate cu ajutorul directivelor HTML specializate
- Să creezi liste de definiţii
- Să creezi liste imbricate.

4.1 Tipuri de liste în limbajul HTML

Clasificare Liste

Listele reprezintă un excelent mijloc pentru a sistematiza informaţia şi de a scoate în evidenţă anumite aspecte importante dintr-un anume context. Limbajul HTML oferă un suport bogat, pentru definirea listelor. Se pot creea trei tipuri de liste:

- Neordonate (*unordered lists*) – Listă fără numere de ordine.
- Ordonate (*ordered lists)* – Listă cu numere de ordine.
- Listă de definiţii (*definition lists)* – Este o listă compusă din termeni şi definiţii ale acestora.

Cu excepţia listelor de definiţii, toate tipurile de listă au aceeaşi structură de bază, fiecare listă constând dintr-o secvenţă de elemente marcate de directiva <LI>.

```
<SPECIFICATOR_TIP_LISTA>
 <LI> ELEMENT 1 text element 1 </LI>
 <LI> ELEMENT 2 text element 2 </LI>
 ...
</SPECIFICATOR_TIP_LISTA>
```

În cadrul unui element al unei liste se pot folosi următoarele construcţii HTML: paragrafe, imagini, legături, alte liste, directive de formatare text.

<LI>

Funcţionalitate:
- Defineşte un element dintr-o listă

Atribute:
- TYPE
- VALUE

Directiva de sfârşit:
- </LI> este opţională

Directiva <LI> este folosită atât în cadrul listelor ordonate cât şi neordonate după cum vom vedea în secţiunile următoare.

4.2 Liste HTML neordonate

Cum definim lista

Listele neordonate sunt folosite pentru enumerarea unor elemente pentru care ordinea de apariţie nu este importantă, cum ar fi, spre exemplu o listă de cumpărături, sau lista de obiective ale acestei secţiuni de învăţare.
Elementele unei liste neordonate vor fi afişate intendat şi fiind precedate de un marcaj. Aceast marcaj este introdus automat de către browser şi prin urmare nu trebuie specificată în codul sursă HTML.

<UL>

Funcţionalitate:
- Defineşte o listă neordonată

Atribute:
- TYPE

Directiva de sfârşit:
- </UL> este OBLIGATORIE

O listă neordonată se defineşte folosind directiva HTML <UL>, iar directiva de sfârşit corepunzătoare </UL> este obligatorie. Înauntrul acestui element fiecare element se specifică folosind directiva <LI>. Directiva de sfârşit </LI> poate fi omisă însă este recomandat ca aceasta să fie totuşi specificată deoarece este utilă în cazul folosirii stilurilor CSS despre care vom vorbii in unitatea de învăţare 12.

```
<P> Acesta este un exemplu de lista neordonata:
 <UL>
  <LI>Primul element al listei</LI>
  <LI>Al doilea element al listei</LI>
  <LI>Al treilea element al listei</LI>
 </UL>
</P>
```

Va avea ca rezultat în browser:

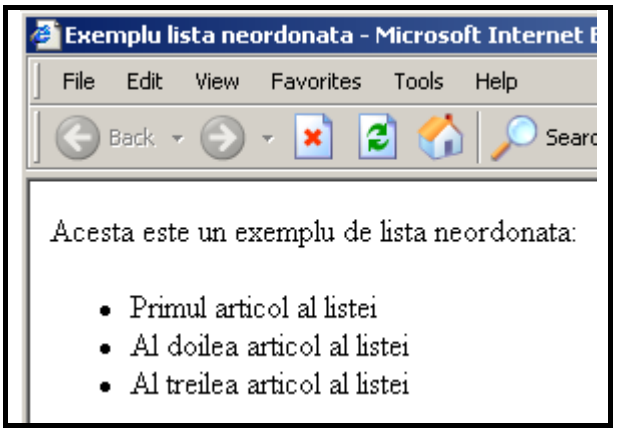


Figura 4.1 Examplu lista neordonata

O listă neordonată introduce o linie liberă între ea şi textul de deasupra şi încă una între ea şi textul ce urmează ei.

Atributul **TYPE**. Forma marcajului pentru fiecare element se poate schimba folosind atributul TYPE, care permite specificarea a 3 forme pentru marcaj:

- Valoarea “DISC” pentru forma de cerc plin – valoarea implicită
- Valoare “CIRCLE” pentru forma de cerc gol
- Valoarea “SQARE” pentru forma de pătrat gol

Atributul type poate fi de asemenea aplicat şi fiecărei directive <LI> în parte, în cazul în care se doreşte o formă diferită pentru fiecare din elementele listei.

Test de autoevaluare:

4.1 De câte tipuri pot fi listele HTML

4.2 Definiţi o listă neordonată care să conţină 3 elemente. Primul şi ultimul element să aibă marcajul de tip pătrat, iar al doilea element să aibă marcajul de tip disc.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 47.

4.3 Liste HTML ordonate

Cum definim lista ordonata?

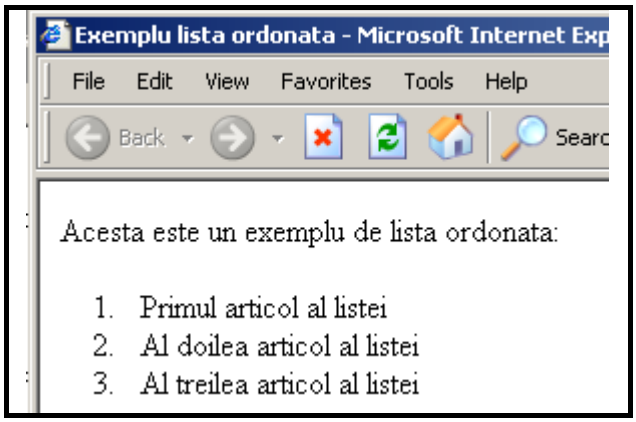
Listele ordonate sunt folosite în cazul în care ordinea elementelor este important să fie respectată, cum ar fi spre exemplu cuprinsul unei lucrări sau o listă de instrucţiuni. La fel ca în cazul listelor neordonate elementele sunt afişate intendat dar în loc de un marcaj grafic browserul va afişa automat în faţa fiecărui element un număr de ordine.

<OL>
Funcţionalitate: • Defineşte o listă ordonată **Atribute:** • START • TYPE **Directiva de sfârşit:** • </OL> - OBLIGATORIE

O listă ordonată are aceeaşi structură de bază ca o listă neordonată. Pentru a defini o listă ordonată se foloseşte directiva <OL> înăuntrul careia fiecare element se specifică folosind directiva <LI>.

```
<P> Acesta este un exemplu de lista ordonata:
 <OL>
  <LI>Primul element al listei</LI>
  <LI>Al doilea element al listei</LI>
  <LI>Al treilea element al listei</LI>
 </OL>
</P>
```

Va avea ca rezultat în browser:



Exemplu 4.2 – Exemplu listă ordonată

Atributul **TYPE.** Implicit numerotarea elementelor dintr-o listă ordonată se face automat folosindu-se cifre de tip arabice. Pentru a modifica stilul de numerotare se poate folosi atributul TYPE în cadrul directivei <OL>. Există 5 tipuri de numerotare suportate. În funcţie de stilul de numerotare dorit se va specifica valoara atributului TYPE aşa cum este specificat în tabelul de mai jos.

Valoare atribut TYPE	Stilul generat	Exemplu
A	Litere majuscule	A, B, C, D...
a	Litere mici	a, b, c, d...
I	Cifre romane majuscule	I, II, III, IV...
i	Cifre romane mici	i, ii, iii, iv...
1	Cifre arabe	1, 2, 3, 4

Spre exemplu pentru a defini o listă care foloseşte stilul de numerotare cu litere majuscule vom scrie astfel:

```
<OL TYPE=”A”>
 < LI> Primul element < /LI>
 < LI> Elementul 2 din lista< /LI>
 < LI> ultimul element din lista< /LI>
</OL>
```

Atributul **START.** Permite specificarea valorii cu care va începe numerotarea elementelor listei. Daca nu este folosit atributul start numerotarea va începe cu 1 sau respectiv cu primul caracter din setul specificat de către atributul TYPE. Spre exemplu pentru a specifica o listă ordonată care foloseşte stilul de numerotare cu cifre romane şi pentru care primul element începe de la valoarea “VI” (6) vom scrie astfel:

```
<OL TYPE=”I” START=”6”>
 <LI>
   Primul element este numerotat cu VI
 </LI>
 <LI>
   Elementul următor va fi numeroat cu VII
 </LI>
 <LI>
   Si asa mai departe.....
 </LI>
</OL>
```

Listele ordonate precum şi cele neordonate pot avea diferite nivele, în sensul că fiecare din elementele unei liste poate consta dintr-o altă sublistă. Fiecare din aceste subnivele va fi intendată corespunzător de către browser, efectul acestor intendări fiind cumulativ. De acest fapt trebuie ţinut cont atunci când se folosesc mai multe nivele de imbricare pentru liste.

Test de autoevaluare:

4.3 Când se folosesc listele ordonate în locul celor neordonate? Daţi câteva exemple de folosire.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 47.

4.4 Liste de definiţie

Cum definim lista de definiţie?

Listele de definiţii permit enumerarea unei liste de elemente fiecare din acestea fiind urmate de explicaţia lor. Spre exemplu Glosarul unei carţi poate fi considerat o listă de definiţii. Pentru definirea acestor tipuri de listă se folosesc trei directive HTML: <DD> , <DL> şi <DT>.

<DL>

Funcţionalitate:

- Creează o listă de definiţii

Atribute:

- COMPACT

Directiva de sfârşit:

- </DL> - OBLIGATORIE

O listă de definiţii este încadrată de directiva pereche <DL>. În interiorul acestor directive fiecare element al unei liste de definiţii este compus din 2 părţi. Prima parte este un termen urmată în partea a doua de definiţia acestuia. Pentru definirea primei parţi se foloseşte directiva HTML <DT>, urmată de definiţa acestuia care se face cu ajutorul directivei <DD>.

În mod normal browserul plasează definiţia termenului pe un rând nou, însă dacă această definiţie este foarte scurtă (3 caractere) o va plasa pe acelaşi rând cu termenul.

Atributul **COMPACT.** Specifică browserului să afişeze lista ocupând cât mai puţin spaţiu posibil.

<DT>	<DD>
Funcţionalitate: • Creează un termen într-o listă de definiţii **Atribute:** • Nu are **Directiva de sfârşit:** • </DT> - opţională	**Funcţionalitate:** • Creează o definiţie pentru un termen **Atribute:** • Nu are **Directiva de sfârşit:** </DD> - opţională

În interiorul elementului <DD> se poate folosi aproape orice directivă HTML, inclusiv alte liste, imagini, directive pentru formatarea textului etc. Totuşi dacă nu este folosit nici o directivă de formatare, textul din cadrul elementului <DD> este afişat intendat.

Mai jos este exemplificat modul de folosire a unei liste de definiţii prin realizarea unui glosar de termeni din domeniul WEB:

```
...
<H3> Glosar termeni WEB </H3>
  <DL>
    <DT>Browser</DT>
    <DD>Aplicatie software utilizat pentru
        vizualizarea paginilor WEB</DD>
    <DT>GIF</DT>
    <DD>Format de imagine comprimat utilizat
        frecvent in Internet</DD>
    <DT>HTML</DT>
    <DD>Limbaj pentru realizarea paginilor
        WEB</DD>
  </DL>
...
```

Va avea ca rezultat în browser:

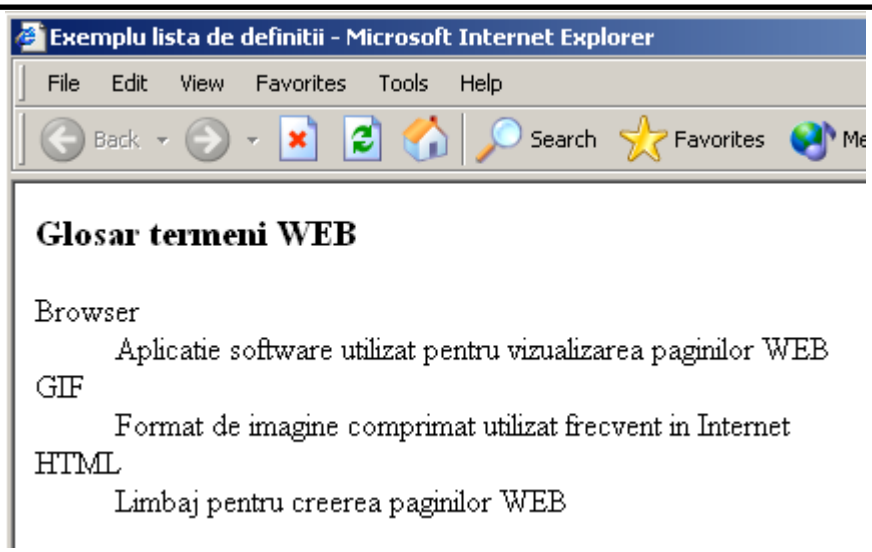


Figura 4.3 Exemplu lista definiţii

Test de autoevaluare:

4.4 . Construiţi o listă de definiţii care să conţină tipurile de liste disponibile în HTML şi definiţia lor.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 47.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 4.1.
Limbajul HTML oferă suport pentru realizarea a trei tipuri de liste: neordonate, ordonate, liste de definiţii. Consultaţi secţiunea 4.1.

Întrebarea 4.2.
Codul HTML pentru realizarea acestei liste este:

```
<UL>
 <LI TYPE=”SQARE”>Element 1</LI>
 <LI>Element 2 </LI>
 <LI TYPE=”SQARE”><LI>Element 3</LI>
</UL>
```

Consultaţi secţiunea 4.2.

Întrebarea 4.3.
Listele ordonate sunt folosite în cazul în care ordinea elementelor este relevantă. Exemple de folosire pentru listele ordonate: Cuprinsul lucrarilor, lista de paşi a unui algoritm etc. Consultaţi secţiunea 4.3.

Întrebarea 4.4.
Codul HTML pentru realizarea acestei liste este:

```
<DL>
 <DT>Neordonate</DT>
 <DD>
   Lista fara numere de ordine
 </DD>
 <DT>Ordonate</DT>
 <DD>
   Lista cu numere de ordine
 </DD>
 <DT>Lista de definitii</DT>
 <DD>
   Lista compusa din termeni si definitii ale
acestora
 </DD>
</DL>
```

Consultaţi secţiunea 4.4.

Indicaţii la problemele propuse

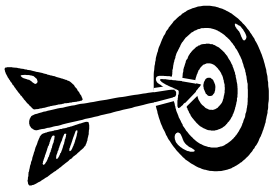
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.85-99
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.36-38
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000, pg.33-38

Unitatea de învăţare Nr. 5
HYPERLINKS ÎN LIMBAJUL HTML

Obiectivele Unităţii de învăţare Nr. 5:



După parcurgerea acestei Unităţii de învăţare veţi fi capabili:

- Să creaţi hyperlink-uri în documente HTML
- Să definiţi tipurile de legături
- Să folosiţi imagini ca hyperlink-uri
- Să întelegeţi definiţia şi structura unui URL

5.1 Hypertext şi Hyperlink

Hypertext, Hyperlinks

Termenul de *Hypertext* este format dintr-o asociere de alţi doi termeni: *Hyper* - care înseamnă “mai incolo de” si *text*. şi se referă la o modalitate de organizare a unui document diferită de organizarea liniară folosită în cazul documentelor tipărite. Mai concret cititorul unui document Hypertext are posibilitatea să-l parcurgă într-o altă ordine decât cea predefinită de către autor. Acest lucru este posibil prin existenţa legăturilor – *Hyperlinks* – între diferite componente ale documentului.

Adevărata putere a limbajului HTML constă tocmai în abilitatea sa de a lega texte sau imagini din cadrul unui document HTML de alt document HTML sau secţiune a acestuia. Aceste legături au scopul de a perminte cititorului să “sară” la un moment dat direct la locul unde va găsi mai multe informaţii despre o chestiune în discuţie. Implementând astfel conceptul de Hypertext.

`Definitie`

În esenţă un sistem Hypertext este o reţea semantică ale cărui noduri sunt fragmente de text. Dacă nodurile conţin şi imagini sau sunete, putem vorbi de un sistem Hypermedia Într-un sistem Hypertext se disting următoarele elemente fundamentale:

- ***nodurile*** *– reprezintă paginile unui document Hypertext*
- ***ancorele*** *(ţintele)– reprezită fragmente unitare de noduri cum ar fi spre exemplu cuvinte evidenţiate în text, imagini, butoane etc.*
- ***legăturile*** *între noduri, având drept punct de plecare fie întreg nodul sursă fie o ancoră a sa*

5.2 URL – Identificator unic de resurse WEB

Fiecare pagină sau resursă WEB are asociată o adresă unică în internet cunoscută sub acronimul de **URL** (*Uniform Resource Locator*).

Componentele URL-ului

Sintaxa unui URL este concepută a fi cât mai generic posibilă, un URL fiind compus din următoarele componente:

1. **Numele** resursei, precedat de:
2. I**erarhia de directoare** unde se află resursa respectivă, precedat de:
3. **Adresa de internet** sau **numele de domeniu** al serverului care găzduieşte resursa, precedat de:
4. **Protocolul** folosit de către browser şi serverul care găzduieşte resursa pentru a o transmite

Nu este obligatoriu pentru toate componentele descrise mai sus să apară într-un URL acestea fiind (în măsura posibilităţilor) completate automat de către browser sau server Aceasta este sintaxa unui URL:

protocol://adresă_server/cale_resursă/nume_resursă

Şi iată câteva exemple de URL-uri:

http://www.hotnews.ro/revista_presei.htm

http://www.desprecopii.com

ftp://ftp.myftp.ro/pub

informatii_contact.html

Primul URL din exemplul de mai sus este un URL în format *absolut* sau *complet*. Browserele oferă de asemenea posibilitatea folosirii URL-urilor *relative* sau incomplet specificate, completând automat părţile lipsă ale URL-ului specificat atât cât este posibil.
Cum ar fi în exemplul de mai sus pentru cel de-al patrulea URL browserul îl va completa automat presupunând că pagina "*informatii_contact*" se află pe acelaşi server şi în acelaşi director cu documentul curent.

5.3 Realizarea legăturilor în HTML

În cadrul unei pagini HTML unul sau mai multe cuvinte consecutive pot să aibă asociate o anumită resursă WEB adică, o altă pagină HTML, o imagine, un film etc. Directiva <A> face posibilă definirea acestei asocieri sau legăturii folosind în acest scop următoarea sintaxă:

```
<A HREF="URL">
   Textul afişat cu care se face asocierea
</A>
```

Legăturile vor fi afişate de către browser cu altă culoare şi vor fi scoase în evidenţă faţă de restul textului, de obicei fiind subliniate.

Iată un exemplu de definire a unei legături HTML:

```
...
<P>Definirea
<A HREF="http://www.invathtml.ro/despre_legaturi.html">
legaturilor HTML
</A>
este simpla!
</P>
...
```

Va avea ca rezulta în browser:

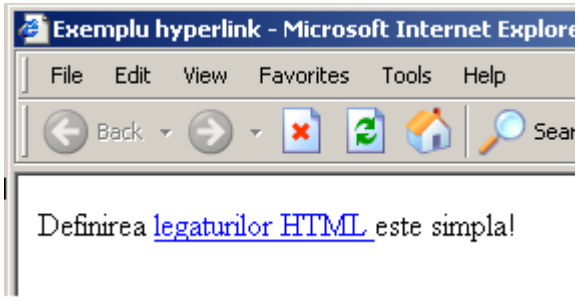


Figura 5.1 Exemplu legătură HTML

Dacă utilizatorul va selecta zona legăturii cu mouse-ul browserul va încărca pagina identificată de URL-ul specificat ca valoare a atributului HREF.

<A>
Funcţionalitate: • Creează o legătură sau o ancoră **Atribute:** • HREF • NAME • TITLE • TARGET **Directiva de sfârşit:** • </A> este OBLIGATORIE

În concluzie, directiva <A> este folosită pentru a creea o legătură (hyperlink) către un document, atributul HREF având rolul de a specifica adresa acelui document, iar cuvintele dintre directivele ancoră de deschidere <A> şi respectiv de închidere vor fi afişate ca hyperlink.

Tipuri de legaturi

Există trei tipuri principale de legături:

- **Legături interne** folosite în cadrul documentelor de dimensiuni mari pentru o mai bună structurare a conţinutului.
- **Legături locale** sunt legături către alte documente aflate pe acelaşi server. Legăturile locale se pot specifica folosind fie URL-ul complet, fie un URL relativ care conţine doar calea către resursa respectivă relativ la directorul curent.
- **Legături externe** sunt legături către pagini găzduite pe alt server de WEB. Pentru legăturile externe se foloseşte întotdeauna URL-ul specificat complet.

Atributul NAME

Aşa cum am arătat mai sus o ancoră este o etichetă folosită pentru a identifica o anumită secţiune a unui document HTML. Pentru a defini o ancoră directiva <A> este folosită împreună cu atributul său **NAME.**

Spre exemplu, presupunând că acest document este un document HTML şi vrem să definim o ancoră pe titlul acestei subunităţii de învăţare pentru a putea face referire la el din altă parte a documentului, vom scrie:

```
<A NAME=”legaturi_html”>
   Realizarea legăturilor HTML
</A>
```

Atributul HREF

Mai departe, în momentul în care se doreşte referirea acestei ancore se va folosi tot directiva <A> dar de această dată împreună cu atributul **HREF**. În cazul în care referinţa se face din aceeaşi pagină în care a fost definită ancora este de ajuns specificarea numelui ancorei precedat de caracterul “#”. Ca în exemplul de mai jos:

```
...
<P>Definirea
 <A HREF=”#legaturi_html”>
   legaturilor HTML
  </A>
este simpla.</P>
...
```

Accesarea acestei legături va spune browserului să deruleze pagina până la începutul Unităţii de învăţare “Realizarea legăturilor HTML”, unde este definită ancora “legaturi_html”.

Dacă referirea se face din altă pagină HTML dar care se află pe acelaşi server şi în acelaşi director se poate de asemenea folosi de asemenea un URL relativ. Presupunând că fişierul ce conţine

Unitatea noastra de învăţare se numeşte “despre_legaturi.html” vom scrie:

```
...
<P>Definirea
 <A HREF=”despre_legaturi.html#legaturi_html”>
   legaturilor HTML
  </A>
este simpla.</P>
...
```

În cazul în care pagina noastră se află pe alt server este necesară folosirea URL-ului complet specificat.

Test de autoevaluare

5.1 Scrieţi directiva ce realizează hyperlink la pagina şi la ancora definite mai sus în cazul în care aceasta se află pe un alt server de WEB decât pagina din care se face referire.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 57.

Atributul Target

Atributul **TARGET.** În mod obişnuit la aceesarea unui hyperlink browserul va încărca resursa specificată în aceeaşi pagină a browserului. Pentru a specifica o locaţie diferită de încărcare a resursei se foloseşte atributul target. Dacă este specifică valoarea “_blank” resursa va fi încărcată într-o fereastră pe care browserul o va crea special pentru afişarea noii resurse.

```
<A
  HREF=”http://www.yahoo.com”
  TARGET=”_blank”
>
 Yahooooo!
</A>
```

La accesarea legăturii browserul va deschide o nouă fereastră în care va afişa pagina cunoscutului portal Yahoo.

Dacă veţi folosi pentru toate legăturile din pagină atributul TARGET specificând valorea “_blank”, fiecare din aceasta va fi lansată într-o nouă fereastra de browser ceea ce nu este intotdeauna elegant

deoarece este foarte probabil ca utilizatorul să pierdă numărul ferestrelor deschise şi să nu mai fie capabil să le gestioneze eficient.

Atributul TITLE

Atributul **TITLE** Permite specificarea unui text descriptiv pentru resursa la care se referă legătura. Acest text va fi afişat de către browser în momentul în care cursorul mouse-ului este deplasat deasura legăturii.
De obicei acest text este afişat sub formă de fereastră *tool tip* precum în exemplul de mai jos.

```
<A HREF="http://www.yahoo.com"
   TITLE="Aceasta legatura ne transporta direct la
          pagina Yahoo"
>
  Yahooooo!
</A>
```

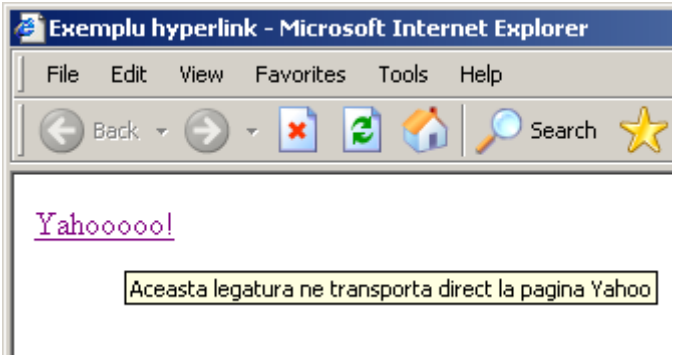


Figura 5.2 Exemplu utilizare atribut TITLE

Atributul TABINDEX

Atributul **TABINDEX.** În mod obişnuit o legătură este accesată în momentul în care utilizatorul se află cu cursorul deasupra legăturii şi apasă butonul mouse-ului. O altă modalitate de accesare a unei legături este prin apăsarea succesivă a tastei <TAB> până în momentul în care cursorul ajunge deasupra legăturii urmată de apăsarea tastei <ENTER>. Ordinea în care se face trecerea de la o legătura la alta la apăsarea tastei <TAB> poate fi stabilită de către valoarea atributului **TABINDEX.** Valoarea atributului TABINDEX poate fi orice întreg mai mare decât zero.

Atributul ACCESSKEY

Atributul **ACCESSKEY.** Valoarea acestui atribut constă într-un identificator al unei taste la apăsarea careia se va accesa legătura.

5.4 Folosirea imaginilor pentru legături

Pentru a face o pagină cât mai dinamică, mai interactivă şi mai uşor de folosit pentru utilizator o tehnică adesea utilizată este cea a folosirii imaginilor ca hyperlink-uri în locul textelor. Spre exemplu pentru legătra către pagina WEB principală se poate folosi în locul

textului HOME o imagine ce înfăţisază o casă precum imaginea de mai jos:  .

Realizarea unei legături care să folosească imagini în locul textului este simplă şi se face cu ajutorul directivelor <A> şi <IMG>, directiva IMG fiind specificată în interiorul perechii <A></A> ca în exemplu de mai jos:

```
<A HREF="Home.html">
     <IMG SRC="home.jpg">Acasa
</A>
```

Codul HTML de mai sus va crea o imagine însoţită de un text (“Acasa”) care se va comporta ca un hyperlink.

Atunci când este folosită ca hyperlink o imagine este în mod normal încadrată într-un pătrat de culoarea setată pentru hyperlink în pagina respectivă. În felul acesta este indicat faptul că hyperlinkul este activ

- Afişarea imaginilor în pagină poate fi dezactivată în anumite browsere de aceea este recomandat în cazul folosirii imaginilor drept hypelink ca directiva <IMG> să conţină şi atributul ALT cu o descriere sugestivă pentru imaginea sau hyperlink-ul respectiv.

