

Cuprins

CUPRINS	1
INTRODUCERE	2
1. INTRODUCEREA CONCEPTULUI DE INTRANET	3
1.1. Definirea unor noțiuni de bază	3
1.2. Ce este INTRANET-ul?	4
2. ELEMENTE INTRANET	6
2.1. Resurse existente într-un INTRANET	6
2.2. Limbaje de programare	8
3. ELEMENTE HTML	12
3.1. Elemente de structură	12
3.2. Elemente de structură logică	13
3.3. Elemente de formatare a textului	16
3.4. Liste	18
3.5. Imagini si Sunet	19
3.6. Formulare	21
3.7. Introducerea elementelor de Java si JavaScript	25
3.8. Tabele	26
3.9. Frame-uri	27
3.10. Comentarii	28
4. COMMON GATEWAY INTERFACE (CGI)	29
4.1. Ce este CGI?	29
4.2. Elemente de bază	30
4.3. Instalarea și rularea programelor CGI	32
4.4. Preluarea intrărilor de la un Browser	34
4.5. Un program CGI simplu	41
5. APLICAȚII INTRANET	42
Căutarea studentilor (http://www.scs.ubbcluj.ro/scs/cautare/index.html)	42
Citirea notelor (http://www.scs.ubbcluj.ro/~sanda)	45
ANEXA A	65
ANEXA B	70
ANEXA C	71
BIBLIOGRAFIE	72

Introducere

Lucrarea de față este structurată în două părți: o parte teoretică și o parte practică.

Partea teoretică este destinată prezentării conceptului de **INTRANET**, iar partea practică prezintă câteva aplicații specifice INTRANET.

În *primul capitol* se face o introducere a conceptului de INTRANET și a altor concepte care se află în legatura cu el.

Al *doilea capitol* descrie mai pe larg conceptul de INTRANET introducând câteva componente acestuia.

Capitolul trei face o scurtă prezentare a elementelor de bază HTML. Partea cea mai consistentă o reprezintă introducerea elementelor care descriu formularele HTML.

Capitolul patru ne introduce în lumea Common Gateway Interface.

Partea practică este cuprinsă în *capitolul cinci* al acestei lucrări și constă în prezentarea a două aplicații specifice INTRANET. Prima aplicație este o mașina de căutarea a studenților, aplicație care rulează deja pe pagina studentilor (<http://www.scs.ubbcluj.ro>). A doua aplicație este o aplicație care întreține o bază de date cu notele obținute de studenți la examene și care permite fiecărui student sa-și vadă doar propriile note. Aceasta aplicație rulează pe pagina studenților de la Informatică la Distanță (<http://math.ubbcluj.ro/~sanda>).

Anexele conțin informații utile în scrierea unei pagini HTML sau a unui CGI.

1. Introducerea conceptului de INTRANET

INTRANET-ul [2] reprezintă o rețea concepută pentru un scop intern, care utilizează tehnologii INTERNET.

1.1. Definirea unor noțiuni de bază

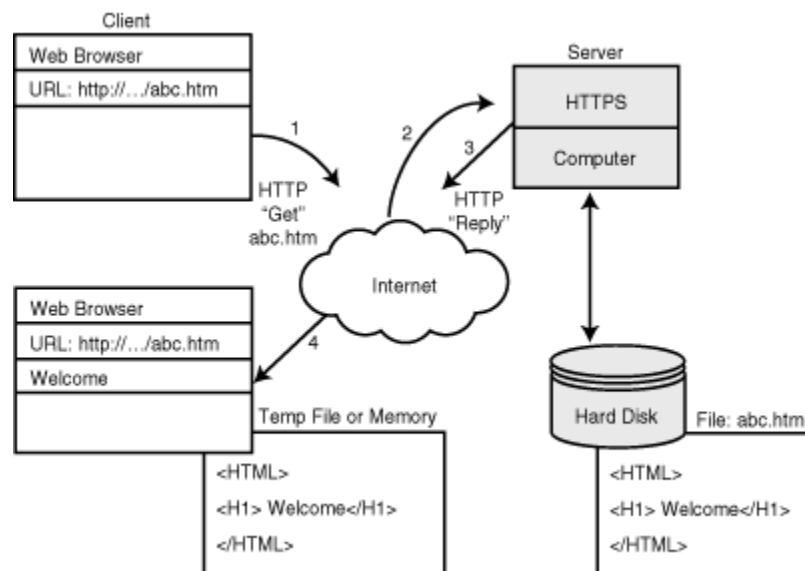
INTERNET-ul poate fi definit ca fiind format din toate calculatoarele și rețelele interconectate între ele folosind ca și protocol de comunicare TCP/IP (Transmission Control Protocol/Internet Protocol).