Test de autoevaluare

5.2 Definiţi un hyperlink de tip imagine care să afiseze un text explicativ în locul imaginii în cazul în care imaginea nu poate fi inărcată în browser.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 57.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 5.1.

```
...
<P>Definirea
 <A
HREF=”http://www.adresa_server.ro/cale_fisiere_html
/despre_legaturi.html#legaturi_html”>
   legaturilor HTML
  </A>
este simpla.</P>
...
```

Pentru nelămuriri revedeţi secţiunea 5.3.

Întrebarea 5.2.
Codul HTML este următorul:

```
<A HREF="Home.html">
     <IMG SRC="home.jpg" ALT=”ACASA”>
</A>
```

Pentru nelămuriri revedeţi secţiunea 5.4.

Indicaţii la problemele propuse

Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

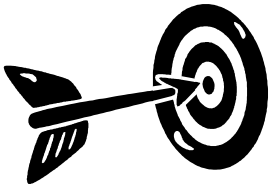
Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.100-108
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.29-32
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000, pg.65-74
4. Sabin Buraga – Proiectarea siturilor Web. Design şi funcţionalitate, Ediţia a II-a, Editura Polirom 2002, pg. 23-30

Unitatea de învăţare Nr. 6
FOLOSIREA IMAGINILOR ÎN LIMBAJUL HTML

Obiectivele Unităţii de învăţare Nr. 6:

Principalele obiective ale Unităţii de învăţare Nr. 6 sunt:



După parcurgerea acestei Unităţii de învăţare vei fi capabil:

- Să identifici diferite formate de imagini suportate de browsere
- Să adaugi imagini la o pagină HTML
- Să specifici modul de aliniere al imaginii în pagină şi al textului din jurul ei
- Să specifici dimensiunea imaginii

6.1 Întelegerea formatelor grafice folosite în WEB

Limbajul HTML nu impune folosirea unui anumit format de imagine ci oferă în shimb o modalitate flexibilă de a include orice tip de imagine în pagina HTML. Afişarea unui anumit tip de imagine depinde însă de tipul browserului, de aceea există totuşi un număr limitat de formate grafice folosite în WEB., şi dintre acestea doar două sunt populare şi foarte des folosite: **GIF** şi **JPEG**.

GIF

Formatul grafic folosit prima oară în cadrul paginilor WEB şi care a rămas cel mai popular este formatul GIF. Denumirea de GIF reprezintă iniţialele expresiei *Graphic Interchange Format* care în limba română se traduce prin *"Format grafic pentru transfer"* .

Avantajele formatului grafic GIF sunt următoarele:

- Este independent de platformă. Aceasta însemnă că dacă o imagine este creată pe o maşină care rulează LINUX aceasta se va vedea la fel pe o maşină care rulează alt sistem de operare Windows sau Macintosh.
- Stocarea datelor se face comprimat ceea ce duce la fişiere de dimensiuni mici care se pot transfera rapid în reţea.
- Comprimarea datelor în cazul formatului grafic GIF se face folosindu-se un algoritm fără pierdere de date. Aceasta însemnă că dintr-o imagine in format GIF se poate restaura în orice moment imaginea originală necomprimată.

JPEG

Dezavantajul formatului GIF este acela că nu permite folosirea unui număr mai mare de 256 de culori pentru o imagine. Acest neajuns este înlăturat în cazul formatului **JPEG**.

Formatul grafic JPEG a fost creat de către "***J**oint **P**hotographic **E**xperts **G**roup*" care înseamnă în limba română "Grupul experţilor fotografi", şi este ca şi formatul GIF independent de platformă. Spre deosebire însă de GIF acest format suportă zeci de mii de culori pentru a putea reprezenta imagini fotografice cât mai aproape de realitate. Un alt avantaj şi dezavantaj în acelaşi timp al formatului JPEG faţă de GIF este acela că formatul JPEG foloseşte un algoritm cu ajutorul căruia se obţine o rată mai bună de compresie a datelor. Spre exemplu este un lucru obijnuit ca un fişier JPEG să aibă dimensiuni de până la 7-8 ori mai mici decât un fişier în format GIF care reprezintă aceeaşi imagine. Am spus dezavantaj în acelaşi timp deoarece această compresie se face spre deosebire de GIF cu pierdere de date. Aceasta înseamnă că o imagine transformată din format JPEG în format necomprimat nu va corespunde exact cu originalul ci vor fi anumite diferenţe. Aceste diferenţe însă nu se pot observa în mod normal cu ochiul liber.

Cum alegem între GIF şi JPEG? Ei bine, ambele formate sunt universal suportate de către majoritatea browserelor, şi prin urmare criteriul compatibităţii nu este unul puternic. Prin urmare vom incerca să folosim avantajele fiecărui format în parte.

6.2 Folosirea imaginilor în cadrul paginilor WEB

Pentru a introduce imagini într-o pagină HTML se foloseşte directiva HTML <IMG>.

<IMG>

Funcţionalitate:

- Introduce o imagine în pagină

Atribute:

- SRC
- ALT
- ALIGN
- BORDER
- HEIGHT
- WIDTH
- HSPACE
- VSPACE

Directiva de sfârşit:

- </IMG> este OPTIONALĂ

Directiva HTML IMG foloseşte valoarea atributului SRC pentru a identifica locul de unde va prelua imaginea care se doreşte a fi adăugată în pagină. Iată mai jos un exemplu cu cel mai simplu mod de utilizare al directivei IMG:

```
<P>
 Acesta este poza iepurasului roz:
 <IMG SRC="iepuras.gif" ALT="iepuras roz">
 </IMG>
</P>
```

va avea ca rezultat în browser:

Figura 6.1 Exemplu folosire directiva <IMG>

Folosirea atributului ALT nu este obligatorie însă este recomandată deoarece browserul se poate configura astfel încât să nu încarce imaginile din pagină, caz în care în locul imaginii este afişat textul oferit de atributul ALT. În mod ideal pagina ar trebui construită astfel încât să fie lizibilă chiar dacă imaginile nu pot fi încărcate.

Folosirea directivei <IMG> nu va introduce imaginea pe o linie nouă ci în continuarea textului. Implicit imaginea va fi aliniată astfel încât marginea inferioară a imaginii să corespundă cu marginea inferioară a textului precum se vede în Figura 6.1. Dacă se doreşte o altfel de aliniere verticală a imaginii sau a textului se vor folosi atributele directivei <IMG> după cum vom vedea mai jos.

Atributul **SRC.** Acest atribut este obligatoriu în cazul directivei <IMG>. Valoarea este un URL şi reprezintă locul în care se află fişierul ce conţine imaginea ce se doreseşte a fi inclusă în pagina.

Etichetare imagine

Atributul **ALT**. Dacă imaginea specificată de către atributul SRC nu poate fi încărcată din diferite motive: fişier imagine incomplet, conexiune intreruptă, URL specificat greşit sau browserul este configurat pentru a nu încarca imaginile, atunci vor fi afişate implicit în locul imaginii o altă imagine generică care să indice faptul că imaginea specificată nu a fost încărcată. Acest lucru nu este întotdeauna de dorit deoarece utlizatoul nu ar putea avea nici o

informaţie în legătură cu imaginea care nu s-a incărcat. În aceste cazuri textul specificat de atributul ALT va fi afişat în locul imaginii generice.

Teste autoevaluare

6.1. Cum se poate face într-o pagină HTML pentru ca imaginea *margareta.jpg* din directorul curent să fie afişată la începutul paginii?

6.2 Cum se poate face ca browserul să afişeze cuvântul “margareta” în locul imaginii dacă aceasta nu a putut fi încărcată de către browser?

a> ☐ `<IMG SRC= ”Margareta.jpg”>margareta`
b> ☐ `margareta<IMG SRC= ”margareta.jpg”>`
c> ☐ `<IMG SRC= ”margareta.jpg” ALT= ”margareta”>`
d> ☐ `<P>margareta<IMG SRC=”margareta.jpg”></P>`

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 70.

Atributul **ALIGN.** Cu ajutorul acestui atribut se poate specifica modalitatea de aliniere pe orizontală sau pe verticală a unei imagini faţă de textul sau altă imagine vecină.

Alinierea pe orizontală

Valorile disponibile pentru alinierea pe orizontală sunt :

- **ALIGN = ”LEFT”** va avea ca efect poziţionarea imaginii în stânga paginii, iar textul va încadra imaginea prin partea dreaptă a acesteia ca în figura 6.2.
- **ALIGN = “RIGHT”** va avea ca efect poziţionarea imaginii în dreapta paginii, iar textul va încadra imaginea prin partea stângă a acesteia ca în figura 6.2.

Dacă se doreşte la un momendat întreruperea încadrării imaginii de către text se foloseşte directiva
 împreună cu atributul său CLEAR.

```
...
<P>
<IMG  SRC="iepuras.gif"
      ALT="Bugs Bunny"
      ALIGN="left">
Un iepuras ... 9 x 9 x 19 h
<BR CLEAR=”LEFT”>
...
<IMG  SRC="iepuras.gif"
```

```
      ALT="Bugs Bunny"
      ALIGN="RIGHT">
Un iepuras ... 9 x 9 x 19 h
<BR CLEAR=RIGHT>
...
```

Va avea ca rezultat în browser:

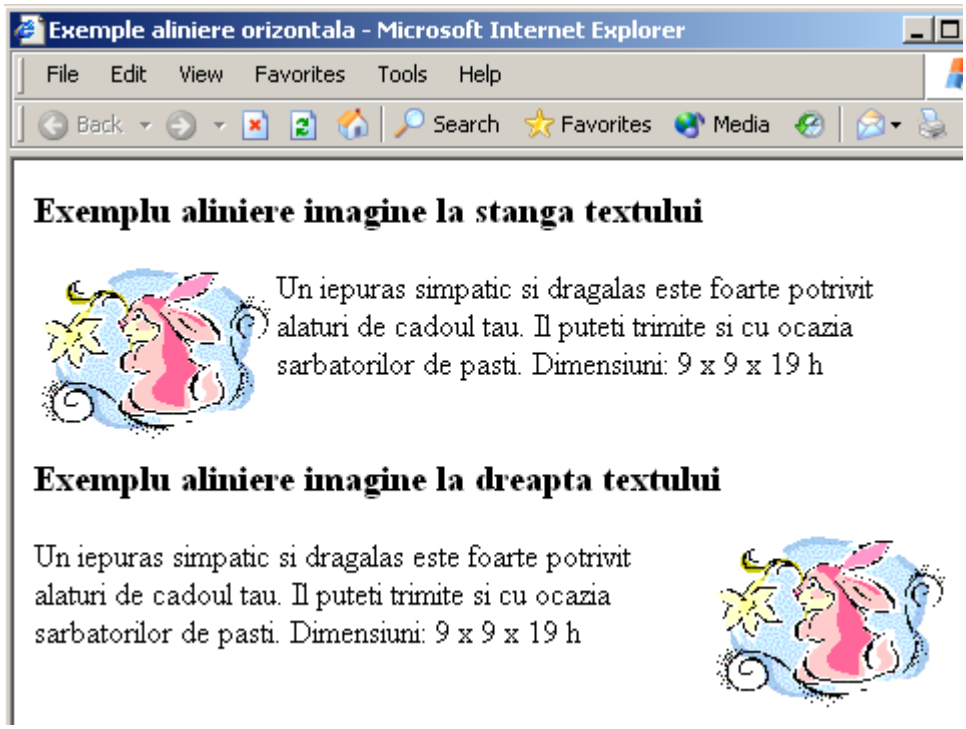


Figura 6.2 Exemplu aliniere orizontală la stanga si la dreapta

Alinierea pe verticală

Pentru specificarea modului de aliniere pe verticală a imaginii faţă de textul vecin sau faţă de alte imagini pe aceeaşi linie se foloseşte de asemenea atributul ALIGN următoarele valori fiind disponibile:

- **ALIGN=”TOP”** – partea superioară a imaginii va fi la acelaşi nivel cu partea superioară a altei imagini sau cu a celei mai înalte litere din textul de pe aceeaşi linie.
- **ALIGN=”MIDLLE”** – mijlocul imaginii va corespunde cu mijlocul altei imagini sau cu mijlocul textului de pe aceeaşi linie.
- **ALIGN=”BOTTOM”** – partea inferioară a imaginii va corespunde cu partea inforioară a textului de pe aceeaşi linie.

```
...
Iepuras sus
<IMG SRC="iepuras.gif"
     ALIGN="TOP">. Ce dragut este!
<BR CLEAR="ALL">
Iepuras la mijloc
<IMG SRC="iepuras.gif"
     ALIGN="MIDDLE">Ce dragut este!
<BR CLEAR="ALL">
```

```
Iepuras jos
<IMG SRC="iepuras.gif"
     ALIGN="BOTTOM">Ce dragut este!
...
```

Va avea ca rezultat în browser:

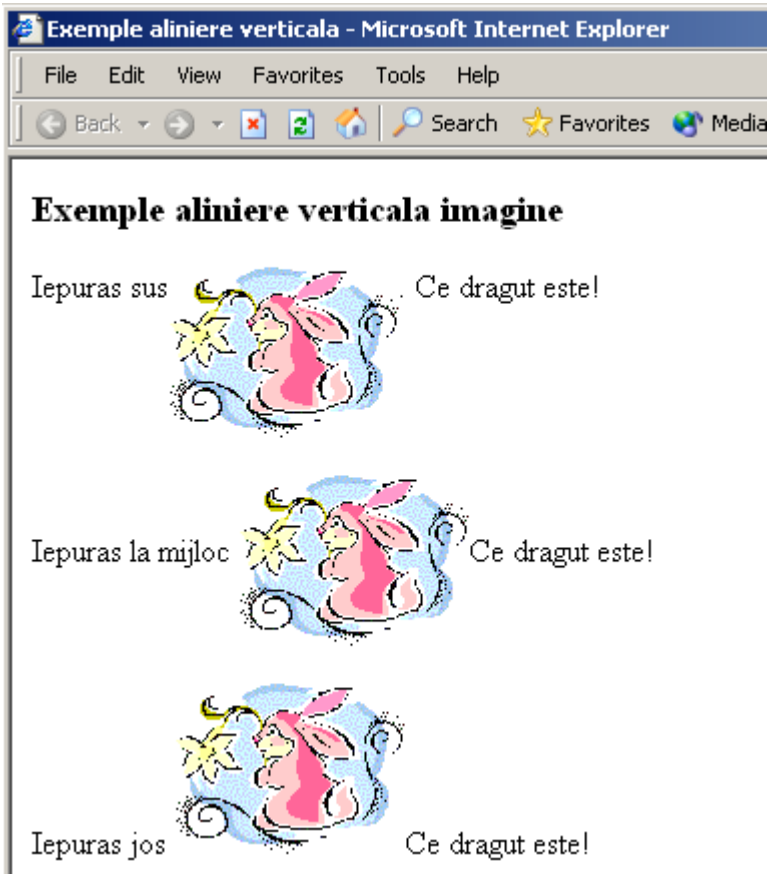


Figura 6.3 Exemple aliniere verticală.

Teste autoevaluare

6.3 Cum se scrie codul HTML care să afişeze o imagine centrată fără text la stânga şi la dreapta ei ?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 70.

Dimensiunea imaginilor

Atributele **HEIGHT** şi **WIDTH** în mod normal browserul determină dimensiunea imaginii şi în funcţie de acesta rezervă un spaţiu corespunzător pentru imagine în pagină pentru fiecare imagine în

parte. Totuşi aceste informaţii nu pot fi determinate decât în momentul în care imaginea a fost complet adusă de pa server. Acest lucru poate face ca timpul necesar pentru afişarea paginii în browser să crească. Dacă aceste informaţii legate de dimensiunile imaginii sunt specificate prin atributele HEIGHT şi WIDTH timpul necesar pentru ca browserul să afişeze pagina va vi mult mai mic (chiar dacă imaginile vor fi afişate în pagină mai târziu, în momentul în care au fost complet încărcate).

Valoarea atributelor HEIGHT şi WITH poate fi un întreg pozitiv şi reprezintă numărul de pixeli pe verticală respectiv pe diagonală pe care îi ocupă imaginea.

Dacă valorile acestor atribute nu corespund cu dimensiunile imaginii browserul va redimensiona imaginea afişată astfel încât să fie conform cu valorile specificate de atribute.

Redimensionarea imaginilor trebuie făcută cu grijă deoarece specificararea unor valori foarte îndepărtate de valoarea originală a imaginii poate rezulta în afişarea unor imagini distorsionate sau cu o rezoluţie nepotrivită.

Dimensiunea spatiilor

Atributele **HSPACE** şi **VSPACE.** Permite definirea spaţiilor între imagine şi textul sau obiectele vecine. Valoarea acestor atribute poate fi un întreg pozitiv şi reprezintă numărul de pixeli ce va fi lăsat între imagine şi textul ce o înconjoară la stânga sau dreapta ei respectiv deasupra sau dedesubtul ei.

6.3 Imagini cu arii sensibile (Image Maps)

Aşa cum am vazut în secţiunea anterioară imaginile pot fi folosite pentru hyperlink prin simpla adăugare a directivei <IMG> în interiorul directivei <A>. Dacă utilizatorul va selecta oricare din punctele imaginii respective se va deschide pagina de la URL-ul specificat de valoarea atributului HREF. HTML oferă în plus o modalitate specială de folosire a imaginilor ca hyperlink permiţând definirea unor suprafeţe în cadrul imaginii care să aibă fiecare asociat câte un URL separat.. Definirea acestor suprafeţe se face cu ajutorul directivelor <MAP> şi <AREA>.

<MAP>	<AREA>
Funcţionalitate: • Defineşte o hartă de suprafeţe sensibile **Atribute:** • NAME **Directiva de sfârşit:** • </MAP> este OBLIGATORIE	**Funcţionalitate:** • Defineşte o suprafaţă sensibilă în cadrul unei hărţi **Atribute:** • SHAPE • COORDS • HREF • NOHREF **Directiva de sfârşit:** </AREA> este OPTIONALĂ

Iată paşii pentru definirea suprafeţelor sensibile pentru o anume imagine:

- Utilizarea în cadrul directivei IMG a atributelor ISMAP şi USEMAP

```
<IMG  SRC=”imagine.gif”
      ISMAP
      USEMAP=”#harta”>
```

- Definirea harţii specificate ca valoarea a atributului MAP cu ajutorul directivelor MAP şi AREA

```
<map  name="harta">
 <area  coords="0,25,50,50"
        href="link1.html">
 <area  coords="0,0,25,25"
        href="link2.html"
</map>
```

Rezultatul va fi o imagine care va avea 2 arii sensibile una în partea stângă şi una în partea dreaptă a imaginii. Dacă utilizatorul va selecta partea stângă browserul va încărca pagina “*link1.html*”, dacă se va selecta aria dreaptă browserul va încărca pagina “*link2.html*” .

Directiva MAP

Directiva MAP are un singur atribut, NAME, cu ajutorul căruia această hartă poate fi identificată şi folosită pentru o anumită imagine. Definirea suprafeţelor sensibile pentru fiecare hartă în parte se face folosind o succesiune de directive <AREA> şi a atributelor acesteia după cum vom vedea mai jos.

Directiva AREA

Directiva AREA defineşte fiecare din suprafeţele sensibile ale unei harţi. Cu ajutorul atributelor sale se pot defini suprafeţe cu diferite

forme şi dimensiuni şi se pot asocia hyperlinkuri pentru fiecare în parte.

Atributul **COORDS** permite definirea limitelor suprafeţei, valoarea sa fiind reprezentată de o serie de numere întregi separate prin virgulă. Număul şi semnificaţia acestor valori este în funcţie de valoarea atributului SHAPE.
Atributul **SHAPE**. Permite definirea formei pentru zona sensibilă. Mai jos sunt prezentate formele ce pot fi specificate de către acest atribut şi valoarea corespunzătoare şi de asemenea este explicat pentru fiecare în parte cum este interpretat :

- RECTANGLE – defineşte o zonă dreptunghiulară.

Coordonatele se specifică sub forma: COORS=”x,y,z” unde

- CIRCLE – defineşte o zonă circulară
- POLYGON – defineşte un poligon neregulat şi închis cu un număr oareacare de laturi.

Lucrare de verificare a Unităţilor de învăţare Nr. 4, 5 şi 6

1. Creaţi o listă precum cea de mai jos din Figura 4.4 . Lista conţine două nivele imbricate. La primul nivel avem o listă de definiţii. Definiţia pentru primul termen este o listă neordonată, iar definitia celui de-al doilea termen este o listă ordonată. Termenii listei de definitie trebuie să fie afişaţi îngroşat.

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 2 din bibliografia unităţii de învăţare.

Nr. de puncte **8** (4p - realizarea listei de definitie conform cerinţelor 2p – realizarea listei neordonate ca definiţie pentru primul termen 2p – realizarea listei ordonate ca definiţie pentru cel de-al doilea termen)

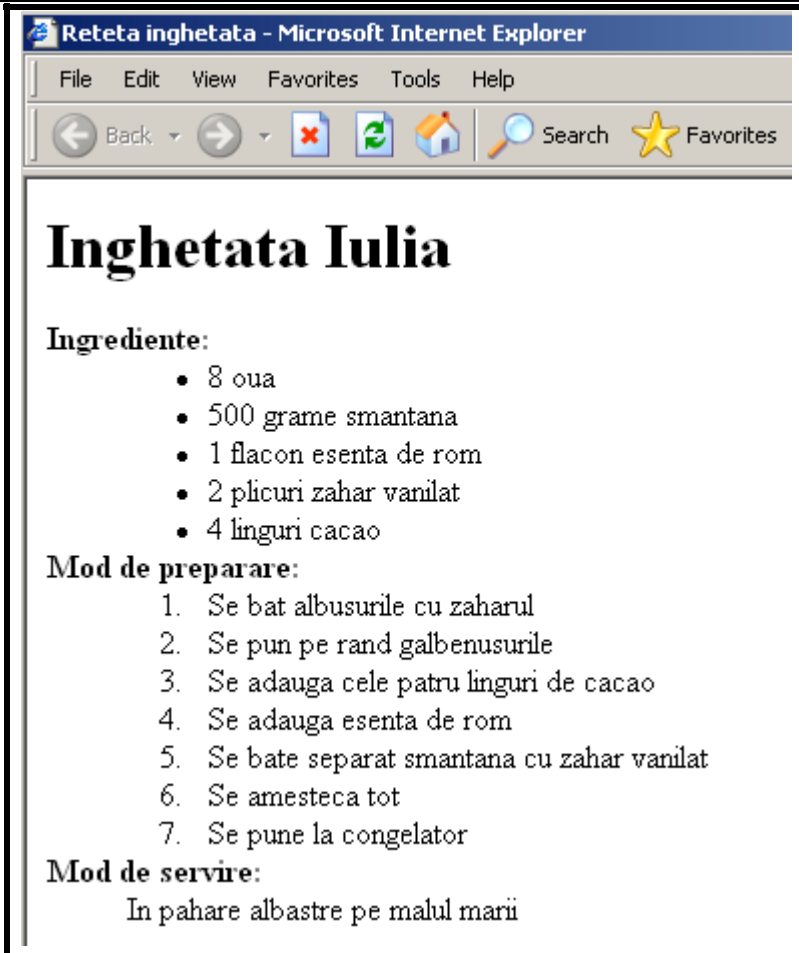


Figura 4.4 Lucrarea de verificare

2. Realizaţi un meniu pentru un site Web de prezentare a unei firme. Site-ul are următoarele pagini: *acasa.html*, *produse.html*, *servicii.html* şi *contact.html*. În fiecare din acestea trebuie să fie o legătură la oricare altă pagină a site-ului. Încercaţi să îl faceţi cât mai atractiv cu putinţă!

Predaţi cele 4 fişiere HTML în format electronic ca rezultat al rezolvării lucrării de verificare.

Nr de puncte **9** (6p – definirea corectă, în fiecare pagină a legăturilor către celelalte pagini html, 3p - pentru aspectul paginii şi ingeniozitate.

Ca ajutor suplimentar utilizati reperele bibliografice 2 din bibliografia unităţii de învăţare.

3. Realizati o listă HTML neordonată care să aibă în locul marcajelor standard imaginea conţinută în fişierul *bulina.jpg* Sugestie: Folosiţi o listă de definiţie pentru a simula lista neordonată cerută.

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 2 şi 3 din bibliografia unităţii de învăţare.

Nr. de puncte **8** (4p - realizarea listei, 4p – folosirea imaginii cerute în locul semnelor predefinite pentru listele neordonate)

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 6.1.
Codul HTML care realizează acest lucru este următorul:

```
...
<BODY>
<P >
 <IMG SRC="margareta.jpg"> </IMG>
</P>
```

Directiva IMG este pusă imediat după directiva BODY pentru ca imaginea să fie afişată la începutul paginii. Revedeţi indicaţiile din secţiunea 6.2.

Întrebarea 6.2.

Varianta corectă de răspuns este **c.** Pentru aceasta se foloseşte atributul ALT pentru a specifica textul înlocuitor ca în exemplul de mai jos:

```
<IMG SRC="margareta.jpg"
     ALT=”margareta”> </IMG>
```

Revedeţi indicaţiile din secţiunea 6.2.

Întrebarea 6.3.

Pentru a realiza acest lucru folosim directiva BR împreună cu atributul CLEAR pentru a evita înconurarea imaginii cu text şi directiva <P> împreună cu atributul ALIGN pentru a centra imaginea în pagină ca în exemplul de mai jos:

```
<BR CLEAR="ALL">
<P ALIGN=”CENTER”>
 <IMG SRC="margareta.jpg"
      ALT=”margareta”> </IMG>
</P>
```

Revedeţi indicaţiile din secţiunea 6.2.

Indicaţii la problemele propuse

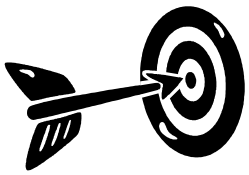
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.127-169
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.46-47
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000, pg.51-59

Unitatea de învăţare Nr. 7
REALIZAREA TABELELOR

Obiectivele Unităţii de învăţare Nr.7 :



După parcurgerea acestei Unităţii de învăţare vei sti:

- Care este structra unui tabel HTML
- Care sunt elementele unui tabel HTML
- Să creezi un tabel HTML
- Să schimbi proprietăţile unui tabel

Scopul iniţial al tabelelor în limbajul HTML a fost pentru prezentarea anumitor date organizate în format tabular. Suportul oferit de către HTML pentru tabele s-a dovedit însă foarte eficient la aranjarea în pagină a diferitelor elemente HTML. Cu ajutorul directivelor oferite de către limbajul HTML este posibilă poziţionarea a practic oricărui tip de element HTML la pozitia dorită în pagină.

Componentele unui tabel

Hai să enumerăm şi să descriem fiecare componentă a unui tabel în parte:

- **ROW** – Se referă la rândul unui tabel
- **COLUMN** – Se referă la coloana unui tabel
- **CELL** – Se referă la intersecţia dintre o linie şi o coloană
- **CAPTION** – Se referă la un text explicativ cu privire la conţinutul tabelului care apare deasupra tabelului
- **HEADERS** – Se referă la primul rând al tabelului sau antetul tabelului
- **BORDERS** – Se referă la liniile de delimitare care înconjoară o celulă a tabelului sau întreg tabelul.

7.1 Realizarea unui tabel simplu în limbajul HTML

În limbajul HTML, un tabel simplu se poate defini cu ajutorul directivei <TABLE>; cu ajutorul directivei <TR> tabelul este împărţit în linii, iar cu ajutorul directivei <TD> fiecare linie este împărţită în celule.

Orice tabel în limbajul HTML va începe întotdeauna cu următoarea directivă pereche cu rol de container:

```
<TABLE>
</TABLE>
```

Paşi

După directiva <TABLE> următorii paşi sunt necesari pentru construirea tabelului:

Pasul 1. *Adăugarea unui rând*. – Se face folosind directiva pereche <TR> </TR> în intoriorul elementului <TABLE>

```
<TABLE>
  <TR>
  </TR>
</TABLE>
```

Pasul 2. *Împărţirea rândului într-un număr de coloane* – Se face folosind directiva pereche <TD></TD> în interiorul unui element <TR>. Fiecare combinaţie <TD></TD> reprezintă o coloană/celulă a tabelului. Spre exemplu dacă tabelul are 3 coloane vom scrie:

```
<TABLE>
  <TR>
    <TD></TD>
    <TD></TD>
    <TD></TD>
  </TR>
</TABLE>
```

Pasul 3. *Introducerea datelor în fiecare celulă a tabelului*. În fiecare celulă a tabelului definită mai sus se introduce textul sau elementele HTML pe care vrem să le afişăm în tabel, precum în exemplul de mai jos:

```
 <TABLE>
  <TR>
    <TD>Iepure</TD>
    <TD>25 Kg</TD>
    <TD><IMG SRC=”iepuras.gif”></TD>
  </TR>
</TABLE>
```

Într-o celulă a unui tabel se poate pune aproape orice element HTML: text formatat, imagini, liste, hyperlink etc. În exemplul de mai sus ultima coloană conţine imagine.

Pasul 4. *Repetarea paşilor 1,2, 3 până ce tabelul este complet*

Test de autoevaluare

7.1 Introduceţi un nou rând în tabelul de mai sus

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 82.

7.2 Definirea proprietăţilor globale ale unui tabel HTML

Directiva HTML <TABLE> semnalează browserului faptul că urmează definiţia unui tabel. La întâlnirea acesteia browserul va trece automat la o linie nouă unde va poziţiona tabelul definit urmând ca la întâlnirea directivei de sfârşit </TABLE> să sară din nou la o linie nouă unde va afişa textul ce urmează tabelului.

Caracteristicile globale ale tabelului precum: modul de aliniere, dimensiuni, culori pot fi specificate folosind atributele acestei directive după cum vom vedea mai jos.

<TABLE>

Funcţionalitate:
- Defineşte un tabel

Atribute:
- ALIGN
- BORDER
- BGCOLOR
- WIDTH
- HEIGHT

Directiva de sfârşit:
- </TABLE> este OBLIGATORIE

Atributul ALIGN

Atributul **ALIGN**. Specifică unde anume va fi poziţionat tabelul în pagină: aliniat la stânga (“LEFT”), la dreapta (“RIGHT”) sau pe centru (“CENTER”). Dacă acest atribut nu este specificat tabelul va fi aliniat la stânga.

Atributul VALIGN

Atributul **VALIGN.** Specifică modul de aliniere al textului din cadrul celulelor tabelului. Valorile posibile pentru atribut sunt “TOP” pentru

aliniere la partea superioară a celulei, “BOTTOM” aliniere la partea inferioară a celulei şi “CENTER” aliniere la centru celulei.

Atributul BORDER

Atributul **BORDER**. Specifică dimensiunea liniilor de delimitare care încadreaza celulele tabelului. Valoarea atributului este un intreg între 0 şi 15 şi reprezită grosimea în pixeli a liniei. Specificarea valorii “0” va face ca aceste linii de delimitare să fie invizibile în browser. Dacă atributul nu este specificat dimensiunea implicită este de 1 pixel.

Atributul CELLSPACING

Atributul **CELLSPACING**. Specifică spaţiul dintre două celule adiacente ale tabelului. Dacă nu este specificat valoarea implicită a acestui atribut este de 2 pixeli.

Atributul CELLPADDING

Atributul **CELLPADDING** – Specifică spaţiul minim în număr de pixeli dintre marginea unei celule şi conţinutul său. Implicit acest atribut are valoarea de un pixel.

Toate atributele de mai sus se pot folosi în acelaşi timp pentru a obţine aspectul dorit pentru tabel.

Iată mai jos un exemplu de folosirea a acestor atribute. S-a folosit o valoare intenţionat exagerată pentru a identifica în figura rezultată rolul fiecăruia în parte. De asemenea în aceeaşi pagină a fost definit un tabel pentru care aceste atribute nu au fost specificate şi deci pentru care browserul a folosit la afişarea lui valorile implicite.

```
...
<TABLE BORDER="1">
 <TR>
   <TD>celula11</TD>
   <TD>celula12</TD>
 </TR>
 <TR>
   <TD>celula21</TD>
   <TD>celula22</TD>
 </TR>
</TABLE>
...

<TABLE
BORDER="15"
CELLPADDING="15"
CELLSPACING="15"
>
 <TR>
   <TD>celula11</TD>
   <TD>celula12</TD>
 </TR>
 <TR>
 <TD>celula21</TD>
 <TD>celula22</TD>
 </TR>
</TABLE>
...
```

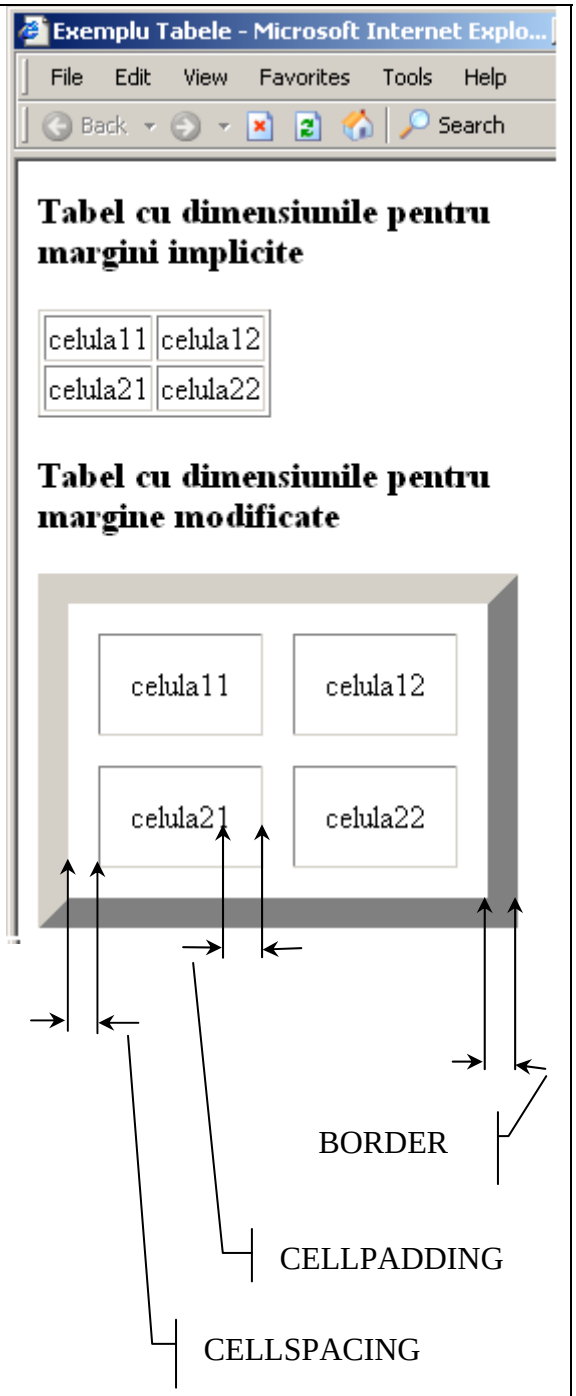


Figura 7.1 Exemplu folosire atribute pentru directiva TABLE

Atributele **WIDTH** şi **HEIGHT**. În mod implicit browserul va dimensiona tabelul în funcţie de dimensiunea ferestrei browserului şi în funcţie de dimensiunea conţinutului tabelului. Dacă este necesar se poate specifica o dimensiune explicită a tabelului cu ajutorul atributului WIDTH. Această dimensiune poate fi fie o valoare absolută, adică un întreg reprezentând numărul de pixeli ocupaţi de tabel pe orizontală, fie o valoare procentuală caz în care browserul va afişa tabelul ocupând procentul specificat din dimensiunea ferestrei browserului.

Spre exemplu dacă se vrea ca dimensiunea tabelului să fie de 100 pixeli vom scrie:

```
<TABLE WIDTH=”100”>
```

sau dacă dorim ca dimensiunea tabelului să fie jumătate din dimensiunea browserului vom scrie:

```
<TABLE WIDTH=”50%”>
```

Dacă dimensiunea tabelului este mai mare decât valoarea specificată de către atributul WIDTH browserul ca ignora acest atribut şi va dimensiona tabelul.

Similar atributul HEIGHT poate fi folosit pentru a specifica înălţimea tabelului. Browserul va afişa tabelul astfel încât înălţimea lui să nu fie mai mică decât valoarea acestui atribut.

Atributul NOWRAP

Atributul **NOWRAP.** Textul dintr-o celulă a unui tabel este “rupt” şi se trece la linie nouă[3] în momentul în care dimensiune acestuia este mai mare decât lungimea celulei. Dacă este specificat atributul NOWRAP atunci browserul va dimensiona celulele astfel încât să nu fie nevoie ca textul conţinut în nici una din ele sa nu fie “rupt”. Dacă se doreşte trecerea la o linie nouă în cadrul unei celule se poate folosi una din directivele HTML
 sau <P>.

Test de autoevaluare

7.2 Cum se defineşte un tabel care să aibă laţimea 80% din fereastra browserului şi care să fie afişat centrat în pagină?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 82.

7.3 Definirea rândurilor unui tabel

Pentru a insera un nou rând într-un tabel se foloseşte directiva <TR> a cărui denumire vine de la *“Table Row”*.[4]

[3] Operaţiunea se numeşte în limba engleză WRAP
[4] Rând de tabel

<TR >

Funcţionalitate:

- Defineşte un rând într-un tabel

Atribute:

- ALIGN
- VALIGN
- BGCOLOR
- NOWRAP

Directiva de sfârşit:

- </TR> este opţională

Atributul ALIGN

Atributul **ALIGN** . Permite configurarea modului de aliniere a textului pentru toate celule dintr-un rând. Valoarea acestuia nu afectează însă modul de aliniere pentru celule din alt rând decât cel curent. Valorile posibile sunt “LEFT” pentru stânga, “RIGHT” pentru dreapta şi “CENTER” pentru aliniere la centru.
Restul atributelor au aceeaşi efect ca a atributelor directivei TABLE despre care tocmai am discutat.

7.4 Definirea celulelor unui tabel

Directiva <TD> este folosită în cadrul directivei <TR> pentru a defini o celulă a tabelului şi conţinutul acesteia. Există cazuri în care într-un tabel primul rând este mai special deoarece conţine informaţii despre tipul datelor conţinute în tabel Acest rând poartă denumirea de antet. În HTML acest rând se poate defini cu ajutorul directivei <TH>

Cele două directive acţionează asemănător şi au aceleaşi atribute diferenţa între ele fiind faptul că în cazul directivei <TH> textul este afişat bold şi centrat (dacă nu este specificat altfel) iar în cazul <TD> textul este afişat aliniat la stânga (dacă nu este specificat altfel);

<TD > sau <TH>
Funcţionalitate: • Defineşte o celulă a unui tabel **Atribute:** • ALIGN • VALIGN • COLSPAN • ROWSPAN • WIDTH • HEIGHT • BGCOLOR • BACKGROUND **Directiva de sfârşit:** • </TD> respectiv </TH> sunt opţionale

Atributul COLSPAN

Atributul **COLSPAN** este folosit pentru a unui celulele învecinate ale unui rând. Valoarea acestui atribut indică numărul de celule de pe rândul curent care vor fi unite astfel încât să formeze o singură celulă.

Spre exemplu dacă vrei ca pe primul rând într-un tabel cu 4 coloane să fie titlul tabelului care să ocupe toată lungimea tabelului şi nu numai o celulă se poate folosi atribut COLSPAN atât pentru <TH> cât si pentru <TD> astfel:

```
<TABEL>
<TR>
  <TH COLSPAN="4">
   Titlul principal
  </TH>
</TR>
 <TR>
  <TD COLSPAN=”4”>
    Subtitlul tabelului
  </TD>
</TR>
<TR>
 <TD>Col1</TD>
 <TD>Col2</TD>
 <TD>Col3</TD>
 <TD>Col4</TD>
</TR>
</TABEL>
```

Rezultatul în browser va fi de forma:

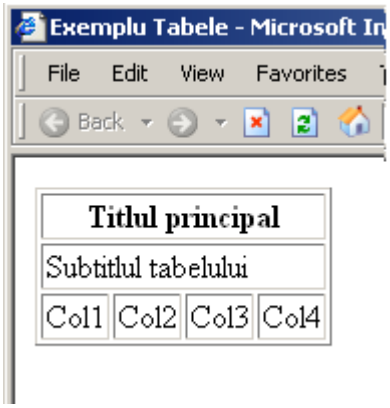


Figura 7.3 Exemplu utilizare COLSPAN

Atributul ROWSPAN

Atributul **ROWSPAN** este folosit pentru a unii mai multe celule învecinate de pe aceeaşi coloană. Se foloseşte similar cu atributul COLSPAN. Prin urmare dacă dorim să extindem o celulă pe mai multe rânduri vom scrie:

```
...
<TR>
 <TD ROWSPAN=”2”>
  Celula12+21</TD>
 <TD>Celula12<TD>
</TR>
<TR>
 <TD>Celula22<>
</TR>
...
```

Si va avea ca rezultat în browser:

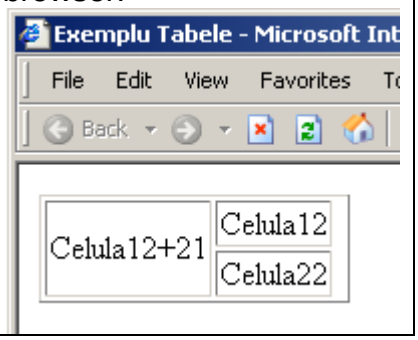


Figura 7.4 Exemplu utilizare ROWSPAN

O celulă se poate extinde pe mai multe celule învecinate de pe acelaşi rând şi în acelaşi timp pe mai multe celule învecinate de pe mai multe rânduri. Acest efect se obţine combinând cele două atribute în cadrul aceleiaşi directive <TD> sau <TH>.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 7.1.
Codul HTML pentru tabelul cu un nou rând adăugat va arăta ca cel de mai jos

```
<TABLE>
  <TR>
    <TD>Iepure</TD>
    <TD>25 Kg</TD>
    <TD><IMG SRC=”iepuras.gif”></TD>
  </TR>
  <TR>
    <TD>Pisica</TD>
    <TD>10 Kg</TD>
    <TD><IMG SRC=”pisica.gif”></TD>
  </TR>
</TABLE>
```

Pentru nelămuriri revedeţi secţiunea 7.1.

Întrebarea 7.2.
Pentru a afişa tabelul cerut se pot folosi atributele directivei TABLE ca în exemplul de mai jos:

```
<TABLE ALIGN=”CENTER” WIDTH=”80%”>
```

A se revedea secţiunea 7.2.

Indicaţii la problemele propuse

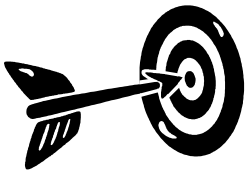
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.171-198
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.32-36
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000, pg.39-50

Unitatea de învăţare Nr. 8
FORMULARE ÎN PAGINA WEB

Obiectivele Unităţii de învăţare Nr. 8:



După parcurgerea acestei Unităţii de învăţare veţi fi capabili:

- Să creeaţi un formular
- Să cunoaşteţi elementele unui formular
- Să adaugaţi elemente la un formular
- Să specificaţi o acţiune pentru un formular
- Să înţelegeţi ce înseamnă CGI

8.1 Ce sunt formularele?

Formularele sunt o metodă de colectare a datelor de la utilizator cu scopul prelucrării lor sau a stocării într-o bază de date pentru o prelucrare viitoare.

Formularele în WEB sunt compuse din obiecte ce permit introducerea de text, selectarea unor opţiuni, liste de selecţie, harţi de imagine sau butoane. Vom discuta despre fiecare în parte în cadrul acestei unităţi de învăţare.

Să vedem care este fluxul de operaţii şi de date pentru o pagină WEB care conţine un formular (Vezi figura 8.1):

Fluxul de operaţii/date

1. Utilizatorul va introduce informaţii în cadrul formularului
2. Utilizatorul va apăsa un buton special în cadrul formularului
3. Formularul va fi trimis la server
4. Serverul va primi informaţia şi o va prelucra
5. Serverul va trimite înapoi la browser o pagină de răspuns ce poate conţine un rezultat al prelucrării datelor trimise.

Prelucrarea informaţiei şi alcătuirea răspunsului pe partea de server este o problemă relativ complexă şi nu vom discuta despre ea în cadrul acestui modul. Această prelucrare se face folosindu-se un program ce poartă denumirea de **CGI.** CGI reprezită iniţialele de la *Common Gateway Interface* care se poate traduce în limba română prin interfaţă comună pentru schimb de date. Un program sau script CGI poate fi un program scris în orice limbaj de programare. Cele mai folosite limbaje sunt: C/C++, Perl, Python, sau anumite limbaje specializate pentru prelucrarea informaţiilor WEB cum ar fi PHP, ASP, JSP sau altele.

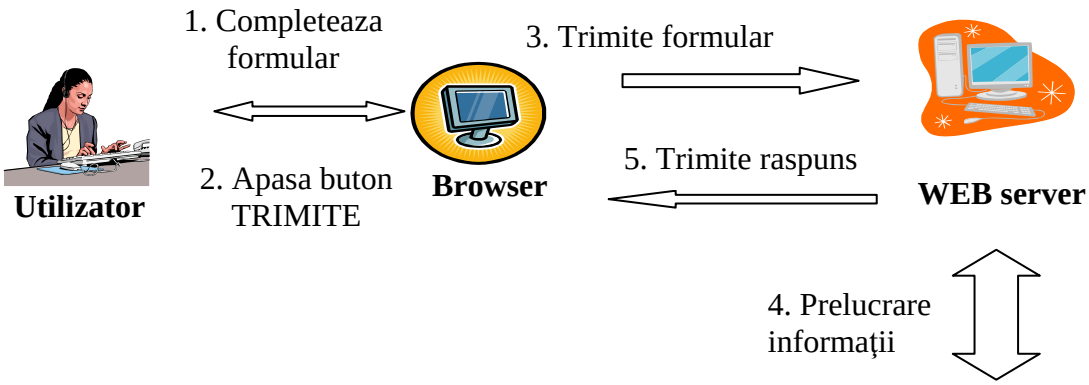


Figura 8.1 Prelucrare formular la server

CGI

În scenariu descris mai sus informaţia din formular este trimisă la un server spre a fi prelucrată, există însă şi posibilitatea ca această informaţie să fie prelucrată de către browser. Prelucrarea se face în acest caz cu ajutorul unui limbaj de script pe partea de client despre care vei învăţa în unitatea de învăţare numărul 9. Fluxul operaţiilor pentru acest caz este prezentată în figura 8.2.

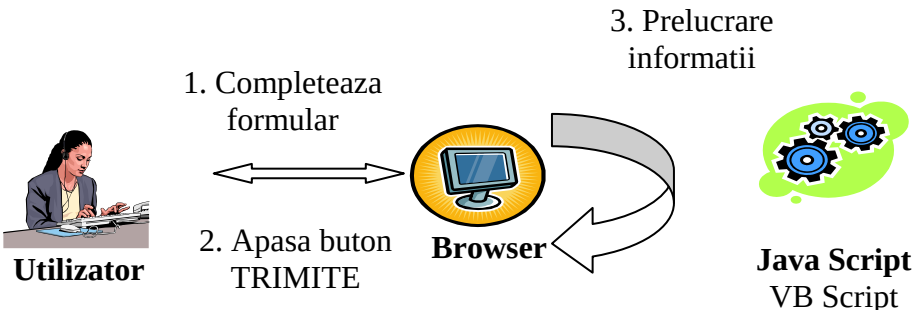


Figura 8.2 Formular prelucrat la client

Test de autoevaluare

8.1 Ce se întâmplă cu datele introduse de către utilizator într-un formular?

Răspunsul se va da în spaţiul gol de mai sus. Răspunsurile se găsesc la pagina 93.

8.2 Introducerea unui formular în pagina WEB

Pentru a introduce un formular într-o pagină WEB se foloseşte directiva pereche <FORM></FORM>. Rolul acestei directive este de a delimita formularul şi de a defini cu ajutorul atributelor sale modul şi de către cine vor fi prelucrate informaţiile introduse de către utilizator.

<FORM>

Funcţionalitate:
- Defineşte un formular

Atribute:
- ACTION
- METHOD
- NAME
- TARGET

Directiva de sfârşit:
- </FORM> - OBLIGATORIE

- Toate elementele unui formular trebuiesc incluse în cadrul unui element de tip <FORM>.
- Elementele unui formular vor fi afişate în fereastra browserului chiar dacă nu sunt incluse într-un formular însă în acest caz informaţiile introduse NU pot fi prelucrate.
- Un formular NU poate fi definit în interiorul altui formular
- O pagină WEB poate conţine mai multe formulare în acelaşi timp.

În figura de mai jos (Figura 8.3) avem un formular simplu aşa cum este afişat de către browser. Vom învăţa în această secţiune cum se poate realiza un astfel de formular şi chiar altele mai complexe:

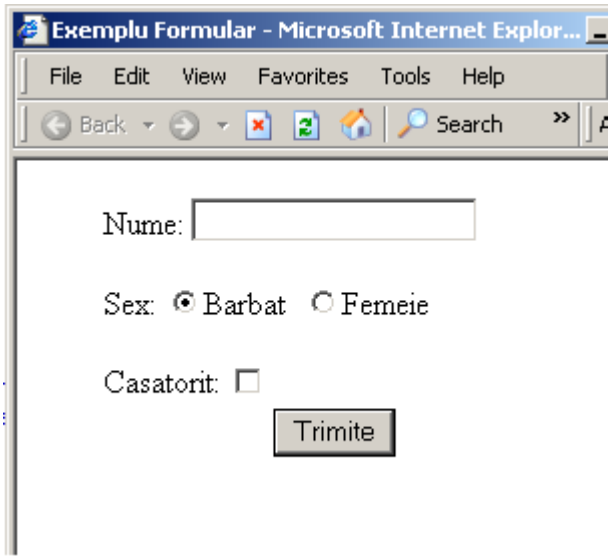


Figura 8.3 Exemplu simpu formular

Atributul ACTION

Atributul **ACTION**. Serverul de WEB va alege scriptul CGI care va prelucra informaţia din formular în funcţie de valoarea acestui atribut.

Atributul METHOD

Atributul **METHOD** se referă la modul în care valorile elementelor din formular vor fi transmise la browser. Există două metode:

- **POST** – Această metodă are 2 etape de transmitere a informaţiilor. Prima etapă constă în stabilirea unei conexiuni cu URL-ul specificat de către atributul ACTION. Odată stabilită conexiunea a doua etapă constă în transmiterea informaţiilor din formular la server
- **GET** – informaţiile din formular sunt adăugate la URL-ul specificat de către atributul ACTION. Scriptul CGI va prelua aceste informaţii din interiorul URL-ului.

Când se foloseşte metoda GET şi când metoda POST? Iată câteva reguli:

- Dacă este importantă viteza de transmisie a informaţiilor atunci va fi folosită metoda GET
- În cazul metodei GET preluarea informaţiilor de către aplicaţia server de prelucrare se face mai uşor
- Dacă securitatea este o problemă atunci este de preferat folosirea metodei POST deoarece în cazul metodei GET informaţiile pot fi citite din URL de către persoane neautorizate.

În exemplul de mai jos definim un formular care va transmite informaţiile la URL-ul “http://www.despremine.ro” folosind metoda GET

```
<FORM ACTION=”http://www.despremine.ro”
      METHOD=”GET”
>
...
</FORM>
```

Atributul TARGET

Atributul **TARGET** Cu ajutorul acestui atribut se poate redirecţiona rezultatul prelucrării formularului într-o altă fereastră de browser.

Test de autoevaluare

8.2 Există mai multe metode de transmitere a datelor dintr-un formular la server? Dacă răspunsul este “DA” care este cea mai nesigură metodă?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 93.

8.3 Elementele HTML folosite în formulare

Elementele unui formular se introduc folosind directiva HTML <INPUT>. Tipul câmpului din formular şi proprietăţile acestuia sunt specificate cu ajutorul atributelor directivei după cum vom vedea mai jos.

<INPUT>

Funcţionalitate:

- Defineşte un câmp într-un formular

Atribute:

- TYPE
- NAME
- ALIGN
- MAXLENGTH
- SIZE
- CHECKED

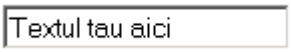
Directiva de sfârşit:

- </INPUT> este opţională

Câmpul TEXTBOX

TEXTBOX – sunt câmpuri de tip text care permit utilizatorului introducerea unui text pe un singur rând. Este util în colectarea informaţiilor de tip nume, adresă, dată, telefon, e-mail şi multe altele

```
<INPUT TYPE=”TEXTBOX”>
```

Browserul va afişa: 

Câmpurile TEXTBOX folosesc următoarele atribute:

- **NAME**: Numele variabilei ce conţine textul introdus de utilizator ce va fi trimis către scriptul CGI
- **SIZE** specifică lungimea vizibilă a câmpului text. Dimensiunea implicită, în cazul în care acest câmp nu este specificat este 20
- **VALUE** textul implicit ce va fi afişat în acest câmp atunci când acesta va fi prima oară afişat de către browser.
- **MAXLENGTH** specifică numărul maxim de caractere pe care le poate accepta câmpul.

Câmpul PASSWORD

PASSWORD – Sunt câmpuri folosite pentru introducerea de parole. Textul introdus de câtre utilizator nu va fi vizibil în interiorul acestui câmp. Pentru acest câmp se pot folosi aceleaşi atribute cu aceeaşi semnificaţie ca pentru câmpul Textbox

```
<INPUT TYPE=”PASSWORD”>
```

Browserul va afişa: 

Se observă că pentru fiecare caracter introdus de utilizator se afişează un caracter "•" sau "∗" în loc. Aceasta cu scopul de a proteja informaţia introdusă în acest câmp de alte persoane neautorizate care văd ecranul utilizatorului în timp ce acesta introduce parola.

Câmpul CHECKBOX

CHECKBOX Este un câmp ce poate avea două stări "*Selectat*" – "CHECK" sau "*Neselectat*". Se foloseşte atunci când este necesară obţinerea unei informaţii de tip adevărat/fals da/nu de la utilizator

```
<INPUT TYPE=”CHECKBOX”>
```

Browserul va afişa:  - neselectat

 - selectat

Câmpurile CHECKBOX folosesc următoarele atribute:

- **CHECKED** dacă acest atribut este prezent atunci la încărcarea paginii acest câmp va fi selectat implicit
- **NAME** –numele variabilei cu valoarea corespunzătoare acestui câmp ce va fi trimisă la scriptul CGI.