TCP/IP reprezintă un potocol standard de comunicare folosit în INTERNET. Protocol de comunicare este un limbaj prin care oricare două calculatoare pot comunica între ele.

WWW (World Wide Web) reprezintă totalitatea paginilor Web legate între ele prin hyperlink-uri.

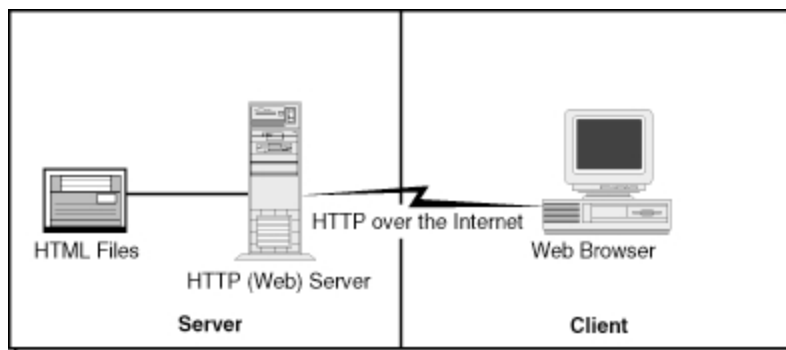
Pagina Web este un document text scris folosind limbajul HTML (Hyper Text Markup Language) și care poate fi interpretat folosind un browser Web.

Browser Web este un utilitar (program) care la cererea unei pagini Web face o cerere, folosind protocolul HTTP (Hypertext Transfer Protocol), către un server Web pentru pagina solicitată și interpretează respectiva pagină.



Serverul Web [3] așteaptă cereri de la browser-e și le trimite pagini folosind protocolul HTTP.

HTTP (HyperText Transfer Protocol) este un protocol care permite comunicarea în WWW.



1.2. Ce este INTRANET-ul?

INTRANET-ul [4] reprezintă o rețea internă, care unește mai mulți utilizatori folosind tehnologii INTERNET. Spre deosebire de INTERNET, o rețea INTRANET are limite teritoriale, își stabilește sectoare cu acces controlat în care utilizatorii pot comunica și interacționa. Construite la nivel de organizație, aceste rețele se află în World Wide Web, permițând utilizatorilor autorizați să comunice în timp real.

Rețelele INTRANET pot coexista cu alte rețele locale. Multe companii au *mainframe*-uri, rețele Novell, minicomputere și diferite baze de date integrate într-o rețea INTRANET. În cadrul acestor rețele se folosesc o largă varietate de utilitare. Scripturile **CGI** (*Common Gateway Interface*) sunt deseori folosite pentru a accesa bazele de date din INTRANET. Tot pentru accesarea bazelor de date din INTRANET se mai folosește și limbajul de programare **Java**.

Rețelele INTRANET pot conține diferite servere și servicii. Serverele INTRANET conțin de obicei informații confidențiale ale proprietarului rețelei respective. Principalul scop al rețelelor INTRANET este de a face munca celor care folosesc aceasta rețea mai ușoară.