Câmpul RADIOBUTTON

RADIOBUTTON Permite utilizatorului să selecteze la un moment dat doar o singură opţiune dintr-un grup de opţiuni disponibile. Dacă un grup de câmpuri radiobutton au celaşi nume numai unul dintre ele va putea fi selectat la un moment dat restul fiind automat deselectate.

```
...
<INPUT TYPE="RADIO" NAME="RADIO1" CHECKED>
...
<INPUT TYPE="RADIO" NAME="RADIO1">
...
```

Browserul va afişa: 

Câmpurile CHECKBOX folosesc următoarele atribute:

- **CHECKED** dacă acest atribut este prezent atunci acest câmp va fi selectat implicit la încărcarea pagii.
- **NAME** specifică numele variabilei care va fi transmisă la CGI. Se foloseşte aceeaşi valoare pentru butoanele din acelaşi grup.

Butonul TRIMITE

BUTTON SUBMIT Creează un buton care odată apăsat de către utilizator va declanşa trimiterea informaţiei din formular la server pentru a fi prelucrată

```
<INPUT TYPE="SUBMIT" VALUE="Trimite">
```

Browserul va afişa: 

Butonul SUBMIT foloseşte următoarele atribute:

- **VALUE** Defineşte textul care va fi afişat pe buton
- **NAME** specifică numele variabilei care va fi transmisă la CGI.

Butonul ANULEAZĂ

BUTTON RESET Creează un buton care odată apăsat de către utilizator va sterge informaţia introdusă până la momentul respectiv în formular de către utilizator. Nici un fel de informaţie nu va fi trimisă la server.

După apăsarea acestui buton formularul va fi afişat ca şi cum ar fi proaspăt reîncărcat. Utilizatorul poate reîncepe introducerea de date în formular imediat după apăsarea acestuia.

```
<INPUT TYPE="SUBMIT" VALUE="Anulează">
```

Browserul va afişa: 

Butonul RESET foloseşte următorul atribut:

- **VALUE** pentru a defini textul care va fi afişat în browser pe acest buton

8.4 Alte tipuri de elemente folosite în formulare WEB

Liste de selecţie

Câmpurile de tip RADIOBUTTON şi CHECKBOX oferă posibilitatae utilizatorului să aleagă o anumită opţiune dintr-un set. Au însă un mic dezavantaj: În cazul în care există multe opţiuni posibile spaţiul fizic necesar pentru prezentarea tuturor acestor opţiuni este foarte mare.

Pentru aceste cazuri există elementele de tip listă de selecţie care se creează în cadrul formularului cu ajutorul directivelor SELECT şi OPTION.

<SELECT>	<OPTION>
Funcţionalitate: • Defineneşte o listă de selecţie într-un formular **Atribute:** • NAME • SIZE • MULTIPLE **Directiva de sfârşit:** • </SELECT> Obligatorie	**Funcţionalitate:** • Defineneşte o opţiune într-o listă de selecţie **Atribute:** • SELECTED • VALUE **Directiva de sfârşit:** • </OPTION> Obligatorie

Există două tipuri de liste de selecţie:

1. Dropdown List – care iniţial afişează un singur element pe un singur rând iar în momentul în care utilizatorul selectează această listă se va extinde afişând toate opţiunile disponibile. Dacă valoarea atributului SIZE este 1 atunci lista de opţiuni va fi afişată în această formă.

Neextinsă: 

şi extinsă: 

Figura 8. 4 Exemplu listă selecţie de tip Dropdown

2. List Box – Elementele listei sunt afişate într-o zonă rectangulară unele sub altele. Dimensiunea vizibilă a acestei zone se defineşte cu ajutorul atributului SIZE. Lista de selecţie va fi afişată în această formă dacă valoarea atributului SIZE este diferită de 1

Figura 8.5 Exemplu listă de selecţie de tip Listbox

Paşii pentru realizarea unei liste de selecţie:

1. Introducerea directivei pereche <SELECT> </SELECT>
2. Stabilirea cu ajutorul atributului SIZE a tipului de listă de selecţie
   i. SIZE =”1” – Dropdown List
   ii. SIZE > 1 – ListBox
3. Adăugarea elementelor la listă cu ajutorul directivei <OPTION>. Un element din listă este introdus de o directivă <OPTION> ca în exemplul de mai jos:

```
<SELECT SIZE="1" MULTIPLE>
 <OPTION>Margareta</OPTION>
 <OPTION SELECTED>Trandafir</OPTION>
 <OPTION >Garoafa</OPTION>
...
</SELECT>
```

Atributul MULTIPLE

Atributul **MULTIPLE** al directivei <SELECT> setează modul de selecţie multiplă permiţând selectarea mai multor elemente din listă odată. In mod implicit nu se poate selecta decât un singur element din listă la un moment dat.

Atributul SELECTED

Atributul **SELECTED** al directivei <OPTION> defineşte care din elementele listei este setat implicit la încărcarea paginii.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 8.1.
La apăsarea butonului Trimite (submit) datele vor fi trimise spre a fi prelucrate în funcţie de modul în care a fost specificat, fie către un server fie către un script care rulează în browser. Pentru nelămuriri revedeţi secţiunea 8.1.

Întrebarea 8.2.
Există două metode de transmitere a datelor dintr-un formular către server: GET şi POST. Cea mai nesigurăte metoda este GET deoarece aceasta trimite datele din formular ca parte din URL care poate fi vizualizat de către utilizatori neautorizaţi. Consultaţi secţiunea 8.2.

Indicaţii la problemele propuse

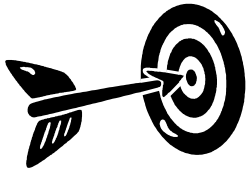
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dave Taylor - Crearea paginilor WEB, Editura Teora 1999, pg.253-277
2. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.38-46
3. T.Gugoiu – HTML prin exemple, Editura Teora 2000, pg.82-95

Unitatea de învăţare Nr. 9
SCRIPT, JAVASCRIPT…

Obiectivele Unităţii de învăţare Nr. 9:



După parcurgerea acestei Unităţii de învăţare vei stii:

- Ce este un script pe partea de client
- Ce este JavaScript
- Ce este un atribut Event Handler
- Să foloseşti un scrip într-o pagină WEB

9.1 Ce este JavaScript?

JavaScript a fost inventat pentru a adăuga paginilor WEB capabilitatea de a prelucra informaţii introduse de utilizator sau de a executa operaţii fără a fi nevoie de intervenţia sau ajutorul serverului de WEB.
Iată câteva exemple practice de întrebuinţare a JavaScript într-o pagină WEB:

- Animaţii
- Prelucrarea unor informaţii introduse de utilizator fără a fi nevoie de trimiterea datelor la server pentru a fi prelucrate de un CGI
- Realizarea de meniuri dinamice în pagina WEB.

Spre deosebire de limbajul Java care este scris exclusiv pentru programatori, JavaScript este un limbaj simplu scris cu scopul de a fi uşor de asimilat şi de folosit de către persoane care nu au experienţă anterioară în programare.

9.2 Inserarea unui JavaScript într-un document HTML

O aplicaţie JavaScript poate fi adăugată într-un document HTML utilizând directiva pereche <SCRIPT>. </SCRIPT>. Se pot include oricâte directive <SCRIPT> într-o pagină WEB în oricare din secţiunile documentului <HEAD> sau <BODY>.
Singura restricţie ar fi că în interiorul acestei directive nu se pot introduce alte directive HTML. Introducerea de directive HTML aici va fi semnalată ca eroare de către browser în momentul în care va afişa pagina.

```
<HEAD>
...
  <SCRIPT LANGUAGE=”JavaScript”>

Aici intră scriptul tău

</SCRIPT>
...
</HEAD>
<BODY>
...
  <SCRIPT LANGUAGE=”JavaScript”>
     Aici intră scriptul tău
  </SCRIPT>
...
</BODY>
```

Exemplul 9.1 Inserare script în document HTML

Browserele care nu suportă JavaScript vor trata scriptul din interiorul acestei directive ca pe un text normal şi prin urmare îl va afişa în pagină. Acest lucru nu este de dorit adesea. Pentru a nu ajunge într-o astfel de situaţie se recomandă ca scriptul din interiorul acestei directive să fie inclus într-un comentariu HTML, ca în exemplul de mai jos:

```
  <SCRIPT LANGUAGE=”JavaScript”>
    <!
      Aici intră scriptul tău
    -->
  </SCRIPT>
```

<SCRIPT>

Funcţionalitate:

- Inserează un script în documentul HTML

Atribute:

- LANGUAGE
- SRC
- TYPE

Directiva de sfârşit:

- </SCRIPT> OBLIGATORIE

Atributele LANGUAGE Si TYPE

Atributele **LANGUAGE** şi **TYPE** JavaScript sunt unele dintre cele mai populare limbaje de script folosite în WEB, însă există si altele precum VBScript. Rolul acestor atribute este de a specifica browserului ce tip şi ce versiune de script este inclus în interiorul directivei.

Este de ajuns folosirea fie a atributului LANGUAGE, fie a atributului TYPE. NU este nevoie a fi folosite ambele în acelaşi timp.

Cele mai utilizate valori pentru atributul LANGUAGE sunt “JavaScript” şi “VBScript”. Acelaşi lucru se poate specifica utilizând valoarele “text/javascript” respectiv text/vbscript” pentru atributul TYPE.

Atributul SRC

Atributul **SRC**. Pentru cazurile în care un anumit script are dimensiuni foarte mari sau este folosit de către mai multe pagini WEB acesta poate fi scris într-un fişier separat. Includerea acestor scripturi pentru a putea fi referite în pagina HTML curentă se face folosind atributul **SRC**.

Valoarea atributului este URL-ul la care se găseşte fişierul ce conţine scriptul. Prin urmare pentru a include un script definit în alt fişier vom scrie:

```
<SCRIPT LANGUAGE=”JavaScript”
        SRC=”scripts/scriptulmeu.js”>
```

Test de autoevaluare

9.1 De ce este recomandat plasarea codului unui script în interiorul unui comentariu HTML?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 102.

9.3 Cum şi când se execută un script într-o pagină WEB

Scripturile se pot executa fie la încărcarea paginii, dacă scriptul este definit în secţiunea <BODY> fie la apariţia unui eveniment care are asociat un script.

Una din cele mai importante facilităţi oferite de JavaScript este posibilitatea de a detecta anumite evenimente care au loc în pagină şi de a reacţiona la acestea.
Exemple de astfel de evenimente pot fi: trecerea cu cursorul mouse-ului peste un anumit obiect (hyperlink, imagine, buton etc) , încărcarea paginii, descărcarea paginii, apăsarea unei taste etc.

Majoritatea directivelor pot fi configurate să răspundă la astfel de evenimente prin executarea unui script JavaScript la producerea acestuia.

Hai sa vedem ce scriem pentru ca o imagine să declanşeze execuţia unui script JavaScript atunci când utilizatorul “trece” cu cursorul mouse-ului pe deasupra imaginii:

```
<HEAD>
  <SCRIPT LANGUAGE=”JavaScript”>
    function mesajIepuras()
     {
      alert(“Salut! Sunt iepurasul pinki!”)
     }
  </SCRIPT>
</HEAD>
<BODY>
...
<IMG SRC="iepuras.gif"
     ALT="Bugs Bunny"
     onMouseOver="mesajIepuras();">
...
```

în browser la încărcarea paginii va fi încărcată imaginea iar în momentul în care cursorul mouse-ului va trece peste imagine va fi afişată o fereastră cu mesaj ca în figura de mai jos:

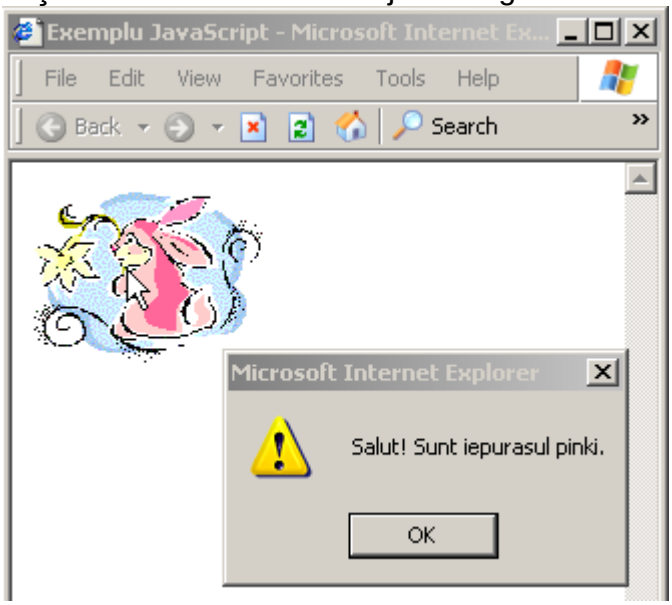


Figura 9.1 Exemplu declanşare execuţie script

De ce a apărut mesajul? Deoarece la definirea directivei IMG mai sus am adăugat atributul ***onMouseOver*** care are ca valoare numele unei funcţii JavaScript.

Asemănător pentru fiecare tip de eveniment există un atribut asociat. Dacă unul din aceste atribute este definit în cadrul directivei HTML, iar evenimentul asociat are loc la un anumit moment dat atunci se va executa funcţia JavaScript definită de valoarea atributului.

Atribute Event Handler

Acest tip de atribute se numesc ***Atribute Event Handler.*** În limba română aceasta se poate traduce prin "atribute pentru procesarea evenimentelor". Pentru majoriatea directivelor HTML se pot defini astfel de atribute asociind astfel un script care să fie executat la apariţia evenimentului. Există doar câteva execepţii de directive care nu suportă astfel de atribute, acestea sunt: <HEAD>, <BODY>, <BASEFONT>, <FONT>,
, <HTML>, <SCRIPT>, <TITLE>, <APPLET>, <SCRIPT> şi <FRAME>

Execuţia scriptului declanşat poate consta în simpla afişare a unui mesaj ca mai sus sau poate fi o prelucrare complexă de date introduse de către utilizator.

Test de autoevaluare

9.2 Când se execută un script JavaScript într-o pagină WEB?

a> ☐ La incarcarea paginii
b> ☐ La aparitia unui eveniment
c> ☐ la cererea vizitatorului paginii WEB
d> ☐ la o anumita ora stabilita de catre vizitatorul paginii WEB

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 102.

9.4 Atribute de tip Event Handler

Mai jos vom enumera cele mai utilizate atribute Event Handler suportate de către directivele HTML, însoţite de o scurtă explicaţie pentru fiecare în parte. Pentru o listă completă a acestor atribute consultă una din lucrările din bibliografia acestei Unităţii de învăţare.

Unele evenimente apar mai rar şi nu pot fi asociate decât anumitor directive. Pentru acestea vom enumera directivele care le suporată.

Directive suportate

- **onClick** Evenimentul apare atunci când utilizatorul execută un click de mouse pe elementul respectiv. Este suportat de majoritatea directivelor.
- **onDblClick** Evenimentul apare atunci când utilizatorul execută un dublu click de mouse pe elementul respectiv. Este suportat de majoritatea directivelor.
- **onMouseOver** Evenimentul apare atunci când utilizatorul trece cu cursorul mouseului pe deasupra elementului. Este suportat de majoritatea directivelor.
- **onSubmit** Evenimentul apare atunci când există un formular în pagină în momentul în care utilizatorul apasă un buton de tip "**submit**" (Trimite). Directivele care suportă acest atribut sunt <FORM> şi <BODY>.

- **onReset** Evenimentul apare atunci când există un formular în pagină în momentul în care utilizatorul apasă un buton de tip "**reset**" (Anulează). Directivele care suportă acest atribut sunt <FORM> şi <INPUT>.
- **onKeyPress** Evenimentul apare atunci când utilizatorul apasă o tastă. Este suportat de majoritatea directivelor.

Test de autoevaluare

9.3 Ce rol au atributele Event Handler?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 102.

Lucrare de verificare a Unităţilor de învăţare Nr. 7, 8 şi 9

1. Să se realizeze un tabel HTML cu următoarele caracteristici:

1. tabelul să fie centrat şi să ocupe 50% din lungimea ferestrei browserului
2. Să aibă 4 rânduri
3. Să aibă 3 colane
4. Datele pe coloana 1 să fie aliniate la stânga
5. Datele pe coloana 2 să fie centrate
6. Datele pe coloana 3 să fie aliniate la dreapta

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 2 şi 3 din bibliografia unităţii de învăţare nr.7.

Nr. de puncte **9** :
1 – 3p, 2 – 1p, 3 – 1p, 4 – 1p, 5 – 1p, 6 – 1p

2. Realizaţi un formular HTML care să poată fi folosit la un sondaj de opinie (alege singur subiectul). Acesta trebuie să conţină câmpuri textbox, butoane radio şi liste de selecţie

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 2 şi 3 din bibliografia unităţii de învăţare nr.8.

Nr. de puncte **8** (2p – utilizarea directivei FORM, 2p -utilizarea câmpuri de tip textbox, 2p – utilizarea câmpuri de tip buton radio; 2p - utilizarea câmpurilor de tip listă de selecţie)

3. Realizaţi o pagină HTML care să conţină o directivă HTML <A> care să aibă următorul comportament: când utilizatorul execută dublu-click pe acest hyperlink să se afişeze mesajul “De doua ori click!” într-o fereastră separată (o fereastră de tip dialog). Pentru aceasta se va utiliza JavaScript şi atribute de tip Event Handler.

Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Pe lângă informaţiile prezentate în manual se poate utiliza reperul bibliografic numărul 1 şi 2 din bibliografia unităţii de învăţare.

Nr. de puncte **8** (4p – pentru definirea corectă a atributului EventHandler pentru directiva <A>, 4p – definirea şi folosirea secvenţei JavaScript pentru afişarea mesajului)

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 9.1.
Este recomandat definirea scripturilor în interiorul comentariilor HTML pentru cazurile în care browserul cu care este vizualizată pagina nu suportă JavaScript. În aceste cazuri browserul va afişa textul scriptului în loc să il execute. Pentru a preveni afişarea scriptului în pagină acesta trebuie introdus în interiorul unui comentariu HTML. A se revedea secţiunea 9.2.

Întrebarea 9.2.
Un scrip Java Script se execută la încărcarea paginii dacă acesta este definit în interiorul directivei<BODY> sau la apariţia unui anumit eveniment. Variante corecte de raspuns: a) şi b). A se revedea secţiunea 9.3.

Întrebarea 9.3.
Atributele Event Handler au rolul de a defini acţiunea (scriptul) care se va executa în momentul apariţiei unui eveniment. Consultaţi secţiunea 9.4.

Indicaţii la problemele propuse

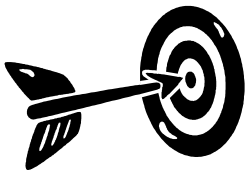
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Richard Wagner, R.Allen Wyke – Java Script, Editura Teora 2000, pg.47-59, 64-76
2. Mihaela Brut, Sabin Buraga - Prezentari multimedia pe Web. Limbajele XHTML + TIME şi SMIL Editura Polirom 2003, pg. 13-22, 30-33
3. Călin Ioan Acu – Optimizarea paginilor Web, Editura Polirom 2003, pg.182-267

Unitatea de învăţare Nr. 10
EXECUTABILE ŞI MULTIMEDIA ÎN PAGINA WEB

Obiectivele Unităţii de învăţare Nr.10:



După parcurgerea acestei Unităţii de învăţare vei stii:

- Ce este un Applet Java
- Cum se foloseşte un applet Java într-o pagină WEB
- Ce este un obiect Active X
- Cum se foloseşte un obiect Active X într-o pagină WEB
- Ce sunt şi cum se pot folosi fişierele multimedia într-o pagină Web

10.1 Java şi Java APPLET

Java

Java este un limbaj de programare dezvoltat de compania “Sun Microsystems”. Este un limbaj care oferă întreg suportul pentru programarea orietată pe obiecte şi a fost creat cu scopul de putea fi utilizat fără nici un fel de diferenţă pe orice sistem care suportă Java (*cross-platform*). Un alt mare avantaj al limbajului Java este faptul că programele scrise în Java şi compilate pentru un sistem de operare vor rula fără a fi recompilate pe orice alt sistem de operare care suportă Java. Acest lucru este posibil deoarece programele Java nu rulează folosind direct resursele oferite de sistemul de operare ci rulează într-un alt program numit Java Virtual Machine care se traduce prin Maşina Virtuală Java.

Avantaj

În literatura de specialitate se face referire la ea prin acronimul său - JVM.

După cum îi spune şi numele acest program creează un mediu virtual în care un program Java poate rula.

APPLET Java

Programele Java pot fi folosite într-o pagină Web sub formă de APPLET Java. Un applet Java este un program executabil într-un JVM. Aceste programe se găsesc în fişiere cu extensia *.class*, şi pot fi incluse direct în pagina Web folosind directiva HTML <APPLET> împreună cu directiva <PARAM> după cum vom vedea mai jos.

Browserele care suportă Java crează un astfel de mediu pe care programele Java de tip APPLET pot să îl folosească pentru a rula.

Dar atenţie nu toate browserele suportă Java! Şi deci la realizarea unei pagini de Web trebuie ţinut cont de acest lucru.

<APPLET>	<PARAM>
Funcţionalitate: • Introduce un Applet Java în pagină **Atribute:** • CODE • CODEBASE • ALIGN • WIDTH • HEIGHT **Directiva de sfârşit:** </APPLET> OBLIGATORIE	**Funcţionalitate:** • Permite setarea unui parametru de intrare pentru un applet **Atribute:** • NAME • VALUE **Directiva de sfârşit:** </PARAM> OPTIONALÂ

Paşi folosiţi

Paşii pentru a introduce un applet Java într-o pagină Web sunt următorii:

Pasul 1: Fişierul *.class* ce conţine appletul Java va trebui copiat în directorul care se află fişierul HTML, sau în directorul specificat de către atributul CODEBASE.

Pasul 2: Appletul Java aflat în fişierul .class va fi introdus în pagina Web folosind directiva HTML <APPLET>. Exemplul de mai jos ilustrează modul în care se poate introduce într-o pagină HTML un simplu applet Java fără parametrii de intrare:

```
<APPLET CODE=”appletulmeu.class”
        WIDTH=”300”
        HEIGHT=”200”>
</APPLET>
```

Dacă appletul are nevoie de anumiţi parametrii de intrare pentru a rula aceştia pot fi specificaţi cu ajutorul unui set de directive <PARAM> în interiorul elementului <APPLET> ca în exemplul de mai jos unde apletul primeşte ca parametrii de intrare un nume şi o parolă:

```
<APPLET CODE=”appletulmeu.class”
        WIDTH=”300”
        HEIGHT=”200”>
   <PARAM NAME=”nume” VALUE=”ana”>
   <PARAM NAME=”parola” VALUE=”bu12cc”>
</APPLET>
```

Hai să vedem ce semnificaţie are şi unde poate fi folosit fiecare atribut în parte:

Atributul CODE

Atributul **CODE** – specifică numele fişierului *.class* sau *.jar* care conţine codul executabil al applet-ului Java folosit. Acest parametru este obligatoriu.

Atributul CODEBASE

Atributul **CODEBASE** – specifică locaţia unde poate fi găsit fişierul specificat de atributul CODE. Dacă acest atribut lipseşte fişierul .class va fi căutat în directorul unde se află documentul HTML.

Atributele WIDTH si HEIGHT

Atributele **WIDTH** şi **HEIGHT** – specifică dimensiunea pe orizontală respectiv pe verticală a spaţiului rezervat de browser pentru afişarea appletului. Este recomandată specificarea acestor atribute pentru ca appletul să afişeze datele corect.

Pentru directiva <PARAM> atribututul **NAME** este folosit pentru a specifica numele, iar atributul **VALUE** pentru a specifica valoarea parametrului de intrare care va fi transmis appletului.

Appleturile Java extind posibilităţile oferite de HTML şi/sau browser permiţând realizarea unor pagini puternic interactive, cu interfaţă grafică ce nu ar putea fi în mod normal creeată cu facilităţile puse la dispoziţie de către limbajul HTML.

Test de autoevaluare

10.1 Un applet Java rulează pe calculatorul client sau rulează pe server şi prezintă doar rezultatul în fereastra browserului?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 110.

10.2 Obiecte ACTIVE X

Active X este o tehnologie dezvoltată de către compania Microsoft care permite adăugarea unor componente executabile în pagina WEB. Obiectele sau controalele Active X sunt asemănătoare cu un applet Java. Spre deosebire de acestea, însă, pot avea un control mult mai puţin restricţionat asupra maşinii pe care rulează, ceea ce poate fi perceput ca o problemă de securitate de către anumiţi utilizatori.

Avantaj

Marele avantaj al obiectelor Active X faţă de appleturile Java este că sunt transferate de la server o singură dată. După ce au fost transferate ele devin parte integrantă din sistemul de operare. Execuţiile ulterioare ale obiectului se vor face astfel rapid. Datorită acestui avantaj şi al altor caracteristici, Active X este folosit în special pentru aplicaţiile grafice livrate către utilizator prin intermediul browserului.

Dezavantaj

Marele dezavantaj al obiectelor Active X este că acestea nu pot rula decât pe un calculator care rulează sistemul de operare Windows.

Controalele Active X pot fi dezvoltate folosind diferite limbaje de programare, cum ar fi Microsoft Visual Basic, Microsoft Visual C++, şi chiar Java.

Un control Active X poate fi adăugat într-o pagină Web folosind directiva HTML <OBJECT> în mod similar cu directiva APPLET pentru appleturi Java. Ca şi în cazul elementului APPLET parametrii de intrare pot fi specificaţi utilizând directiva <PARAM>.

10.3 Fişiere multimedia în pagina WEB

Fişierele multimedia sunt fişiere care conţin sunete, secvenţe muzicale, sau chiar clipuri video. Acestea sunt din ce în ce mai folosite în paginile Web.

În momentul în care browserul detectează un fişier multimedia fie va lansa un program extern fie va folosi o facilitate internă (plug-in) pentru a vă rula clipul multimedia aflat în fisierul respectiv. Ce program extern va folosi pentru a rula clipul multimedia depinde de mai mulţi factori, precum tipul sistemului de operare sau tipul şi versiunea browserului. De aceea în calitate de creator al unei pagini web nu poţi controla direct programul în care va rula clipul multimedia pe care îl oferi în pagină.

Metode

Există două metode de a livra conţinutul fişierelor multimedia către browserul unui utilizator:

- Metoda statică sau ***non-streaming*** caz în care fişierul multimedia trebuie să fie complet descărcat pe calculatorul utilizatorului înainte ca acesta să fie rulat.
- Metoda dinamică sau ***streaming*** caz în care clipul conţinut de un fişier este rulat aproape imediat ce a fost referit transferul restului de fişier făcându-se pe măsură ce clipul este rulat.

Avantaj

Avantajul primei metode este că nu necesită existenţa nici unui program pe partea de server care să facă posibilă rularea clipului.

Dezavantaj

Principalul dezavantaj al acestei meteode este că în cazul în care dimensiunea fişierelor este mare, durata de transfer creşte foarte mult şi deci si timpul până când clipul poate fi rulat. De asemenea datorită faptului că fişierul ajunge direct pe calculatorul utilizatorului face mai dificil pentru creatorii acelor clipuri să îşi protejeze drepturile de autor asupra unor lucrări.

Streaming

Pentru a înlătura aceste dezavantaje majore a fost dezvoltată metoda dinamică de transfer a fişierelor multimedia referită în

literatura de specialitate cu termenul de "***streaming***". Această tehnologie presupune existenţa următoarelor componenete software:

- *Streaming server* este o componentă care gestionează cererile de streaming pentru diferite formate multimedia de la mai mulţi utilizatori simultan, folosind în mod eficient resursele calculatorului server.
- *Encoder* – Este o componentă software care converteşte un fişier dintr-un anumit format multimedia într-un format potrivit pentru streaming.
- *Player* - este o aplicaţie software care rulează pe maşina utilizatorului şi care are rolul de a reliza la cerere conexiunea cu serverul de streaming şi de a rula clipul servit de acesta.

Avantaje si dezavantaje

Avantajele acestei metode sunt clare după cum am descris mai sus deoarece utilizatorul nu trebuie să aştepte transferul complet al clipului multimedia înainte de a putea să-l vizioneze. Dezavantajele însă se leagă de faptul că această metodă necesită existenţa unui server de streaming care este de obicei scump şi destul de dificil de configurat şi menţinut.

10.4 Adăugarea clipurilor multimedia la o pagină Web

Pentru a include un clip multimedia într-o pagină Web se poate folosi fie directiva <A> fie directiva <EMBED>. Fişierele multimedia audio pot fi de asemenea rulate ca fundal sonor la o pagină Web cu ajutorul unor comenzi JavaScript.

Prin urmare poţi folosi o directivă simplă de tip ancoră ca în exemplul de mai jos:

```
<A HREF=”audio/clipulmeu.avi”>
No.1 clip (1.3Mo)
</A>
```

Atunci când utilizatorul selectează hiperlinkul de mai sus browserul va transfera pe hardiskul local clipul respectiv şi îl va rula cu ajutorul unei aplicaţii externe sau folosind facilităţile oferite de către browser în acest sens.

Directiva EMBED

Folosirea directivei <EMBED> oferă mai multe opţiuni pentru rularea clipurilor multimedia decât în cazul folosirii unui simplu hiperlink. Browserul va afişa în pagină în locul acestei directive o interfaţă grafică ce va permite utilizatorului să controleze derularea clipului – să-l oprească sau să-l repornească, să deruleze înainte şi înapoi etc.

Cu ajutorul atributelor directivei <EMBED> se poate configura ca clipul să fie rulat automat, ori să poată fi rulat în buclă sau doar o singură dată. Iată un exemplu de folosire a acestei directive:

```
<EMBED
 SRC=”http://www.siteulmeu.ro/clip1.wav”>
```

Singurul atribut obligatoriu pentru această directivă este **SRC** Valoarea lui reprezintă URL-ul unde se află clipul multimedia care se doreşte a fi rulat în pagină.
Atributele **WIDTH** şi **HEIGHT** permit definirea dimensiunilor interfeţei grafice de control afişate în browser.

Atributul AUTOSTART

Atributul **AUTOSTART** poate lua una din valorile TRUE sau FALSE şi specifică browserului să ruleze la încărcarea paginii clipul specificat de atributul SRC.

Atributul LOOP

Atributul **LOOP** poate lua una din valorile TRUE sau FALSE şi specifică browserului să ruleze sau nu în buclă clipul specificat de atributul SRC.

Atributul HIDDEN

Atributul **HIDDEN** poate lua una din valorile TRUE sau FALSE şi specifică browserului să afişeze sau să nu afişeze interfaţa grafică de control în pagina Web.

Test de autoevaluare

10.2 Realizaţi o pagină web care să includă clipul de la URL-ul http://www.multamuzica.ro/melo1.wav care să fie rulată automat la încărcarea paginii şi care să fie cântată în buclă

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 110.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 10.1.
Un applet Java rulează întotdeauna direct pe maşina client după ce a fost transferat de la server. Acesta fiind unul din avantajele acestei tehnologii pentru că în acest fel serverul este eliberat de anumite sarcini. În acelaşi timp, însă poate fi şi un dezavantaj pentru cazurile în care maşina client nu este destul de puternică. Consultaţi secţiunea 10.1.

Întrebarea 10.2.
Codul HTML corespunzător pentru a include acest clip este:

```
<EMBED
 SRC="http://www.multamuzica.ro/melo1.wav"
 LOOP="TRUE"
 AUTOSTART="TRUE"
>
```

Revedeţi indicaţiile din secţiunea 10.4.

Indicaţii la problemele propuse

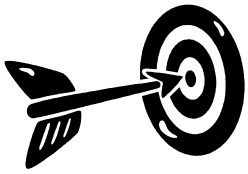
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Cioata Mihai – ActiveX. Concepte şi aplicatii Editura Polirom, pg.95-102, 174-175
2. Călin Ioan Acu – Optimizarea paginilor Web, Editura Polirom 2003, pg.95-99, 144-156
3. Mihaela Brut, Sabin Buraga - Prezentari multimedia pe Web. Limbajele XHTML + TIME şi SMIL Editura Polirom 2003, pg. 13-22 şi 233-247

Unitatea de învăţare Nr. 11
LIMBAJUL PHP ȘI FOLOSIREA LUI ÎN PAGINA WEB.

Obiectivele Unităţii de învăţare Nr.11:



Principalele obiective ale Unităţii de învăţare Nr. 11 sunt:

- Ce este limbajul PHP şi la ce poate fi folosit.
- Cum poate fi integrat un script într-un document HTML.
- Care sunt principalele tehnici şi metode de programare în cadrul limbajului PHP.

11.1 Introducere în limbajul PHP

PHP (Hypertext Preprocessor) este un limbaj de scripting de uz general, utilizat pe scară largă, potrivit pentru dezvoltarea aplicaţiilor Web şi care poate fi integrat în paginile HTML. Sintaxa sa este asemănătoare cu a limbajelor de programare C, Java sau Perl fiind uşor de învăţat. Scopul principal al său este de a permite programatorilor web să genereze rapid pagini web cu un comportament dinamic.
Limbajul PHP diferă de limbajul de scripting JavaScript prin faptul că este rulat pe partea de server, generând HTML care este apoi trimis către client. Clientul va primi rezultatele rulării acelui script, fără a putea cunoaşte codul sursă iniţial. Având în vedere rularea pe partea de server, limbajul PHP permite lucruri similare cu aplicaţiile de tip CGI cum ar fi realizarea de formulare de colectare de date, generarea dinamică a conţinutului de pagini web, trimiterea sau analiza obiectelor de tip „cookies”.

Pentru a putea vizualiza paginile web cu conţinut PHP trebuie să dispuneţi de un server web cu extensia PHP instalată. PHP poate fi utilizat pe toate sistemele de operare majore (Linux, Unix, MacOS sau Microsoft Windows) fiind suportat de majoritatea serverelor web (apache, Microsoft IIS, lighthttpd sau nginx). Instalarea în cadrul serverului web se poate face ca modul sau ca procesor CGI.

În cadrul utilizării PHP nu sunteţi limitaţi doar la afişarea HTML ci există posibilitatea de a afişa (ca răspuns al rulării scriptului PHP) imagini, fişiere PDF sau chiar animaţii Flash generate în timp real. O altă facilitate foarte importantă a limbajului PHP este posibilitatea de interconectare cu un server de baze de date, utilizând conţinutul preluat în generarea paginii web trimisă către client (acest aspect va fi detaliat în unitatea de învăţare următoare). Pe lângă posibilitatea de generare de pagini şi obiecte transmise prin intermediul protocolului HTTP de către serverul web, limbajul PHP permite interconectarea şi cu alte servicii de reţea (LDAP, IMAP, SNMP, NNTP, POP3, SMTP) prin intermediul librăriilor proprii specializate.

Limbajul PHP posedă facilităţi foarte puternice de procesare a textului şi multe extensii de accesare şi interpretare a limbajelor XML.

Exemplu de fişier PHP:

```
<html>
 <head>
  <title>Exemplu PHP</title>
 </head>
 <body>
 <?php echo '<p>Salut!</p>'; ?>
 </body>
</html>
```

În urma prelucrării de către server a scriptului PHP, clientul va primi următorul fişier HTML (comanda „echo” din limbajul PHP are efect trimiterea către client a şirului de caractere dintre ghilimele):

```
<html>
 <head>
  <title>Exemplu PHP</title>
 </head>
 <body>
  <p>Salut!</p>
 </body>
</html>
```

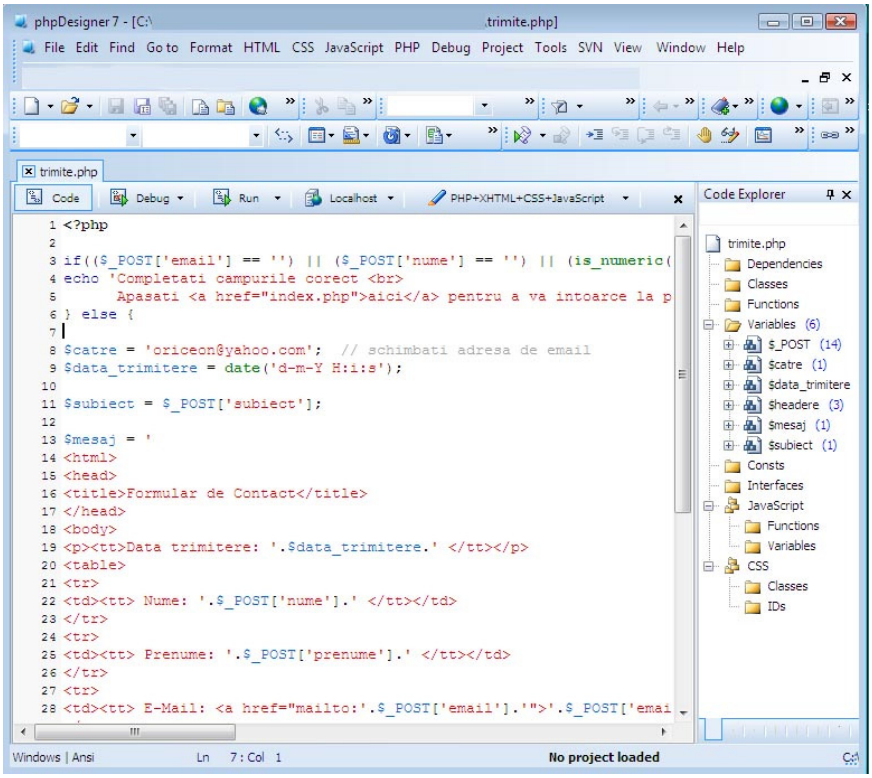


Figura 11.1 Realizarea unui script PHP utilizând phpDesigner 7

Cum testăm fișierele PHP?

La fel ca şi în cazul fişierelor HTML editarea fişierelor PHP se poate cu orice editor de text (Notepad sau Notepad++) sau cu un editor specializat (phpDesigner, PHP Editor, Zend Studio). Salvarea programelor PHP trebuie făcută în fişiere cu extensia .php . Pentru a putea evalua efectul rulării unei pagini web ce include scripturi php este necesară instalarea unui server web cu extensie php. Există posibilitatea instalării extensiei php sub Microsoft IIS sau putem utiliza o suită integrată ce cuprinde serverul web apache, extensia php şi serverul de baze de date MySQL (de exemplu WampServer – http://www.wampserver.com).

11.2 Includerea de cod PHP în documente HTML

Secvenţele de cod PHP se regăsesc integrate în documentele HTML între instrucţiuni de procesare de început şi sfârşit: <?php şi ?>.

```
<?php echo '<p>Salut!</p>'; ?>
```

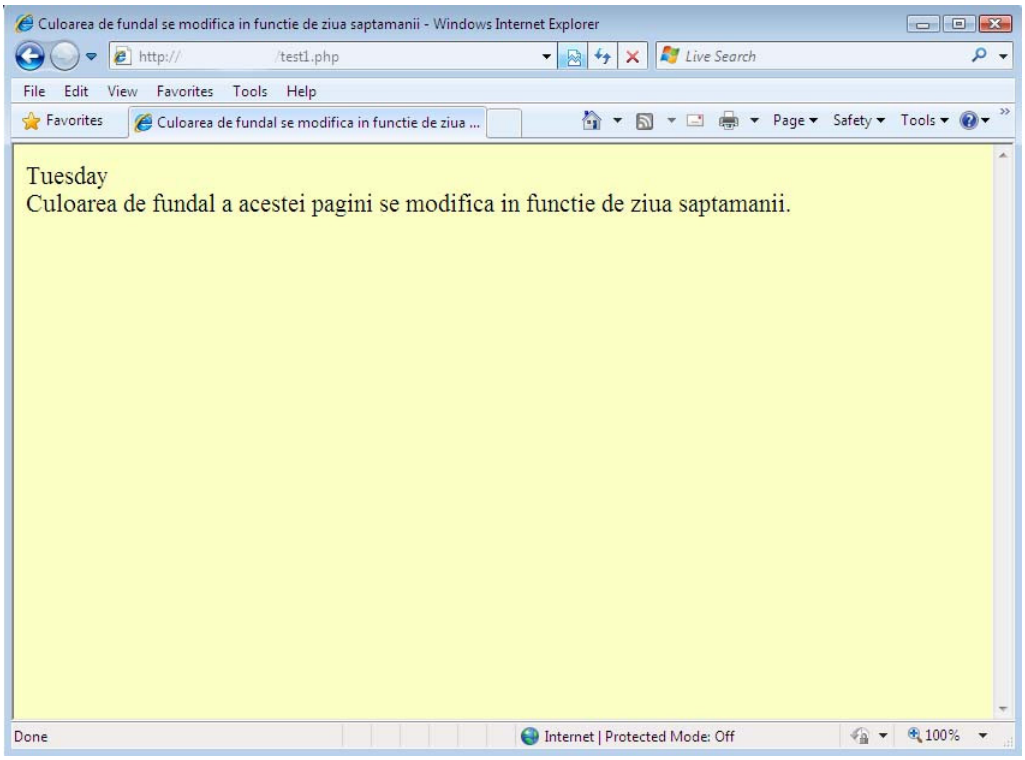
Cele două instrucţiuni au semnificaţia de intrare şi ieşire din modul PHP. Tot ce este în afara celor două instrucţiuni delimitatoare nu se interpretează la nivel de server (este cod HTML simplu), codul cuprins între cele două instrucţiuni va fi înlocuit de către server cu ieşirea generată de execuţia scriptului.

Un alt exemplu de cod sursă PHP care în funcţie de ziua săptămânii modifică culoare fundalului unei pagini HTML:

```
<html>
<head>
<title>Culoarea de fundal se modifica in functie de
                          ziua saptamanii</title>
</head>
<?php
    // extragem ziua săptămânii cu ajutorul
     funcţiei date
    $today = date("l");
    // afişăm ziua săptămânii
    print("$today");
    // verificăm dacă este duminică (sunday)
    if($today == "Sunday") { $bgcolor = "#FEF0C5"; }
    // verificăm dacă este luni (monday)
    elseif($today == "Monday")
                    { $bgcolor = "#FFFFFF";}
    // verificăm dacă este marţi (tuesday)
    elseif($today == "Tuesday")
                    { $bgcolor = "#FBFFC4"; }
    // verificăm dacă este miercuri (wednesday)
    elseif($today == "Wednesday")
                     { $bgcolor = "#FFE0DD"; }
    // verificăm dacă este joi (thursday)
```

```
    elseif($today == "Thursday")
                        { $bgcolor = "#E6EDFF"; }
    // verificăm dacă este vineri (friday)
    elseif($today == "Friday")
                        { $bgcolor = "#E9FFE6"; }
    // dacă nu este nici una din zilele de mai sus
     atunci este sâmbătă (Saturday)
    else { $bgcolor = "#F0F4F1"; }
    print("<body bgcolor=\"$bgcolor\">\n");
?>
<br>Culoarea de fundal a acestei pagini se modifica
in functie de ziua saptamanii.
</body>
</html>
```

Dacă vom accesa pagina de mai sus prin intermediul unui browser web de pe un server care are extensia PHP instalată vom obţine următorul rezultat:



În cadrul exemplului anterior putem observa că putem introduce un comentariu cu ajutorul `//` . Sunt valabile şi alte forme de delimitare a textelor explicative: `/* */` - varianta C clasică şi `#` - varianta de bash.

Pentru a înţelege modul de funcţionare a scriptului anterior trebuie să înţelegem modul de funcţionare a variabilelor (am utilizat variabila `$today` şi `$bgcolor`), a funcţiilor ce permit interacţiune cu sistemul de calcul (funcţia `date`) şi a instrucţiunilor program obişnuite (instrucţiunea `if – else`). Toate aceste aspecte vor fi detaliate în cele ce urmează.

11.3 Variabile, constante, operatori în limbajul PHP

Variabile în PHP

Variabilele în limbajul PHP nu necesită o declarare prealabilă sau stabilirea tipului precum în alte limbaje. O variabilă se consideră declarată şi devine utilizabilă printr-o simplă atribuire:

```
$var = 0;
$nume = ”George”;
```

Pentru a utiliza o variabilă, numele acesteia trebuie precedat de semnul $. Numele unei variabile este format din litere, cifre sau semnul _ şi nu poate începe cu o cifră.

Chiar dacă nu este necesar să indicăm tipul unei variabile în momentul primei utilizări asta nu înseamnă că datele stocate în variabilele utilizate nu au tip, doar că limbajul PHP stabileşte în mod automat tipul datelor utilizate de noi. Astfel, vorbim de următoarele tipuri în limbajul PHP:

- **Boolean** – variabila poate lua valorile True sau False.
- **Float** – număr în virgulă mobilă.
- **Integer** – număr întreg.
- **String** - şir de caractere.
- **Array** – matrice (listă de variabile).
- **Object** – instanţa unui obiect de tip clasă.
- **Resource** – pointer către o bază de date, poate fi conexiune sau set de date rezultat.

Tipurile sunt necesare pentru ca în cadrul scripturilor să putem efectua diverse operaţii asupra variabilelor (operaţii matematice cu numere, operaţii de căutare sau concatenare cu şirurile de caractere ş.a.m.d.).

Constante în PHP

Constantele reprezintă aliasuri pentru diverse valori; în loc să utilizăm o anume valoare putem utiliza o denumire pentru a uşura procesul de referire. Pentru a declara o constantă utilizăm cuvântul cheie `const` sau `define`:

```
const DENSITATE_APA = 1;
define(”VOLUM”,100);
$greutate = DENSITATE_APA*VOLUM;
```

După cum se poate observa referirea unei constate nu necesită utilizarea semnului $. Bineînţeles, valoarea unei constante nu poate fi modificată după declarare. Nu se pot construi matrice de constante.

Variabile și constante predefinite

Pe lângă variabilele şi constantele definite de programator, limbajul PHP pune la dispoziţie o serie de elemente predefinite. Constantele predefinite ţin mai mult de diverşi parametrii ce nu se modifică în timpul execuţie iar variabilele predefinite reprezintă o metodă pusă la

dispoziţie de mediul de programare pentru a afla diverşi parametrii ce se modifică în timpul execuţiei.

```
<html>
<head>
<title>Exemplu de utilizare pentru constante si
                variabile predefinite</title>
</head>
<body>
<?php
    /* se va afişa versiunea de limbaj PHP prin
       intermediul constantei sistem predefinite
       PHP_VERSION */
    echo "<br>Versiunea de PHP este ".PHP_VERSION;
    /* se va afişa versiunea sistemului de operare
       prin intermediul constantei sistem
       prededinite PHP_OS */
    echo "<br>Rulati sistemul de operare: ".PHP_OS;
    /* se va afişa adresa IP şi denumirea
       serverului prin intermediul setului de
       variabile sistem predefinite _SERVER */
    echo "<br>Adresa IP: ".$_SERVER['SERVER_ADDR'];
    echo "<br>Denumire: ".$_SERVER['SERVER_NAME'];
?>
</body>
</html>
```

Operatori

Precum cum se poate observa în cele două exemple precedente, atât cu variabilele cât şi cu constantele se pot efectua diverse operaţii. Primul exemplu foloseşte operatorul * pentru a efectua înmulţirea a două constante numerice şi operatorul = pentru a atribui rezultatul operaţiei unei variabile. Cel de-al doilea exemplu utilizează operatorul . pentru a concatena două şiruri de caractere. Pentru fiecare tip de date în parte există operatori pentru execuţia unor operaţii specifice:

Operatori aritmetici

Operator	Operaţie	Exemplu
-	Negaţie	-$a
+	Adunare	$a+$b
-	Scădere	$a-$b
*****	Înmulţire	$a*$b
/	Împărţire	$a/$b
%	Modulo	$a%$b

Operatori de atribuire compuşi

Operator	Operaţie	Exemplu
=	Atribuire	$a=$b

+=	Însumare	$a+=$b echivalent $a=$a+$b
-=	Diferenţă	$a-=$b echivalent $a=$a-$b
***=**	Multiplicare	$a*=$b echivalent $a=$a*$b
/=	Împărţire	$a/=$b echivalent $a=$a/$b
%=	Modul	$a%=%b echivalent $a=$a%$b
&=	Şi la nivel de bit	$a&=$b echivalent $a=$a&$b
|=	Sau la niv de bit	$a\|=$b echivalent $a=$a|$b
^=	Sau exclusiv	$a^=$b echivalent $a=$a^$b
<<=	Deplasare st	$a<<=$b echivalent $a=$a<<$b
>>=	Deplasare dr	$a>>=$b echivalent $a=$a>>$b

Operatori pentru şiruri de caractere

Operator	Operaţie	Exemplu
=	Atribuire	$a="Test"
.	Concatenare	$a="Te"."st"
.=	Atribuire cumulativă	$a.=” nr.11”

Exemple de lucru cu diverşi operatori:

```
<?php
     /* Operaţie de concatenare de şiruri de
          caractere */
     $nume = "Popescu";
     $prenume = "Ionel";
     $nume_complet = $nume." ".$prenume;
     echo $nume_complet;
     /* Rezolvare ecuatie grad 2  ax2+bx+c=0 */
     $a = 2;
     $b = -7;
     $c = 5;
     $delta = ($b*$b)-(4*$a*$c);
     if ($delta==0)
          {
          $x1=(-$b)/(2*$a);
          echo "<br>x1=x2=".$x1;
          }
     elseif ($delta>0)
          {
          $x1=(-$b-sqrt($delta))/(2*$a);
          $x2=(-$b+sqrt($delta))/(2*$a);
          echo "<br>x1=".$x1;
          echo "<br>x2=".$x2;
          }
     else
          {
          echo "<br>Nu exista solutii reale";
          }
?>
```

Test de autoevaluare

11.1 Scrieţi un script PHP care să calculeze aria unui cerc bazându-se pe o constantă PI şi o variabilă $raza.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 125.

11.4 Structuri de control şi funcţii PHP

Structuri de control

Structurile de control în limbajul PHP sunt reprezentate de instrucţiuni de bază care permit efectuare operaţiilor elementare la nivel de cod: instrucţiuni decizionale (if – elseif – else, switch), instrucţiuni repetitive (while, for, break, continue) sau instrucţiuni de includere (include, require). Fără aceste instrucţiuni de bază nu am putea construi programe în limbajul PHP. Chiar dacă, în exemplele precedente, am utilizat deja instrucţiunea if – elseif – else este cazul să detaliem modul de utilizare şi a celorlalte structuri de control puse la dispoziţie de limbajul PHP.

Este momentul să facem o mică paranteză pentru a lămuri două aspecte sintactice importante ale limbajul PHP (aspecte deja utilizate în exemplele de cod precedente).

1. Toate instrucţiunile din program trebuie să aibă ca delimitator final simbolul ;

2. Un grup de instrucţiuni poate fi delimitat de simbolurile { } obţinându-se astfel un bloc de instrucţiuni sau o macroinstrucţiune care poate fi indicată ca ramură de execuţie pentru o instrucţiune decizională.

Instrucţiunea **if – elseif – else** este o instrucţiune decizională care permite evaluarea unor condiţii multiple.

```
if (condiţie1)
    instrucţiune (sau macroinstrucţiune) care se
    execută dacă condiţie1 este îndeplinită (are
    valoare de adevăr – true)
elseif (conditie2)
    instrucţiune care se execută dacă conditie2
    este îndeplinită
else
```

```
    instrucţiune care se execută dacă nici una din
    condiţiile anterioare nu sunt îndeplinite
```

Ramurile elseif şi else nu sunt obligatorii – putem avea o instrucţiune if simplă, cu o singură condiţie. Pot exista oricâte ramuri de tip elseif dar o singură ramură else.

Instrucţiunea **switch** este o instrucţiune similară cu instrucţiunea if dar care permite declararea simplă a unui număr mare de ramuri decizionale.

```
switch ($variabilă) {
    case valoare1
         instrucţiune care se execută dacă
         $variabilă are valoarea valoare1
    case valoare2
         …idem…
    case valoare3
         …idem…
    ….
    }
```

Instrucţiunile **while** şi **do – while** permit efectuarea repetitivă a unui grup de instrucţiuni atâta timp cât este adevărată o anumită condiţie.

```
while (condiţie):
    grup de instrucţiuni;
endwhile;

do
    grup de instrucţiuni;
while (condiţie);
```

Instrucţiunea **for** permite realizarea unei operaţii în mod repetitiv, putând controla numărul de iteraţii.

```
for ($i=1; $i<=10; $i++):
    echo("<br>Ma aflu la iteratia numarul: ".$i);
endfor;
```

La prima execuţie a instrucţii variabila $i este iniţializată cu valoarea 1. La următoarea execuţia variabila $i este incrementată în mod automat şi se verifică dacă a atins valoarea maximă 10. Instrucţiunile cuprinse între instrucţiunea for şi instrucţiunea endfor se vor executa în exemplul anterior de 10 ori.

O instrucţiune similară este instrucţiunea **foreach** care în loc să parcurgă un interval de valori parcurge elementele unui vector sau a unei colecţii de valori.

```
$vector = array(1,2,5,10,55);
$i=1;
```

```
foreach ($vector as $element) {
    echo("<br>Elementul ".$i." este ".$element);
    $i++;
}
```

Două instrucţiuni utile în conjuncţie cu instrucţiunile de ciclare sunt **continue** şi **break**. Apariţia instrucţiunii **continue** conduce la trecerea la următorul pas al ciclului fără a executa restul de instrucţiunii din cadrul buclei iar apariţia instrucţiunii **break** conduce la oprirea procesului de ciclare indiferent de numărul de paşi rămaşi.

```
$vector = array(1,2,5,10,55);
$i=1;
foreach ($vector as $element) {
 if ($element==10):
  echo("<br>Elementul 10 se afla pe pozitia: ".$i);
  break;
 endif;
 $i++;
}
```

Funcţii

Pe lângă instrucţiunile de bază, descrise anterior, în cadrul limbajului PHP mai întâlnim şi funcţii care sunt extrem de utile în scrierea de aplicaţii web. La momentul actual limbajul PHP include peste 700 de funcţii predefinite. Exemple de tipuri de funcţii şi funcţii existente în limbajul PHP:

- Funcţii de lucru cu şiruri de caractere
  - strcmp – compară două şiruri de caractere
  - strlen – returnează dimensiunea unui şir de caractere
  - trim – elimină spaţiile
  - crc32 – calculează suma CRC pe 32 de biţi
- Funcţii de lucru cu vectori şi matrice
  - sort – sortează elementele
  - range – generează un vector pentru un interval
  - sizeof – returnează numărul de elemente
- Funcţii de lucru cu date calendaristice
  - date_sunrise – returnează ora răsăritului pentru o anume dată
  - getdate – returneaza data curenta
  - checkdate – verifica validitatea unei date
- Funcţii matematice
  - max – returnează maximul dintre două numere
  - log – logaritm
  - sqrt – radical
- Funcţii diverse
  - sleep – introduce o întârziere
  - die – stopează rularea scriptului curent
  - get_browser – returnează versiunea de browser

Pentru o listă completă şi detaliată a funcţiilor puteţi consulta manualul oficial al limbajului PHP. El poate fi accesat gratuit la adresa:

http://www.php.net/manual/ro/index.php (limba română)
http://www.php.net/manual/en/index.php (limba engleză)

În felul acesta puteţi să fiţi siguri că lista este completă şi actualizată cu ultimele funcţii implementate în limbajul PHP.

Pe lângă funcţiile deja existente în limbajul PHP utilizatorul poate defini propriile funcţii pe care le poate apela în mod similar cu funcţiile deja existente.

```
function patrat($x)
{
    $rezultat =$x*$x;
    return $rezultat;
}

$a = 5;
$a2 = patrat($a);
```

Test de autoevaluare

11.2 Scrieţi o funcţie proprie care să returneze elementul cel mai mare element dintr-un vector primit ca parametru.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 125.

11.5 Clase şi obiecte PHP

Ultimele generaţii de limbaje de programare introduc un nou concept referitor la modul în care poate fi structurat un program şi anume programare orientată obiect (OOP – Object Oriented Programming). Această modalitate de programare are o mare posibilitate de abstractizare a elementelor folosite în program şi din această cauză reutilizarea şi depanarea codului este mult mai uşoară.

Principiile de bază ale OOP sunt:

- **Abstractizarea** – este posibilitatea ca un program să ignore unele aspecte ale informaţiei pe care o manipulează existând posibilitatea să se concentreze asupra esenţialului. Fiecare

obiect are posibilitate să îşi poată modifica starea sau să comunice cu alte subiecte din sistem fără a dezvălui cum au fost implementate aceste facilităţi.

- **Încapsularea** – numită şi ascunderea de informaţii: permite ca obiectele nu pot modifica starea altor obiecte în mod direct ci doar prin metode puse la dispoziţie de obiectul respectiv.
- **Polimorfismul** – este abilitatea de a procesa obiectele în mod diferit, în funcţie de tipul sau de clasa lor. Altfel spus, este abilitatea de a redefini metode pentru clase derivate.
- **Moştenirea** – permite organizarea şi facilitează polimorfismul şi încapsularea, permiţând definirea şi crearea unor clase specializate plecând de la clase (generale) deja definite care pot avea acelaşi comportament fără a fi nevoie să redefinească diverse metode sau instrumente.

Prin aplicarea acestor concepte programele devin, din simple liste de instrucţiuni sau apeluri de proceduri şi funcţii, colecţii de obiecte – unităţi individuale de cod care interacţionează unele cu altele.

Facilităţile de programare POO au existat şi în versiunea 4 a limbajului PHP dar au fost îmbunătăţite semnificativ în versiunea curentă – 5.

Exemplu de utilizare a unui container de date încapsulat – a unui obiect:

```
<?php
     $elev->nume="Ionescu Gabriel";
     $elev->varsta=14;
     printf("Elevul %s are %d ani.", $elev->nume,
          $elev->varsta);
?>
```

Secvenţa de cod anterioară va afişa:

```
Elevul Ionescu Gabriel are 14 ani.
```

După cum se poate observa două variabile nume şi varsta au fost grupate în cadrul aceluiaşi obiect elev. Un **obiect** este un tip de date care permite gruparea datelor şi funcţiilor în cadrul aceleiaşi variabile. O **clasă** este un şablon reutilizabil din care pot genera oricâte obiecte similare (**instanţe** ale clasei). De exemplu pot defini o clasă de tip elev care să arate în felul următor:

```
<?php
     class elev
     {
      function __construct($nume, $varsta)
      {
       $this->nume = $nume;
       $this->varsta = $varsta;
```

```
        }
        public function afisare()
        {
         printf("Elevul %s are %d ani.", $this->nume,
            $this->varsta);
        }
       }
       $elev_VIIIA = new elev("Ionescu Gabriel",14);
       $elev_VIIIA->afisare();
?>
```

În acest caz *elev* identifică clasa definită pentru a putea fi instanţiată pentru mai multe obiecte similare. După cum se poate observa instanţierea se face utilizând cuvântul cheie *new* ce primeşte ca parametrii valorile iniţiale pentru componentele de date interne ale obiectului. Aceste valori pot fi modificate ulterior pe parcursul utilizării obiectului. În momentul instanţierii unei clase se apelează în mod automat funcţia internă *__construct* care efectuează iniţializarea structurii de date interne a obiectului în cauză – această funcţie poartă denumirea de constructor. Există posibilitatea ca la instanţiere să existe valori predefinite pentru valorile interne ale obiectului fără a fi nevoie să le transmitem în momentul declarării iniţiale. Valorile interne ale unui obiect se numesc şi proprietăţi ale obiectului.

După cum se poate observa am adăugat şi o funcţie internă clasei *elev* – funcţia afişare. Aceasta poartă denumirea de metodă a clasei. În cadrul declarării acestei funcţii apare cuvântul cheie *public* care indică faptul că această metodă (funcţie) este accesibilă din exterior pentru obiectele instanţiate din această clasă. În momentul în care omiteam această declaraţie metoda era implicit accesibilă din exterior (toate proprietăţile şi metodele sunt implicit publice). În afară de posibilitatea de a declara o metodă sau o proprietate public mai există variantele *private* – acele metode sau proprietăţi sunt accesibile doar metodelor interne clasei, nu pot fi accesate prin intermediul obiectului instanţiat – sau *protected* – acele metode sau proprietăţi sunt accesibile doar intern în cadrul clasei sau în cadrul claselor derivate din clasa iniţială. Prin intermediul acestor metode de limitare a accesului la componentele interne ale unei clase se poate realiza o iniţializare controlată a proprietăţilor clasei (de exemplu, nu pot iniţializa proprietatea *varsta* în mod direct ci doar prin intermediul unei metode care verifică dacă valoarea introdusă este un întreg pozitiv mai mic de 100.

Test de autoevaluare

11.3 Definiţi o clasă care să poată fi utilizată pentru gestionarea mărfii într-un depozit de textile.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 125.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Problema 11.1.

```
define("PI",3.14);
$raza=3;
$arie=PI*pow($raza,2);
echo("<br>Aria cercului de raza ".$raza." este
                                        ".$arie." .");
```

Problema 11.2.

```
function max_vector($vector) {
 $maxim=0;
 foreach ($vector as $element) {
  if($element>$maxim):
   $maxim=$element;
  endif;
 }
 return $maxim;
}
$vector_test = array(1,4,63,45,45);
$element_maxim = max_vector($vector_test);
echo("<br>Elementul maxim este
                        ".$element_maxim." .");
```

Problema 11.3.

```
class marfa
{
 function __construct($stoc=0, $valoare_unitara=0)
 {
  $this->stoc = $stoc;
  $this->valoare_unitara = $valoare_unitara;
 }
 public function calcul_valoare_totala()
 {
  return($this->stoc*$this->valoare_unitara);
 }
}
$pijama = new marfa();
echo("Valoarea totala a stocului este ".
          $pijama->calcul_valoare_totala()." .");
```

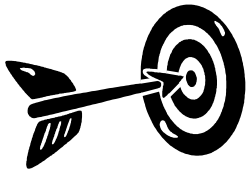
Bibliografie

1. Luke Welling şi Laura Thomson, Dezvoltarea aplicaţiilor web cu PHP şi MySQL, Editia a II-a, Editura Teora, 2005
2. Manualul PHP - http://php.net/manual/ro/index.php

Unitatea de învăţare Nr. 12

CAPACITATEA DE A FOLOSI INFORMAŢII DIN BAZE DE DATE ÎN CADRUL PAGINILOR WEB

Obiectivele Unităţii de învăţare Nr.12:



Principalele obiective ale Unităţii de învăţare Nr. 12 sunt:

- Ce este un sistem de baze de date şi care sunt tehnologiile de acces existente.
- Ce este şi cum funcţionează limbajul SQL.
- Cum putem utiliza sistemele de baze de date utilizând limbajul PHP şi limbajele de script pe partea de client.

12.1 Sisteme de baze date

Un sistem de baze de date este o **colecţie de date** centralizate, creată şi menţinută computerizat, în scopul prelucrării datelor în contextul unui set de aplicaţii. Prelucrarea datelor se referă la operaţiile de introducere, ştergere, actualizare şi interogare a datelor.

Simple colecţii de fişe sau tabele (de exemplu documente fizice pe hârtie sau tabele în Excel) ce conţin date dar nu permit operaţii de prelucrare automată (interogare, modificare, raportare) nu sunt considerate sisteme de baze de date.

Există mai multe tipuri de clasificări cu privire la sistemele de baze de date: clasificare după modelul de date (relaţionale, orientate obiect, obiect-relaţional, date ierarhizate), clasificare după numărul de utilizatori (monoutilizator, multiutilizator), clasificare după arhitectura de stocare (centralizate, distribuite) sau clasificare după arhitectura internă (sisteme de tip client-server sau sisteme integrate). Din perspectiva utilizării sistemelor de baze de date aceste clasificări influenţează modul în care se poate face accesul la baza de date, uşurinţa utilizării informaţiilor din baza de date şi performanţele de utilizare.

Exemple de sisteme de baze de date utilizate în cadrul aplicaţiilor web:

- Oracle – sistem de baze de date comercial destinat aplicaţiilor de dimensiuni şi complexitate mare (de exemplu site-ul amazon.com utilizează acest sistem de baze de date).
- Microsoft SQL Server – sistem de baze de date client-server destinat aplicaţiilor dezvoltate pe platforme Microsoft. Permite realizarea de aplicaţii complexe dar are performanţe foarte bune împreună cu alte tehnologii şi produse Microsoft (server web IIS, tehnologie de scripturi web ASP.NET).

- MySQL / PostgresSQL – baze de date open-source utilizate cu precădere sub sistemul de operare Linux. Sunt întâlnite într-o gamă largă de aplicaţii web de la site-uri personale până la site-uri ale companiilor mici şi mijlocii. Aplicaţiile web care utilizează aceste baze de date sunt scrise în mare parte în limbajul PHP.
- Microsoft Access – este un sistem de baze de date gândit pentru aplicaţii de tip office de mici dimensiuni. Este utilizat în aplicaţii web simple (formulare de preluare a datelor, pagini de înscriere). Are avantajul de a avea o modalitate de acces similară cu Microsoft SQL Server astfel încât o bază de date împreună cu aplicaţia corespondentă pot fi migrate uşor în momentul în care dimensiunea de stocare a datelor şi complexitatea operaţiilor efectuate cresc.

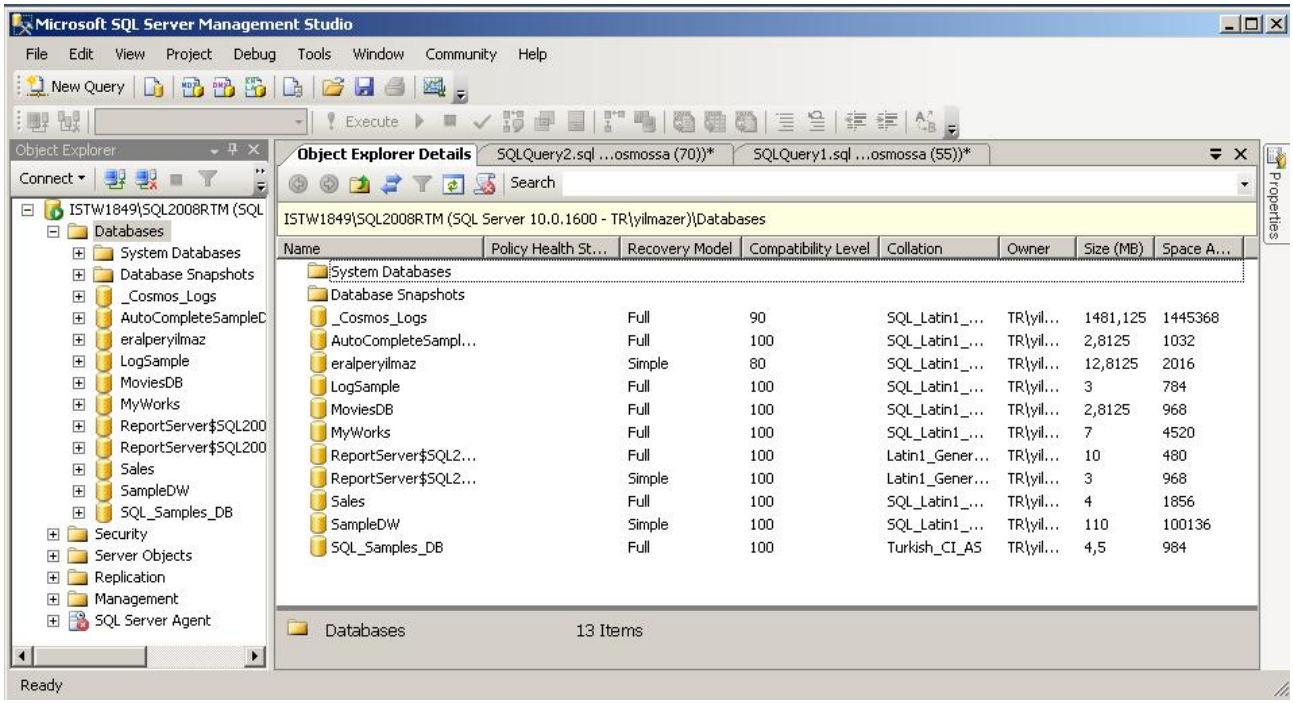


Figura 12.1 Interfaţa de management pentru un server de baze de date Microsoft SQL Server 2008

12.2 Tehnologii de acces la baze de date

Prin acces (sau conectare) la un sistem de baze de date înţelegem o modalitate în care putem să comunicăm în mod primar cu sistemul de baze de date – nu este vorba de manipularea informaţiilor conţinute în baza de date ci despre modul în care se transmit comenzile către sistemul de management al bazei de date (cum efectuăm autentificarea utilizatorilor, cum selectăm baza de date şi tabelele cu care dorim să lucrăm, cum transmitem comenzile de prelucrare a informaţiilor şi cum primim înapoi informaţiile dorite). Astfel majoritatea sistemelor de baze de date permit următoarele tehnici de acces:

- Acces prin intermediul unui **client propriu** – fie că vorbim de sisteme de tip server fie că vorbim de sisteme gândite să funcţioneze în mod monoutilizator întotdeauna va exista o modalitate pusă la dispoziţie de producătorul sistemului de baze de date care va permite accesul la sistem. Această tehnică este implementată prin punerea la dispoziţia programatorului a unei biblioteci de funcţii de acces sau a unui client în reţea. De cele mai multe ori această tehnică oferă performanţe optime şi permite accesarea întregii game de facilităţi a sistemului de baze de date.

- Acces prin intermediul unor **biblioteci de funcţii** puse la dispoziţie de limbajul de programare. Multe limbaje de programare vin în ajutorul programatorului punând la dispoziţia acestuia o serie de funcţii deja existente pentru facilitarea accesului la o serie de sisteme de baze de date des întâlnite. Avantajul acestor funcţii este faptul că ascund particularităţile sistemului de baze de date încercând să realizeze o modalitate similară de lucru independentă de sistemul de baze de date. Un exemplu de bibliotecă de acces specifică unui limbaj este **JDBC** (Java Database Connectivity) – tehnică de acces destinată limbajului de programare Java. Permite accesul bazelor de date relaţionale prin intermediul unei biblioteci de funcţii API. Alt exemplu constă în faptul că limbajul PHP (în instalare implicită) pune la dispoziţie o serie de funcţii pentru lucrul cu serverul MySQL.

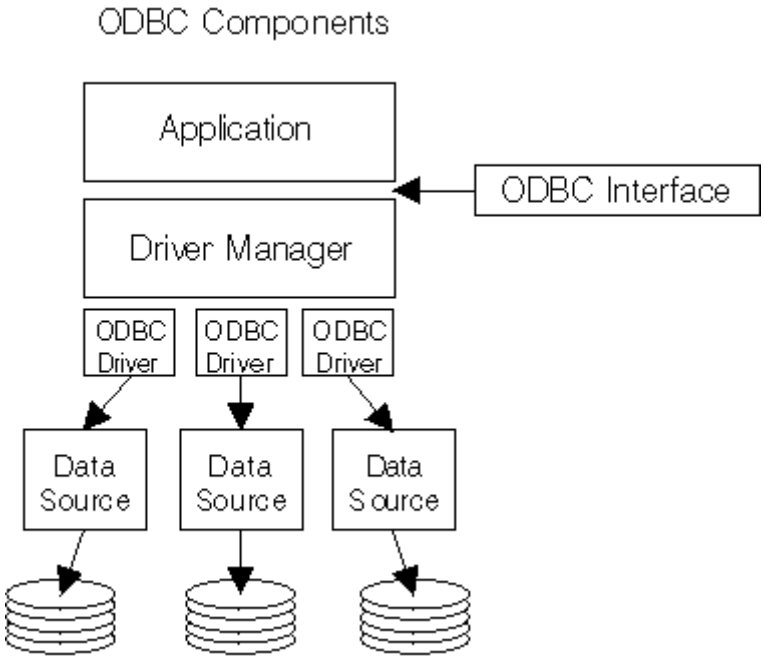


Figura 12.2 Arhitectura interfeţei ODBC

- **ODBC** (Open Database Connectivity) – este o tehnică care permite accesul la sistemul de management al bazei de date prin intermediul unei interfeţe software standard. Necesită un driver software care să efectueze translaţia între comenzile generice agreate de formalismul ODBC şi particularităţile sistemului de baze de date. În principal utilizarea acestei tehnici scuteşte programatorul de un efort suplimentar referitor la înţelegerea modalităţii de funcţionarea a sistemului de baze de date cu care lucrează dar degradează performanţele de prelucrare şi acces la informaţiile din baza de date. Tehnologia ODBC reprezintă o modalitatea de abstractizare a metodelor de acces la un sistem de baze de date. Există şi alte tehnologii de abstractizare a accesului la sisteme de baze de date dar au o popularitate mai redusă: DBA, dbx, PDO (PHP Data Objects), ADO, ADO.NET.

12.3 Introducere în SQL

Limbajul SQL (Structured Query Language) a fost conceput ca un limbaj standard de descriere a datelor şi acces la informaţiile din baze de date, ulterior dezvoltându-se ca o adevărată tehnologie dedicată standardizării modului de lucru cu informaţiile din sistemele de baze de date relaţionale.

Comenzile principale în cazul limbajului SQL se referă la cele cinci operaţii de bază care se pot efectua într-un limbaj relaţional:

- Crearea / ştergerea / modificarea unei tabele
- Inserarea de noi linii într-o tabelă
- Ștergerea unor linii dintr-o tabelă
- Modificarea unor linii dintr-o tabelă
- Listarea selectivă a datelor din una sau mai multe tabele

Operații cu tabele

Comanda de creare de noi tabele în baza de date curentă în limbajul SQL standard este **CREATE TABLE**. Sintaxa acestei comenzi este:

```
CREATE TABLE denumire_tabela (
    coloana1  tip_date_coloana1,
    coloana2  tip_date_coloana2,
    ........
);
```

Pe lângă declararea tipurilor de date pentru fiecare coloană de date a tabelei create pot apare şi alte informaţii care pot indica diverse constrângeri asupra datelor care vor putea fi introduse în acel câmp:

- NOT NULL – valorile din acel câmp nu vor putea lipsi în cadrul unei noi înregistrări;
- PRIMARY KEY – câmpul care are specificat acest cuvânt cheie este cheie primară a tabelului şi permite indexarea (sortarea şi căutarea mai rapidă) după valorile conţinute;

- FOREIGN KEY – acest câmp este o referinţă către o cheie primară dintr-o altă tabelă – lucru ce permite efectuarea de operaţii complexe pe ambele tabele simultan;
- DEFAULT – indică o valoare implicită pentru un câmp.

Exemplu de frază SQL care permite crearea unei tabele:

```
CREATE TABLE tabela_catalog (
     nr_crt integer AUTONUMBER PRIMARY KEY,
     nume text(50) NOT NULL,
     medie_matematica double,
     medie_limba_materna double,
     medie_geografie double,
     medie_istorie double);
```

În limbajul SQL comanda de ştergere a unei tabele în baza de date este **DROP TABLE**:

```
DROP TABLE denumire_tabela;
```

Ștergerea unei tabele este o operaţie permisă atâta timp cât nu există legături cu alte tabele (indexul primar al tabelei şterse nu este index extern – foreign key – al unei alte tabele din aceiaşi bază de date).

Modificarea unei tabele utilizând limbajul SQL presupune utilizarea instrucţiunii **ALTER TABLE**.

```
ALTER TABLE denumire_tabela
ADD nume_coloana tip_date_coloana
[DROP COLUMN nume_coloana]
[ALTER COLUMN nume_coloana nou_tip_date_coloana];
```

Lucru cu informații

Pe lângă instrucţiuni de lucru la nivel de tabelă, limbajul SQL oferă o serie de instrucţiuni care permit manipularea informaţiilor din tabelă.

Pentru a introduce date într-o tabelă avem la dispoziţie comanda **INSERT**:

```
INSERT INTO denumire_tabela [denumire_coloana,...]
VALUES (valoare1, valoare2,...);
```

Dacă nu se precizează coloanele ce urmează a fi populate cu date înseamnă că setul de valori precizat după cuvântul cheie VALUES cuprinde un set complet de valori pentru acea tabelă. Inserarea selectivă de valori (doar anumite câmpuri) trebuie făcută având în vedere eventualele câmpuri declarate NOT NULL.

Pentru a şterge informaţii dintr-o tabelă se utilizează comanda **DELETE**:

```
DELETE FROM denumire_tabela
[WHERE (conditie)] [LIMIT numar_maxim_linii]
```

Dacă în cadrul unei comenzi de tip DELETE nu apare clauza WHERE atunci toate liniile din acea tabelă vor fi şterse. Nu se pot şterge linii care conţin indecşi externi pentru alte tabele. Clauza LIMIT, care este opţională, limitează numărul de linii şterse la execuţia comenzii. Exemplu de comandă DELETE care va şterge din tabela tabela_catalog, dată ca exemplu anterior, toţi elevii care au media_matematica mai mică de 5:

```
DELETE FROM tabela_catalog
WHERE (medie_matematica<5);
```

Modificarea unor informaţii (linii) în cadrul unei tabele se face cu ajutorul comenzii **UPDATE**:

```
UPDATE denumire_tabela
          SET denumire_coloana1=valoare1,
          denumire_coloana2=valoare2,....
WHERE (conditie);
```

Efectul unei comenzi UPDATE este acela că va înlocui valorile din coloanele precizate după clauza SET pentru liniile care îndeplinesc condiţia specificată după clauza WHERE.

Limbajul SQL şi posibilitatea de filtrare a informaţiilor provenite dintr-un tabel. Acest lucru se face utilizând comanda **SELECT** care are următoarea sintaxă:

```
SELECT [DISTINCT] lista_coloane_rezultat
FROM denumire_tabela(e)
[WHERE (conditie)]
[GROUP BY denumire_coloana1, denumire_coloana2,...]
[HAVING conditie_de_grup]
[ORDER BY denumire_coloana1,...[ASC/DESC]];
```

După cum se poate observa comanda SELECT permite solicitarea extragerii unui set de informaţii dintr-o tabelă sau mai multe tabele. Clauza DISTINCT specifică că dorim să nu primim ca rezultat înregistrări duplicat. Acest lucru se poate întâmpla când efectuăm o selecţie pe mai multe tabele între care există legături şi condiţiile specificate în cadrul clauzei WHERE generează selecţii de perechi de înregistrări duplicate. Parametrul lista_coloane_rezultat este format dintr-o listă de denumiri de coloane care dorim să apară în filtrarea efectuată. Parametrul denumire_tabela(e) specifică denumirea tabelei sau a setului de tabele pe care executăm filtrarea.

Următorul exemplu este cea mai simplă frază de selecţie SQL care are ca efect selectarea tuturor înregistrărilor dintr-o tabelă. Se poate observa că în acest caz nu sunt prezente decât comanda SELECT şi clauza FROM.

```
SELECT * FROM denumire_tabela;
```

Următoarea frază SQL filtrează rezultatul selecţiei din cadrul tabelei tabela_catalog astfel încât să ne fie returnate doar coloanele nume şi medie_geografie pentru înregistrările care îndeplinesc condiţia ca medie_geografie să fie mai mare de 6. Înregistrările rezultate vor fi ordonate alfabetic după câmpul nume.

```
SELECT nume, medie_geografie
FROM tabela_catalog
WHERE (medie_geografie>6)
ORDER BY nume ASC;
```

În cazul în care definim şi următoarea tabelă:

```
CREATE TABLE tabela_clasa (
    nr_clasa integer PRIMARY KEY,
    denumire text(50) NOT NULL,
    );
```

şi modificăm tabelul tabela_catalog astfel încât să conţină şi informaţii legate de clasă:

```
ALTER TABLE tabela_catalog
ADD nr_clasa integer FOREIGN KEY;
```

putem efectua o filtrare combinată pe cele două tabele tabela_catalog şi tabela_clasa astfel încât să obţinem un rezultat care să includă nume, medie_istorie, denumire (clasa) ordonate după clasă şi după medie_istorie în mod descrescător.

```
SELECT tabela_catalog.nume,
 tabela_catalog.medie_istorie,tabela_clasa.denumire
FROM tabela_catalog, tabela_clasa
WHERE
    (tabela_catalog.nr_clasa=tabela_clasa.nr_clasa)
ORDER BY tabela_clasa.denumire,
                tabela_catalog.medie_istorie DESC;
```

Test de autoevaluare

12.1 Utilizând cele două tabele definite în cadrul capitolului scrieţi o frază SQL care să returneze numele elevilor din clasa a VII-a B (denumire) şi care să aibă medii mai mari ca 5 la toate materiile. Rezultatele trebuie să fie ordonate alfabetic după nume.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 125.

12.4 Folosirea PHP pentru accesul la baze de date din cadrul paginilor Web

Prin intermediul utilizării limbajului PHP pentru accesul informaţiilor din sisteme de baze de date putem genera pagini web dinamice sau interactive a căror conţinut să fie generat pe baza informaţiilor stocate în baza de date.

Limbajul PHP permite accesul sistemelor de baze de date prin mai multe metode:

- Metode abstracte:
  - DBA – Oracle Berkley DB
  - dbx
  - ODBC
  - PDO – PHP Data Object
- Biblioteci specifice unor sisteme de baze de date (selecţie):
  - dBase
  - Firebird/InterBase
  - Informix
  - IBM DB2 — IBM DB2, Cloudscape and Apache Derby
  - Mssql — Microsoft SQL Server
  - MySQL
  - OCI8 — Oracle OCI8
  - Paradox — Paradox File Access
  - PostgreSQL
  - SQLite
  - Sybase

Funcţiile utilizate pentru conectarea şi transmiterea comenzilor către sistemul de baze de date depind de metoda aleasă. Exemple de comenzi de conectare pentru diverse metode de acces:

ODBC:
```
odbc_connect(denumire_bd, user, parola, driver);
```

PDO (MySQL):
```
new PDO('mysql:host=...;dbname=...',user, parola);
```

MSSQL (Microsoft SQL Server):
```
mssql_connect(server, user, parola);
```

MySQL:
```
myssql_connect(server, utilizator, parola);
```

Cele mai multe aplicaţii web scrise în PHP ce utilizează baze de date sunt scrise pentru serverul de baze de date MySQL. Acest lucru se datorează tehnologiilor similare ale celor două produse – ambele produse sunt open-source şi rulează cu performanţe bune sub sistemul de operare Linux. Pentru aplicaţii web de dimensiune mică şi mijlocie (magazine virtuale, pagini de socializare, site-uri de

prezentare a unor produse, agenţii de turism sau chiar pagini personale) o soluţie integrată ce nu necesită costuri de licenţiere este o alegere la îndemână. Din acest motiv pentru exemplificarea completă a tuturor operaţiilor de lucru cu o bază de date prin intermediul limbajului PHP o să utilizăm biblioteca specifică de MySQL inclusă în limbajul PHP.

Operaţiile necesare pentru utilizarea informaţiilor dintr-o tabelă MySQL într-o pagină web sunt următoarele:

- conectarea la serverul MySQL
- selectarea bazei de date cu care lucrăm
- construirea frazei SQL pe care dorim să o rulăm
- execuţia acesteia
- primirea rezultatelor şi afişarea în cadrul paginii web
- repetarea ultimelor trei operaţii de câte ori este necesar
- închidere conexiunii cu serverul MySQL

După cum am văzut deja conectarea la un server MySQL se face specificând denumirea serverului, a unui nume de utilizator şi a unei parole.

```
<?php
$db_numeserver = "localhost";
$db_user = "utilizator";
$db_parola = "parola";
$db_server = mysql_connect($db_numeserver,$db_user,
                                        $db_parola);
if (!$db_server) die ("Nu ma pot conecta la
          serverul MySQL. Motiv: ". mysql.error());
?>
```

Scriptul anterior utilizează funcţia *mysql_connect* pentru a realiza conexiunea cu serverul de baze de date utilizând adresa şi setul de credenţiale furnizat. De cele mai multe ori serverul web care rulează scriptul este instalat pe aceiaşi maşină de calcul cu server de baze de date. Din această cauză la numele serverului exemplul foloseşte denumirea *localhost* ce identifică acelaşi sistem de calcul. Funcţia *mysql_connect* returnează un identificator al conexiunii realizate, identificator reţinut în variabila *$db_server*. Dacă conexiunea nu a putut fi realizată (server nefuncţional sau credenţiale necorespunzătoare) conţinutul variabilei este nul. Prin intermediul instrucţiunii *if* se verifică acest lucru şi în cazul în care eşuăm în realizarea conexiunii scriptul PHP este oprit (cu ajutorul comenzii *die*) şi este afişat motivul (cu ajutorul funcţiei *mysql.error()*). Această abordare tratează în mod primar apariţia unei erori la conectare. Pentru aplicaţii reale putem înlocui acest mesaj cu unul mai protocolar care să ascundă utilizatorului motivul erorii şi să semnaleze doar administratorului detaliile tehnice necesare (prin intermediul unui email sau prin consemnarea într-un fişier jurnal).
Selectarea bazei de date cu care dorim să lucrăm se face în mod similar cu realizarea conexiunii.

```
<?php
 $db_numebd = "test";
 mysql_select_db($db_numedb) or die ("Nu ma pot
  conecta la baza de date. Motiv: ". mysql.error());
?>
```

Comanda *or* permite ca în cazul în care prima procedură (mysql_select_db) se încheie cu o eroare intră în execuţie cea de a doua instrucţiune (*die*).

Pentru a lansa în execuţie o frază SQL avem la dispoziţie funcţia *mysql_query*. Prin intermediul acesteia putem executa atât fraze SQL la nivel de structură de tabelă (CREATE, ALTER, DELETE) cât şi fraze SQL de prelucrare sau filtrare a informaţiilor (SELECT, UPDATE).

Exemplu de execuţie a unei fraze SQL de modificare a unei tabele.

```
<?php
 $query = "ALTER TABLE tabela_catalog
                ADD nr_clasa integer FOREIGN KEY;";
 $result = mysql_query($query);
 if (!$result) die ("Executie esuata.
                          Motiv: ". mysql.error());
?>
```

Exemplu de execuţie a unei fraze SQL de filtrare a informaţiilor. Scriptul va afişa datele rezultate în cadrul unei pagini web.

```
<?php
 $query = "SELECT nume, medie_istorie
           FROM tabela_catalog
           WHERE (medie_istorie>5)
           ORDER BY nume ASC;";
 $result = mysql_query($query);
 if (!$result) die ("Executie esuata.
                          Motiv: ". mysql.error());
 $nr_linii = mysql_num_rows($result);

 echo "<table><tr> <th>Nume </th> <th>Medie
                                istorie</th></tr>";

for ($j = 0 ; $j < $nr_linii ; ++$j)
{
 $linie = mysql_fetch_row($result);
 echo "<tr>";
 echo "<td>$linie["nume"]</td>";
 echo "<td>$linie["medie_istorie"]</td>";
 echo "</tr>";
}
echo "</table>";
?>
```

Exemplul anterior lansează în execuţie o frază SQL de tip SELECT şi preia în variabila *$result* setul de informaţii rezultat. Cu ajutorul funcţiei *mysql_num_rows* determină dimensiunea rezultatului (numărul de linii care respectă condiţia din clauza WHERE) şi efectuează o buclă *for* în scopul de a construi un tabel HTML care să afişeze rezultatele. În cadrul buclei fiecare linie din rezultat este preluată cu ajutorul funcţiei *mysql_fetch_row* şi afişată ca o linie în tabel.

Pentru a închide conexiunea cu serverul MySQL, la finalul operaţiilor efectuate, se utilizează comanda *mysql_close*. Dacă în cadrul unei pagini PHP (sau a unei alte aplicaţii cu orice alt sistem de baze de date) nu se închide conexiunea la finalul secţiunii de cod această conexiune va rămâne deschisă un timp prestabilit după care se va închide în mod automat. Dacă scriptul PHP este accesat de un număr mare de ori într-un interval de timp mai mic decât cel de închidere automată presetat există posibilitatea ca serverul de baze de date să atingă numărul maxim de conexiune deschise simultan şi să se blocheze (să nu accepte alte conexiuni).

```
<?php
 mysql_close($db_numeserver);
?>
```

Atenţie! Chiar dacă anumite credenţiale permit efectuarea unor operaţii de tip SELECT sau UPDATE la nivel de tabelă asta nu înseamnă că aceleaşi credenţiale nu pot avea interzise alte tipuri de operaţii (de exemplu de modificare a tabelei – ALTER sau DELETE).

Pentru a înţelege mai bine modul în care funcţionează drepturile de acces în cadrul serverului MySQL puteţi consulta manualul online a acestuia:

http://dev.mysql.com/doc/refman/5.5/en/

Test de autoevaluare

12.2 Utilizând următoarea tabelele definite în secţiunea anterioară (tabela_catalog şi tabela_clasa) realizaţi un script PHP care şă afişeze într-un tabel HTML toţi elevi grupaţi pe clase şi ordonaţi descrescător după media de la matematică.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 140.

12.5 Limbaje de script pe partea de client

Din punct de vedere al prelucrării datelor existente într-o bază de date pe partea de client putem să identificăm următoarele situaţii:

- Utilizarea limbajelor de script pentru validarea datelor ce urmează a fi trimise către un script (pe partea de server) de introducere în baza de date. Această situaţie este foarte des întâlnită în cadrul aplicaţiilor web ce utilizează baze de date deoarece este mai eficient (şi mai rapid) să verifici corectitudinea datelor introduse de utilizator la nivel de script client decât să le trimiţi către un script server care, în cazul în care ceva este în neregulă, să returneze un mesaj de avertizare şi să redirecţioneze înapoi în pagina de introducere.

- Utilizarea limbajelor de script client pentru accesul bazelor de date locale (aflate pe maşina client). Majoritatea scripturilor de tip client (JavaScript, VBScript) permit instanţierea de obiecte de acces la baze de date (ODBC sau ADO). Prin această metodă se poate avea accesul la baze de date locale sau declarate local (bazele de date ODBC permit şi conexiuni la baze de date aflate pe alte calculatoare) dar nu permit accesul la baze de date aflate de serverul web (ca în exemplele prezentate în secţiunea anterioară). Din păcate această situaţie ridică semne de întrebare legate de securitatea datelor şi despre rostul aplicaţiei web. Putem valida din punct de vedere funcţional o pagină web care înlocuieşte aplicaţia Microsoft Access şi permite lucrul cu fişiere de baze de date locale dar nu putem concepe rostul unei aplicaţii web care îşi propune să utilizeze o bază de date centrală prin intermediul unei conexiuni ODBC declarate pe maşina client (în acest caz nu mai putem vorbi de protecţia bazei de date).

- Utilizarea de tehnologii de tip Ajax (Asynchronous JavaScript and XML) care permit transferul, în timpul execuţiei scriptului client, de informaţii de pe server. În felul acesta putem vorbi de un schimb de informaţii între scriptul client şi scriptul server şi putem imagina o combinaţie de tehnologii client şi server (JavaScript / PHP de exemplu) care să permită scriptului client să obţină şi să prelucreze informaţii provenite dintr-o bază de date aflată pe server. Un exemplu de aplicaţie web care utilizează tehnologia Ajax este Google Maps.

În continuare vom exemplifica prima situaţie de utilizarea a scripturilor client (de altfel şi cea mai des întâlnită în cadrul aplicaţiilor web) şi anume vom utiliza un script JavaScript pentru a valida câmpurile dintr-un formular web (câmpuri ce pot fi trimise ulterior unui script PHP pentru a le introduce într-o tabelă dintr-o bază de date).

```
<html>
<head>
<title>Exemplu de formular cu validare</title>
</head>
<body>
<form method="POST" action="insert.php"
                                name="info_form">
<p>Nume:
<input type="text" name="nume" size="50"></p>
<p>Medie matematica:
<input type="text" name="medie_mat" size="2"></p>
<p><input type=button value="TRIMITE"
onclick="javascript:
if(window.document.info_form.nume.value=='')
{window.alert('Completati numele!');
 window.document.info_form.nume.focus();}
else {window.document.info_form.submit();}"></p>
</form>
</body>
</html>
```

Iar pe partea de server putem avea următorul script PHP (insert.php):

```
<?php
$nume = $HTTP_POST_VARS[nume];
$medie_mat = $HTTP_POST_VARS[medie_mat];
$query = "INSERT nume, medie_matematica VALUES ('".
          $nume."'," . $medie_mat. ") ".
          "INTO tabela_catalog";
$result = mysql_query($query);
if (!$result) die ("Introducere esuata.
                          Motiv: ". mysql.error());
?>
```

Test de autoevaluare

12.3 Completaţi funcţia JavaScript din exemplul de mai sus pentru a verifica şi corectitudinea completării câmpului medie matematică.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 140.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Problema 12.1.

```
SELECT tabela_catalog.nume
FROM tabela_catalog, tabela_clasa
WHERE
((tabela_catalog.nr_clasa=tabela_clasa.nr_clasa)
AND
(tabela_clasa.denumire= 'VII-a B ') AND
(tabela_catalog.medie_matematica>5) AND
(tabela_catalog.medie_limba_materna>5) AND
(tabela_catalog.medie_geografie>5) AND
(tabela_catalog.medie_istorie>5))
ORDER BY tabela_catalog.nume ASC;
```

Problema 12.2.

Se modifică din exemplul de la pagina 136 doar fraza SQL:

```
SELECT tabela_catalog.nume,
                    tabela_catalog.medie_matematica
FROM tabela_catalog, tabela_clasa
WHERE
(tabela_catalog.nr_clasa=tabela_clasa.nr_clasa)
ORDER BY tabela_clasa.denumire,
          tabela_catalog.medie_matematica DESC;
```

Problema 12.3.

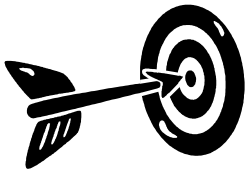
```
javascript:
if(window.document.info_form.nume.value=='')
{window.alert('Completati numele!');
 window.document.info_form.nume.focus();}
else {
if(window.document.info_form.medie_mat.value=='')
{window.alert('Completati media de la
                                    matematica!');
 window.document.info_form.medie_mat.focus();}
else {window.document.info_form.submit();}}
```

Bibliografie

3. Luke Welling şi Laura Thomson, Dezvoltarea aplicaţiilor web cu PHP şi MySQL, Editia a II-a, Editura Teora, 2005
4. Manualul PHP - http://php.net/manual/ro/index.php

Unitatea de învăţare Nr. 13
XML ŞI FOLOSIREA LUI IN PAGINA WEB.

Obiectivele Unităţii de învăţare Nr.13:



Principalele obiective ale Unităţii de învăţare Nr. 13 sunt:

- Ce este limbajul XML şi la ce poate fi folosit
- Ce este şi care este structura unui document XML
- Cum poate fi folosit un document XML într-o pagină HTML

13.1 Introducere în XML

Limbajul XML (e**X**tended **M**arkup **L**anguage) este un nou limbaj de adnotare, care este folosit intens de tehnologiile software actuale pentru *schimburi de informaţii* între diverse aplicaţii din Internet sau reţele. În cazul în care aceste aplicaţii funcţionează conform unor standarde diferite, XML este limbajul comun în care acestea pot schimba informaţii. Această facilitate este foarte importantă în cazul comunicării datelor între companii şi firme. Dacă este folosit XML nici una din companii nu trebuie să cunoască modul de structurare a datelor în baza de date a partenerului ci doar să stie structura XML de export a acestor date pentru a avea acces la informaţiile necesare.

XML nu este un înlocuitor al HTML, ci este mai degrabă un complement al acestuia. În vreme ce în cazul HTML atenţia este orientată spre modul în care trebuie să fie afişată informaţia conţinută de directivele HTML, XML este realizat cu unicul scop de a oferi suport pentru structurarea informaţiilor.

La realizarea acestui limbaj s-au stabilit următoarele obiective:

- XML va fi compatibil cu SGML (un alt limbaj de adnotare).
- Documentele XML vor putea fi uşor citite şi interpretate de către om.
- Documentele XML vor putea fi proiectate şi realizate uşor.
- Programele pentru a interpreta documentul XML trebuie să poată fi realizate uşor.
- Să fie destul de flexibil pentru a putea fi utilizat într-o gamă largă de aplicaţii.

Limbajul XML nu poate executa nici o operaţie

Limbajul XML nu este conceput să ofere instrucţiuni pentru execuţia unor operaţii, ci este conceput pentru a “înpacheta” informaţia cu scopul de a fi trimisă sau stocată. Pentru a face ceva cu aceste date cineva trebuie să scrie un program într-un limbaj de programare oarecare care să interpreteze aceste date şi să le folosească într-un anume scop.

Iată un exemplu de fişier XML:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<articol>
  <titlu>
       Impozitarea tranzactiilor imobiliare
  </titlu>
  <descriere>
       Terenurile sunt din ce in ce mai
       scumpe in Capitala dupa ce ...
  </descriere>
   <URL>
       http://stiri.ro/articol123.html
   </URL>
</articol>
<articol>
...
</articol>
```

În exemplul de mai sus se constată că este vorba despre un document XML care ar putea fi folosit de o aplicaţie ce afişează revista presei. Documentul XML din exemplul de mai sus conţine informaţii despre articolele disponibile în ziarele de azi

- Nume articol - `<titlu>`
- O scurtă descriere - `<descriere>`
- Locaţia unde poate fi găsit acest articol - <URL>

Se vede că în acest document sunt prezente numai informaţii cu privire la structura sau tipul datelor şi bineînteles datele propriuzise. Acestea vor putea fi folosite de către o aplicaţie care să afişeaze revista presei folosind datele din acest document XML care ar putea fi transmise periodic de la un server.

13.2 Caracteristici ale XML

Principalele caracteristici ale XML sunt:

- **XML nu are directive predefinite**, acestea vor fi relizate în momentul proiectării documentului XML în funcţie de scopul acestuia şi tipul sau structura datelor.
- Pentru descrierea directivelor folosite, XML utilizează un set de reguli denumit **DTD** (**D**ocument **T**ype **D**efinition) sau o **schemă XML**. Împreună cu această schemă se spune despre XML că este autodescriptiv.
- Documentul XML trebuie să respecte strict regulile definite în DTD pentru a fi un document **valid.**
- Documentele XML au o structură strictă **(Well-formed)**. Asta înseamă că documentul trebuie să respecte anumite reguli de sintaxă prezentate în Unitatea de învăţare următoare.

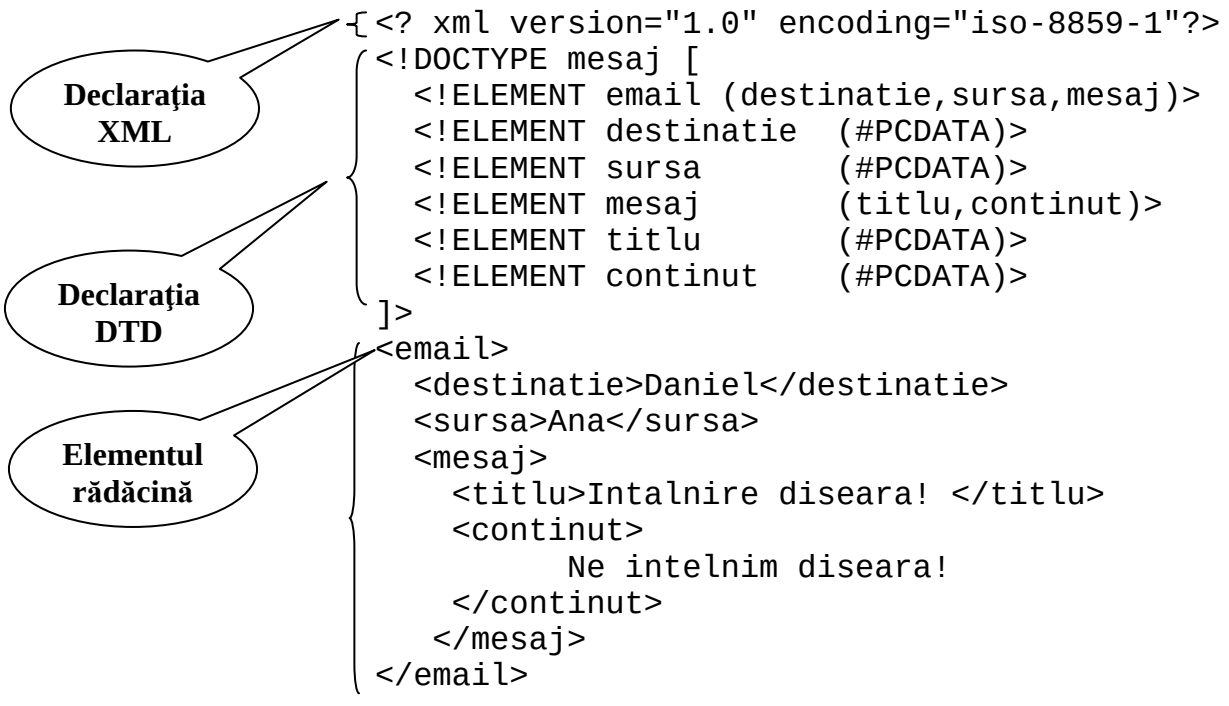
- XML este un limbaj de adnotare complementar cu HTML şi **NU** înlocuieşte HTML. Cele două limbaje, după cum s-a arătat, au scopuri distincte: HTML afişează informaţii ce apar în pagină Web, în timp ce XML descrie aceste informaţii.

13.3 Sintaxa XML

Un document XML este format din două blocuri principale astfel:

- **Antetul** documentului care cuprinde la rândul său:
  - ***Declaraţia XML*** – Se află întotdeauna pe prima linia a documentului şi conţine informaţii despre *versiunea XML* şi *standardul de codificare al caracterelor*.
  - ***Declaraţia DTD*** care conţine informaţii despre structura documentului. Documentul este verificat dacă este valid sau nu conform cu această declaraţie
- **Documentul XML propriuzis** Un document XML are o structură arborescentă având un singur element rădăcină marcat de o pereche de directive XML. Toate celelalte elemente sunt descendenţi şi conţinute în interiorul elementului rădăcină Un element poate conţine la rândul său alte subelemente.

Un exemplu complet:




```
<? xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE mesaj [
  <!ELEMENT email (destinatie,sursa,mesaj)>
  <!ELEMENT destinatie  (#PCDATA)>
  <!ELEMENT sursa       (#PCDATA)>
  <!ELEMENT mesaj       (titlu,continut)>
  <!ELEMENT titlu       (#PCDATA)>
  <!ELEMENT continut    (#PCDATA)>
]>
<email>
  <destinatie>Daniel</destinatie>
  <sursa>Ana</sursa>
  <mesaj>
    <titlu>Intalnire diseara! </titlu>
    <continut>
          Ne intelnim diseara!
    </continut>
   </mesaj>
</email>
```

În exemplul de mai sus:

- Prima linie reprezintă Declaraţia XML,
- Este urmată de Declaraţia DTD care conţine reguli despre structura documentului într-un anumit format
- Documentul XML propriuzis are ca element rădăcină elementul reprezentat cu ajutorul directivei

<email></email>. Elementele care urmează marcate de directivele <destinatie>, <sursa> şi <mesaj> sunt descendenţi de rangul 1. Elementul <mesaj> are la rândul său descendenţi elementele marcate de directivele <titlu> şi <conţinut>. Nivelul de imbricare poate continua atât cât este nevoie şi nu sunt stabilite limite.

Directive XML

Directivele XML sunt identificatori pentru elementele XML şi sunt plasate ca şi în cazul directivelor HTML între paranteze ascuţite “<” şi “>” Iată care sunt regulile pentru folosirea directivelor XML:

Orice directivă, ex. <directivă>, trebuie să aibă o directivă pereche de încheiere ex </directivă>. Omiterea directivei de încheiere nu este permisă ca în cazul unor directive HTML
Directivele XML sunt sensibile la scrierea cu majuscule sau litere mici prin urmare directiva <Titlu> este diferită de directiva <titlu>.
În cazul în care se folosesc mai multe niveluri de imbricare ordinea închiderii directivelor trebuie să fie inversă decât ordinea în care au fost deschise.
Pentru elementele care nu conţin text se poate folosi o altă modalitate de definire a directivelor.. Iată cum:

```
<directiva />
```

Se observă că s-a folosit un caracter “/” după numele elementului. În acest caz NU se mai specifică directiva pereche de încheiere

Atribute XML

Directivele XML pot avea atribute pentru a putea specifica informaţii suplimentare despre acel element. Atributele sunt definite sub formă de perechi nume-valoare. Valorile atributelor se scriu obligatoriu între ghilimele ca în exemplul de mai jos:

```
...
<mesaj data=”01/08/2005”>
...
</mesaj>
...
```

În XML se pot folosi comentarii care au o sintaxă similară cu cea din HTML, adică:

```
<!-- Orice text explicativ --!>
```

Acestea pot apărea oriunde în documentul XML.

Test de autoevaluare

13.1 XML poate fi considerat ca fiind o alta versiune de HTML?

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 149.

13.4 Modul de folosire a XML în pagina Web

Aşa cum am spus mai sus XML a fost realizat ca un limbaj folosit pentru a structura datele pentru a fi trimise spre prelucrare în cadrul altor aplicaţii. HTML are ca principal scop definirea modului în care sunt afişate date.

Cele două concepte pot fi folosite împreună pentru a putea separa datele care trebuiesc afişate de instrucţiunile de afişare într-o pagină Web, implementând astfel conceptul de ***insulă de date*** şi permiţând afişarea, selectivă sau totală a datelor conţinute într-un document XML într-o pagină HTML

Acest lucru este posibil utilizând în cadul paginii HTML directiva HTML <XML> pentru a o “lega” pe aceasta de datele din fişierul XML ca în exemplul de mai jos:

Presupunând ca avem următorul fişier XML care contine date despre studentii admisi la un anumit examen şi nota obtinuta, ne propunem să realizăm o pagină HTML care să utilizeze datele din fişierul XML pentru a afişa un tabel cu studenţii admişi într-o pagină Web.

Fişierul listaadmitere.xml:

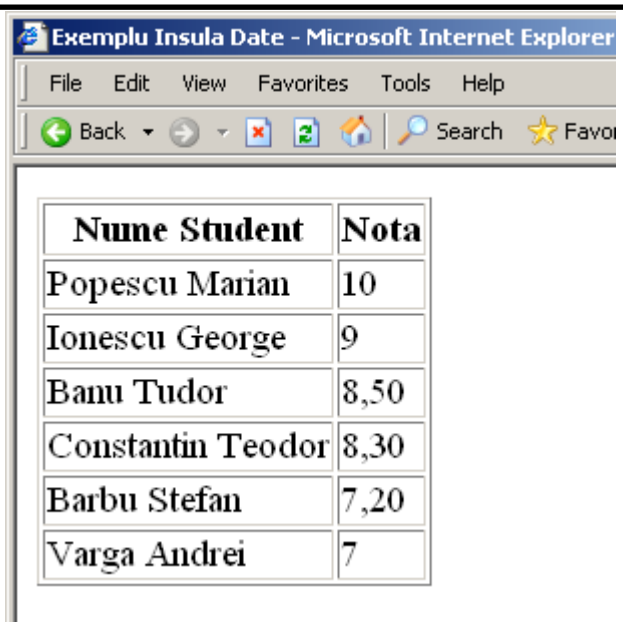
```
<?xml version="1.0" ?>
- <listaadmitere>
  - <student>
      <nume>Popescu Marian</nume>
      <nota>10</nota>
    </student>
  - <student>
      <nume>Ionescu George</nume>
      <nota>9</nota>
    </student>
  - <student>
      <nume>Banu Tudor</nume>
      <nota>8,50</nota>
    </student>
  - <student>
      <nume>Constantin Teodor</nume>
      <nota>8,30</nota>
    </student>
  - <student>
      <nume>Barbu Stefan</nume>
      <nota>7,20</nota>
    </student>
  - <student>
      <nume>Varga Andrei</nume>
      <nota>7</nota>
    </student>
  </listaadmitere>
```

Fişierul publicListaAdmitere.html

```
<HTML>
<HEAD>
<TITLE>
   Exemplu Insula Date
</TITLE>
</HEAD>
<BODY>
<XML ID="admitere"
     SRC="listaadmitere.xml"
     async="false"></XML>

<TABLE BORDER="1"
       DATASRC="#admitere">
 <THEAD>
  <TH> Nume Student </TH>
  <TH> Nota </TH>
 </THEAD>
 <TR>
 <TD>
   <SPAN DATAFLD="nume">
   </SPAN>
 </TD>
 <TD>
   <SPAN DATAFLD="nota">
   </SPAN>
 </TD>
 </TR>
</TABLE>
</BODY>
<HTML>
```

Deschiderea paginii publicListaAdmitere în browser va avea ca rezultat afişarea datelor din fişierul XML sub forma unui tabel a cărui formă de afişare a fost definit în pagina HTML precum în figura 13.1:



Nume Student	Nota
Popescu Marian	10
Ionescu George	9
Banu Tudor	8,50
Constantin Teodor	8,30
Barbu Stefan	7,20
Varga Andrei	7

Figura 13.1 Exemplu insula de date.

Test de autoevaluare

13.2 Definiţi conceptul de insulă de date.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 149.

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 13.1.

NU XML este un limbaj de adnotare complementar cu HTML şi **NU** înlocuieşte HTML. Cele două limbaje, după cum s-a arătat, au scopuri distincte: HTML afişează informaţii ce apar în pagină Web, în timp ce XML descrie aceste informaţii. Revedeţi indicaţiile din secţiunea 11.1.

Întrebarea 13.2.

Conceptul de ***insulă de date*** se referă la separarea datelor de afişat de instrucţiunile de afişare dintr-un document HTML. Datele vor fi citite dintr-un document XML şi afişate conform instrucţiunilor de afişare din documentul HTML. Pentru nelămuriri revedeţi secţiunea 11.4.

Indicaţii la problemele propuse

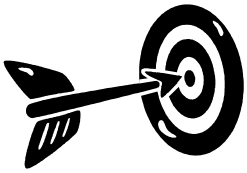
Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

5. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.59-82
6. http://www.w3schools.com/default.asp

Unitatea de învăţare Nr. 14
FOI DE STIL

Obiectivele Unităţii de învăţare Nr. 14:



După parcurgerea acestei Unităţii de învăţare vei şti:

- Care sunt avantajele folosirii foilor de stil
- Sintaxa folosită pentru definirea unei foi de stil
- Metode de folosire a informaţiilor de stil într-un document HTML

14.1 Foi de stil în HTML - CSS

Foie de stil

Stilurile CSS au fost recent adăugate la limbajul HTML cu scopul de a adăuga mai multe facilităţi pentru controlul modului de apariţie al paginii. Începând cu HTML 4, toate instrucţiunile de formatare pot fi definite în afara documentului HTML într-o strucutră denumită **foaie de stil**.

În literatura de specialitate foile de stil sunt referite de acronimul **CSS** care vine de la **C**ascading **S**tyle **S**heets. Cuvântul "*Cascading* " înseamnă "în cascadă" şi se referă la modul de aplicare al acestor stiluri asupra elementelor din pagina WEB .atunci când sunt folosite simultat mai multe foi de stil.

Avantaje

Acestea sunt avantajele folosirii foilor de stil:

- Separarea instrucţiunilor de formatare de structura documentului HTML
- Documentele HTML care folosesc foi de stil pot avea dimensiuni mai mici
- Site-uri mai uşor de menţinut. Prin folosirea unui set comun de foi de still pentru toate paginile HTML se poate schimba înfăţişarea unui site care conţine sute de pagini WEB prin editarea unui singur fişier.

Dezavantaje

Dezavantajul este că încă există browsere care nu au implementat suport pentru foi de stil. Dar chiar si pentru acest dezavantaj există un remediu. Paginile de web pot fi iniţial realizate fără a folosi facilităţile de stil.

În acest fel se poate verifica modul în care acestea sunt afişate de browserele fără suport pentru CSS. Informaţiile de stil urmâd a fi adăugate pe urmă păstrând în acelaşi timp nealterat conţinutul documentului HTML.

14.2 Cum funcţionează stilurile în HTML

Sintaxa de bază

O foaie de stil constă în una sau mai multe reguli care descriu modul în care un element dintr-o pagină HTML va fi afişat. O regulă într-o foie de stil are următorul format:

```
selector{proprietate:valoare}
```

unde:

Selector

selector – identifică elementul HTML pentru care se specifică modul de afişare. Practic orice element HTML poate fi un posibil selector CSS. În exemplul de mai jos, **P** este un selector de stil şi va defini pentru elementele de tip <P> din documentul care foloseşte această foie de stil dimensiunea fontului de 12 puncte

```
P {font-size: 12pt;}
```

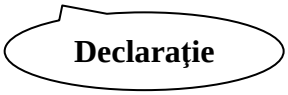
Proprietate -valoare

proprietate şi **valoare** definesc stilul care va fi aplicat elementului identificat de selector. Proprietatea este separată de valoare prin caracterul ":" urmată de un spaţiu. Această pereche, proprietate-valoare, este încadrată între acolade şi reprezintă ***declaraţia*** stilului.

Declaratie

O declaraţie de stil poate conţine mai multe perechi proprietate-valoare separate prin caracterul ";" ca în exemplul de mai jos:

```
P {font-size: 12pt;
    font-face: Arial}
```



Gruparea stilurilor

Pentru a nu folosi în mod repetat aceeaşi declaraţie pentru mai multe tipuri de elemente, CSS oferă posibilitatea grupării selectorilor. Spre exemplu toate titlurile dintr-un document HTML pot fi configurate să aibă anumite proprietăţi comune:

```
H1, H2, H3, H4, H5, H6 {
       color: blue;
       font-face: Arial}
```

Test de autoevaluare

14.1 Care este principalul avantaj al folosirii foilor de stil la realizarea paginilor de WEB?

14.2 Ce este un selector într-o foaie de stil?

Răspunsurile corecte şi comentarii asupra acestora se găsesc la pagina 159.

14.3 Adăugarea foilor de stil în documentul HTML

Regulile şi seturile de reguli pot fi incluse în documentul HTML în trei feluri: ca stiluri inline, ca foaie de stil intern sau ca foie de stil externă.

Stiluri INLINE

Stilurile inline.sunt adăugate fiecărui element HTML în parte prin folosirea atributului STYLE care este disponibil pentru majoritatea directivelor HTML. Valoarea atributului este reprezentată de una sau mai multe declaraţii de stil ca în exemplul de mai jos:

```
<H1 STYLE="color: blue">
   Acest titlu este albastru
</H1>
```

sau:

```
<P
   STYLE="font-size: 12pt;
          font-face: Verdana">
     Acesta este un paragraf cu stilul definit
   inline.
```

Deşi este o construcţie perfect valabilă pentru HTML, nu este recomandată deoarece nici unul din avantajele foilor de stil nu poate fi folosit în acest caz. Dacă se doreşte modificarea modului de afişare pentru un set de elemente este necesară intervenţia în documentul HTML asupra fiecărui element în parte.

Foi interne de stil

O metodă mult mai compactă şi mai eficientă este folosirea foilor interne de stil. O foaie internă de stil se defineşte în headerul documentului HTML folosind directiva pereche <STYLE> </STYLE>.

În interiorul acestui element se pot defini stilurile elementelor din pagină folosind sintaxa descrisă mai sus, Spre exemplu:

```
<STYLE TYPE=”test/css”>
  <!--
     P {color: blue};
     H1, H2, H3, H4, H5{
        color: aqua;
        font-size: 12pt
     }
  --}
</STYLE>
```

Se observă că în interiorul elementului STYLE declaraţiile de stil sunt încadrate de comentariu HTML: <!-- şi -->. Aceasta este pentru cazul în care pagina este vizualizată într-un browser care nu suportă foi de stil, pentru a evita afişarea informaţiilor din cadrul acestui element în pagină.

O foaie internă de stil se poate aplica unui singur document HTML. Aceasta poate modifica aspectul paginii păstrând în acelaşi timp nealterat conţinutul documentului HTML.

Foi externe de stil

Cea mai puternică metodă de folosire a foilor de stil este de a le defini pe toate în interiorul unui fişier separat la care să se facă referire în toate documentele HTML care vor folosi acest stil. Aceste foi de stil poartă denumirea de foi externe de stil, iar pentru referirea lor există două modalităţi.

Legarea

Legarea (linking) Este cea mai des folosită metodă şi constă în folosirea directivei <LINK>. Această directivă se defineşte în interiorul headerului .

<LINK>
Funcţionalitate: • Importă o foaie de stil dintr-un fişier Atribute: • HREF • REL • TYPE Directiva de sfârşit: **</LINK>** OBLIGATORIE

Iată un exemplu de folosire a acestei directive pentru importarea unei foi de stil definită într-un fişier *stilulmeu.css* extern aflat în acelaşi director cu documentul HTML:

```
<HEAD>
<LINK
      REL="STYLESHEET"
      HREF="stilulmeu.css"
      TYPE="text/css">
</HEAD>
```

Atributul REL Atributul **REL** – Defineşte relaţia dintre documentul HTML şi fişierul cu care se face legătura – o foaie de stil: “STYLESHEET”.

Atributul HREF Atributul **HREF** – Reprezintă URL-ul fişierului care conţine declaraţiile de stil.

Atributul TYPE Atributul **TYPE** – Defineşte formatul fişierului ce conţine informaţiile de stil. Se foloseşte valoarea “*text/css*”.

Import O metodă alternativă pentru folosirea foilor externe de stil este folosirea directivei <STYLE> împreună cu declaraţia *@import* după cum este arătat în exemplul de mai jos:

```
<STYLE
     TYPE="text/css">
  <!--
   @import url(http://stiluri.ro/stilulmeu.css);
   @import url(altstil.css);
   DT { background: yellow;
        color: black }
   -->
</STYLE>
```

Declaraţia @import trebuie să fie înainte de orice altă declaraţie de stil.

Test de autoevaluare

14.3 Enumeraţi metodele de adăugare a informaţiilor de stil într-o pagină HTML.

Răspunsul corect şi comentarii asupra acestuia se găsesc la pagina 159.

14.4 Proprietăţi CSS

O foaie de stil este compusă dintr-un set de instrucţiuni de afişare. Fiecare instrucţiune este la rândul său compusă dintr-un *selector* care identifică elementul căruia i se aplică stilul şi o *declaraţie* care defineşte modul în care va fi afişat elementul respectiv. O declaraţie este o colecţie de proprietăţi CSS.

O proprietate CSS este o construcţie de forma nume/valoare. CSS împarte aceste proprietăţi in câteva categorii pricipale. Vom enumera aceste categorii şi vom trece în revistă proprietăţile din fiecare categorie în parte. Pentru o listă completă a proprietăţilor disponibile recomandăm consultarea materialelor din bibliografie.

Proprietati ale fontului

Proprietăţi ale fontului - Aceste proprietăţi definesc aspectul grafic al textului afişat. Majoritatea modifică caracteristicile fontului cu care este afişat textul. Iată câteva exemple de astfel de proprietăţi:

- font-family – defineşte tipul fontului folosit pentru afişarea textului
- font-size – defineşte dimensiunea fontului folosit pentru afişarea textului
- font-style – defineşte modul de afişare a fontului: bold, italic, subliniat

Exemplu de utilizare

```
P {font-family: Arial;
   font-size: 14pt;
   font-style: italic
   }
```

Proprietati pentru text

Proprietăţi pentru text - Aceste proprietăţi definesc modul de aliniere, şi spaţiere la afişarea unui text. Iată câteva exemple de astfel de proprietăţi:

- letter-spacing – defineşte dimesiunea spaţiului între litere
- text-align – defineşte modul de aliniere al textului
- text-intend – defineşte spaţiul folosint pentru intendarea textului

Exemplu de utilizare:

```
P {letter-spacing: 0.1em;
   text-align: center
   text-intend: 5em
   }
```

Proprietati pentru spatierea elementelor

Proprietăţi pentru spaţierea elementelor - Aceste proprietăţi sunt folosite pentru a definii spaţiul dintre elemente HTML, marginile acestora şi alte aspecte privind poziţionarea lor. Iată câteva exemple de astfel de proprietăţi:

- padding-top – defineşte dimensiunea spaţiului ce va fi pus între conţinutul elementului şi marginea superioară
- padding-right - defineşte dimensiunea spaţiului ce va fi pus între conţinutul elementului şi marginea dreaptă
- padding-left – similar cu padding-right pentru stânga.

Proprietati pentru definirea culorilor

Proprietăţi pentru definirea culorilor:

- color - Această proprietate este folosită pentru a defini culoarea textului din cadrul elementului specificat de către selector. Culoarea poate fi specificată fie prin codul RGB, fie prin numele predefinit.

Proprietati pentru definirea fundalului

Proprietăţi pentru definirea fundalului - Aceste proprietăţi sunt folosite pentru a definii caracteristicile fundalului pentru elementul specificat de către selector. Iată câteva exemple de astfel de proprietăţi:

- background-color – defineşte culoarea fundalului
- backround-image – defineşte imaginea ce va fi afişată ca fundal.

Lucrare de verificare a Unităţilor de învăţare Nr. 10, 11, 12, 13 şi 14

1. Realizaţi un formular de înscriere pentru un concurs în care să solicitaţi numele, data naşterii, adresa poştală şi adresa de email. Validaţi datele cu ajutorul unui script client de tip JavaScript. Preluaţi datele introduse cu ajutorul unei pagini PHP şi introduceţi-le într-o bază de date.
Predaţi fişierul HTML şi cel PHP în format electronic ca rezultate al rezolvării lucrării de verificare.

Puteţi porni în rezolvarea problemei de la exemplul din exerciţiul 12.3 din testul de evaluare al unităţii de învăţare nr. 12.

Nr. puncte: **7** (4p. – scrierea corectă a codului HTML şi JavaScript din cadrul formularului, 3p – scrierea corectă a codului PHP de inserare în baza de date)

2. Realizaţi o pagină Web care să includă un applet Java al cărui cod se află în fişierul *film.class*. Appletul are deja scris codul pentru a rula un film de la un anumit URL însă acest URL trebuie transmis ca parametru de intrare. Poziţionaţi appletul în centrul

paginii.
Predaţi fişierul HTML în format electronic ca rezultat al rezolvării lucrării de verificare

Ca ajutor suplimentar în rezolvarea lucării se recopmandă utilizarea reperului bibliografic numărul 1 din bibliografia unităţii de învăţare nr.10.
Nr. puncte: **6** (3p. – utilizarea corectă a directivei APPLET, 2p - utilizarea atributului PARAM, 1p - poziţionarea apletului pe centrul paginii)

3. Realizaţi un document XML care să descrie şi să conţină informaţii despre produsele unui magazin, precum: Nume produs, descriere, pret, cantitate etc. Realizaţi pe urmă un document HTML care să afişeze aceste date din fişierul XML sub formă de tabel.
Predaţi documentele în format electronic (un fişier HTML şi un fişier XML) ca rezultat al rezolvării lucrării de verificare.

Ca ajutor suplimentar în rezolvarea lucării se recopmandă utilizarea reperului bibliografic numărul 1 din bibliografia unităţii de învăţare nr.11.

Nr. puncte: **6** (3p – realizarea corectă a fişierului XML, 3p realizarea fişierului HTML care să afişeze infirmaţiile din cadrul documentului XML)

4. Definiţi un fişier *stilulmeu.css* pe care să îl folosiţi pentru a construi o pagină HTML care să formateze elementele folosind foaia de stil din fişierul stilulmeu.css. Stilurile definite în acest fişier trebuie să afecteze următoarele elemente din pagina HTML:

- H1, H2 , H3 să fie scris cu font Veranda cu culoarea galbena
- Textul din documentul HTML să fie scris cu font de dimensiunea 10pt, culoarea textului să fie albă, iar culoarea de fundal să fie neagră.

Predaţi fişierul *stilulmeu.css* precum şi documentul HTML în format electronic ca rezultat al rezolvării lucrării de verificare
Ca ajutor suplimentar în rezolvarea lucării se recopmandă utilizarea reperului bibliografic numărul 1 din bibliografia unităţii de învăţare.

Nr. puncte: **6** (4p – definirea corectă în cadrul foii de stil a atributelor cerute, 2p – folosirea foii de stil in cadrul documentului HTML)

Răspunsuri şi comentarii la întrebările din testele de evaluare

Întrebarea 14.1.

Principalul avantaj este separarea informaţiilor de formatare de structura documentului HTML şi posibilitatea modificării modului de afişare a paginii fără a intervenii în documentul HTML. Pentru nelămuriri revedeţi secţiunea 14.1.

Întrebarea 14.2.

Un selector, este acea parte dintr-o regula CSS care identifică elementul HTML pentru care se specifică modul de afişare. A se revedea secţiunea 14.2.

Întrebarea 14.3.

Informaţiile de stil pot fi adăugate într-un document HTML folosind una din cele 3 metode: adăugarea informaţiilor inline, folosirea foilor interne de stil, folosirea foilor de stil definite într-un fişier extern. Consultaţi secţiunea 14.3.

Indicaţii la problemele propuse

Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învăţare sau după tipicul acestora.

Bibliografie

1. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004, pg.51-53
2. Sabin Buraga – Proiectarea siturilor Web. Design şi funcţionalitate, Ediţia a II-a, Editura Polirom 2002, pg. 163-195
3. Călin Ioan Acu – Optimizarea paginilor Web, Editura Polirom 2003, pg. 35-73
4. Richard Wagner, R.Allen Wyke – Java Script, Editura Teora 2000, pg.481-495

BIBLIOGRAFIE

1. Călin Ioan Acu – Optimizarea paginilor Web, Editura Polirom 2003
2. Cioata Mihai – ActiveX. Concepte si aplicatii Editura: Polirom 2003
3. Dorin Cârstoiu, Ecaterina Oltean – Introducere în HTML şi XML, Editura Printech Bucureşti 2004
4. Luke Welling şi Laura Thomson, Dezvoltarea aplicaţiilor web cu PHP şi MySQL, Editia a II-a, Editura Teora, 2005
5. Manualul PHP - http://php.net/manual/ro/index.php
6. Mihaela Brut, Sabin Buraga - Prezentari multimedia pe Web. Limbajele XHTML + TIME şi SMIL Editura Polirom 2003
7. McFedries Paul, Trad Voin, Doru Sorin – Crearea paginilor WEB, Editura ALL 2003
8. Negrino Tom, Smith Dori - JavaScript pentru World Wide Web. Ghid de invatare rapida prin imagini, Editura Corint 2004
9. Richard Wagner, R.Allen Wyke – Java Script, Editura Teora 2000
10. Sabin Buraga – Proiectarea siturilor Web. Design şi funcţionalitate, Ediţia a II-a, Editura Polirom 2002
11. T.Gugoiu – HTML prin exemple, Editura Teora 2000
12. http://www.w3schools.com/default.asp

ANEXA 1 – Lucrări de laborator

Sesiunea de lucru de laborator L1

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr.1 şi 2.
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:
- Însuşirea unor noţiuni de bază din domeniul reţelelor de calculatoare
- Însuşirea paşilor necesari într-un ciclu de realizare-vizualizare rezultat, pentru o pagină Web
- Însuşirea regulilor de sintaxă pentru directivele HTML
- Însuşirea modului de definire a culorilor pentru elementele HTML

În cadrul lucrării de laborator numarul *2* se recomandă efectuarea următoarelor tipuri de activităţi:
- Discuţie pe tema noţiunilor introduse în Unitatea de învăţare numărul 1 cu scopul de a clarifica nelămuriri în ceea ce priveşte arhitectura World Wide Web sau Internet
- Efectuarea unui exerciţiu la calculator, asistat de tutore pas cu pas care să urmărească clarificarea paşilor necesari într-un ciclu de realizare-vizualizare rezultat, pentru o pagină WEB
- Discuţie despre structura de bază a unui document HTML (schelet) şi despre reguli ce privesc sintaxa directivelor HTML
- Efectuarea unui exerciţiu la calculator care să urmărească definirea şi utilizarea corectă a culorilor într-un document HTML

Sesiunea de lucru de laborator L2

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr.3 şi 4.
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:
- Definirea şi folosirea titluri şi paragrafe în HTML
- Definirea şi folosirea directivelor HTML pentru formatare logică
- Definirea şi folosirea directivelor HTML pentru formatare fizică
- Definirea şi folosirea listelor în limbajul HTML

În cadrul lucrării de laborator numarul 2 se recomandă efectuarea următoarelor tipuri de activităţi:

- Discuţie pe tema formatării textelor HTML în care se vor clarifica noţiunile însuşite în Unitatea de învăţare numărul 3 legate de formatare textelor cu ajutorul HTML
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea noţiunilor şi tehnicilor de formatare a textelor HTML.
- Discuţie pe tema celor 3 tipuri de liste HTML în care se vor clarifica noţiunile însuşite în Unitatea de învăţare numărul 4 legate de utilizarea listelor HTML
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de definire a listelor in limbajul HTML.

Sesiunea de lucru de laborator L3

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr. 5, şi 6.
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:
- Înţelegerea conceptelor de Hypertext, Hyperlink şi URL
- Definirea şi folosirea legăturilor în cadrul paginilor Web
- Întelegerea formatelor grafice folosite in Web
- Întelegerea modului de adăugare şi utilizare a imaginilor într-o pagina Web

În cadrul lucrării de laborator numarul 3 se recomandă efectuarea următoarelor tipuri de activităţi:

- Discuţie teoretică pe marginea conceptelor de Hypertext, Hyperlink şi URL. Se va urmări clarificarea acestor concepte, şi înţelegerea modului de folosire al acestor concepte în cadrul paginilor WEB
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea noţiunilor şi tehnicilor de utilizare a linkurilor în cadrul paginilor HTML.
- Discuţie asupra tipurilor de formate grafice folosite pentru realizarea paginilor WEB, avantajele şi dezavantajele fiecărui tip de format.
- Discuţie teoretică pe tema utilizării imaginilor în cadrul paginilor WEB cu scopul clarificării noţiunilor din cadrul Unităţii de învăţare numărul 6
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de folosire a imaginilor, integrare, aliniere, dimensionare în pagina HTML

Sesiunea de lucru de laborator L4

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţii de învăţare Nr.7
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:

- Înţelegerea modului de realizare şi de configurare al tabelelor HTML
- Explicarea părţilor componente ale unui tabel HTML şi a modului în care proprietăţile acestora pot fi schimbate
- Prezentarea diferitelor moduri de folosire a tabelelor în pagina HTML

In cadrul lucrării de laborator numarul 4 se recomandă efectuarea următoarelor tipuri de activităţi:

- Explicarea modului de utilizare a tabelelor şi clarificarea noţiunilor prezentate în Unitatea de învăţare numărul 7.
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de definire şi folosire a tabelelor HTML

Sesiunea de lucru de laborator L5

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţii de învăţare Nr. 8.
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:

- Clarificarea conceptului de formular HTML
- Explicarea modului de folosire al fiecărui tip de element din cadrul unui formular HTML
- Înţelegerea modului de folosire al formularelor HTML

În cadrul lucrării de laborator numarul 5 se recomandă efectuarea următoarelor tipuri de activităţi:

- Discuţie asupra modului de utilizare a tabelelor şi clarificarea noţiunilor prezentate în Unitatea de învăţare numărul 8.
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de definire şi folosire a formularelor HTML. Exerciţiul ar trebui sa acopere modul de folosire al fiecărui tip de element din cadrul unui formular HTML

Sesiunea de lucru de laborator L6

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr. 9 şi 10.
Sesiunii de laborator îi sunt alocate 4 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer, precum şi accesul la componente JavaApplets şi/sau ActiveX.

Obiectivele acestei sesiuni de lucru sunt:

- Înţelegerea rolului limbajului JavaScript
- Clarificare noţiunilor de tip script pe partea de client, Evenimente şi tratarea acestora
- Înţelegerea modului de folosire al JavaScript în cadrul unui document HTML
- Clarificarea aspectelor legate de folosirea executabilelor şi componentelor multimedia în pagina WEB

În cadrul lucrării de laborator numarul 6 se recomandă efectuarea următoarelor tipuri de activităţi:

- Discuţie asupra modului de utilizare a tabelelor şi clarificarea noţiunilor prezentate în Unitatea de învăţare numărul 8.
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de definire şi folosire a formularelor HTML. Exerciţiul ar trebui sa acopere modul de folosire al fiecărui tip de element din cadrul unui formular HTML
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de folosire al diferitelor tipuri de componente multimedia şi executabile în cadrul unei pagini WEB

Sesiunea de lucru de laborator L7

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr. 11 şi 12
Sesiunii de laborator îi sunt alocate 2 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Notepad++, sau a unui editor specializat PHP, a unui server web cu extensia PHP instalată, a unui server de baze de date MySQL şi a unui program de navigare internet: Internet Explorer sau FireFox.

Obiectivele acestei sesiuni de lucru sunt:
- Clarificarea noţiunilor legate de limbajul PHP precum variabile, constante, sintaxă, funcţii şi obiecte.
- Includerea de cod PHP în cadrul paginilor HTML.
- Familiarizarea cu noţiunile de bază legate de limbajul SQL.
- Utilizarea informaţiilor din cadrul unei baze de date în cadrul unei pagini web.

În cadrul lucrării de laborator numarul 7 se recomandă efectuarea următoarelor tipuri de activităţi:

- Explicarea noţiunilor de bază legate de limbajul PHP.
- Efectuarea unui exerciţiu la calculator care să pună în practică noţiunile învăţate despre limbajul PHP.
- Discuţie legată de sistemele de baze de date şi despre limbajul SQL.
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea tehnicilor de utilizare a bazelor de date în cadrul paginilor web.

Sesiunea de lucru de laborator L8

Setul de lucrări are ca structură aprofundarea şi consolidarea cunoştinţelor dobândite în cadrul Unităţilor de învăţare Nr. 13 şi 14
Sesiunii de laborator îi sunt alocate 2 ore. Se recomandă a se efectua sub supravegherea cadrului didactic.

Pentru efectuarea laboratorului este necesar utilizarea unui editor de texte, spre exemplu: Windows Notepad, şi a unui program de navigare internet: Internet Explorer

Obiectivele acestei sesiuni de lucru sunt:

- Clarificarea noţiunilor legate de XML precum definiţii, sintaxă, caracteristici
- Clarificarea modului de folosire al XML în cadrul unei pagini WEB, şi a conceptului de separare a datelor utile de interfaţa grafică
- Clarificarea noţiunii de pagină de stil.
- Evidenţierea avantajelor folosirii paginilor de stil

În cadrul lucrării de laborator numărul 8 se recomandă efectuarea următoarelor tipuri de activităţi:

- Discuţie asupra conceptelor de bază XML şi clarificarea noţiunilor de bază
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea noţiunilor fundamentale ale limbajului XML.
- Discuţie asupra modului de utilizare al paginilor de stil în cadrul unei pagini WEB
- Efectuarea unui exerciţiu la calculator care să urmărească însuşirea modului de folosire al paginilor de stil

Felicitări pentru parcurgerea acestui modul! Acesta reprezintă un prim pas in vastul domeniu al Internetului care este într-o continuă evoluţie. Si acum va dorim succes în realizarea paginilor de WEB pe care le veţi realiza!

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI
MINISTERUL MUNCII, FAMILIEI ŞI PROTECŢIEI SOCIALE
AMPOSDRU

Fondul Social European
POSDRU 2007-2013

Instrumente Structurale
2007-2013

MINISTERUL EDUCAŢIEI CERCETĂRII TINERETULUI ŞI SPORTULUI
OIPOSDRU

MINISTERUL EDUCAŢIEI CERCETĂRII TINERETULUI ŞI SPORTULUI
UMPFE

Proiect cofinanţat din Fondul Social European prin Programul Operaţional Sectorial Dezvoltarea Resurselor Umane 2007-2013
Investeşte în oameni!

Formarea profesională a cadrelor didactice din învăţământul preuniversitar pentru noi oportunităţi de dezvoltare în carieră

MERGI MAI DEPARTE...

O NOUĂ SPECIALIZARE, ŞANSA TA!

Unitatea de Management al Proiectelor cu Finanţare Externă

Str. Spiru Haret nr. 12, Etaj 2,
Sector 1, Cod poştal 010176,
Bucureşti

Tel: 021 305 59 99
Fax: 021 305 59 89

http://conversii.pmu.ro
e-mail: conversii@pmu.ro

ISBN 978-606-515-130-7