Utilizatorii INTRANET-ului sunt persoane care utilizează o pagină Web internă. Folosind INTRANET-ul, aceste persoane pot schimba ușor informații, chiar dacă acest schimb se realizează doar la nivel de fișier text. Costul unei astfel de informări este mult mai redus decât tipărirea și distribuirea informației respective. Folosind un server de Web intern se poate realiza chiar și un anumit nivel de interacțiune, iar browserele Web sunt ușor de folosit și nu sunt dependente de sistemul de operare.

Ultimele cercetări arată că există mai multe site-uri interne de Web decât cele externe. Creșterea explozivă a site-urilor interne de Web se datorează mai multor motive. Unul dintre aceste motive este faptul că protocoalele sunt deschise, ceea ce înseamnă că oricine poate dezvolta aplicații pentru ele. Un alt motiv ar fi faptul că Web-ul este *cross platform* și *client/server*, două tehnologii după care s-a așteptat mult timp.

Aceste site-uri interne de Web sau *INTRANET*-uri au diferite utilități, scopuri și sunt mai eficiente decât site-urile INTERNET.

Există un număr nelimitat de probleme care pot fi soluționate folosind un server INTRANET. Aplicațiile care soluționează aceste probleme se pot grupa în trei categorii:

- ❑ de informare (*publish applications*);
- ❑ interactive (*interactive applications*);
- ❑ de comunicare (*discussion applications*).

Aplicații de informare

Aplicațiile de informare sunt documente HTML care permit unei persoane să se adreseze unui grup. Aceste documente pot fi simple pagini statice HTML, fișiere *PostScript* sau imagini. Orice document care conține o informație poate fi convertit în format electronic și pus pe un server Web. Publicarea electronică oferă posibilitatea de a lega paginile între ele folosind hypertext-uri.

Informarea se realizează astfel rapid și cu un cost scăzut.

Aplicații interactive

Cu ajutorul *aplicațiilor interactive* se poate realiza comunicarea între două persoane sau două programe. Ele ajută la comunicarea între sistemele existente. Realizează căutări în baze de date, adaugă informații, întocmesc rapoarte sau fac alte acțiuni. Crearea unei aplicații interactive se realizează deseori folosind programe **CGI**, programare la nivel de server și **Java**.

Când se crează aplicații interactive se dorește ca interfața cu utilizatorul să fie cât mai prietenoasă și ușor de utilizat.

Aplicații de comunicare

Chat room-urile, *newsgroup*-urile, listele de *mail*, grupurile de discuții pot fi foarte productive. Acestea permit diferitelor grupuri să discute idei noi sau probleme curente fără ca persoanele din grupurile respective să se afle în aceeași zonă geografică.

2. Elemente INTRANET

HTML este parte integrantă dintr-o rețea INTRANET datorită faptului că este un instrument de distribuire a documentelor de informare statice (pagini HTML). Acest instrument nu permite însă o interacțiune între client și server. Pentru aceasta există o mulțime de utilitare care extind capacitățile HTML, permițând manipularea datelor, generarea dinamică a paginilor și un nivel ridicat de interactivitate.

Utilitarele sau aplicațiile care rulează într-o rețea INTRANET trebuie să aibă următoarele proprietăți:

- **Portabilitate**
- **Scalabilitate**
- **Robustete**
- **Usor-de-folosit**
- **Complexitate**

Acești factori trebuie luați în considerare atât la scrierea unei aplicații specifice INTRANET cât și la construirea unei astfel de rețele.

2.1. Resurse existente într-un INTRANET

Server-Side Includes (SSI)

Folosirea unui Server-Side Includes (SSI) reprezintă cea mai ușoară metodă de a introduce crearea semi-dinamică a paginilor HTML în INTRANET. O analiză a SSI este ilustrată în tabelul următor.

	<i>Scăzut</i>	<i>Mediu</i>	<i>Înalt</i>
Funcționalitate	*		
Cerințe tehnice	*		
Ciclu de dezvoltare	*		
Pericol	*		
Încărcarea CPU			*

Application Programming Interface (API)

API reprezintă o altă modalitate de a extinde funcționalitatea unei pagini Web. Tabelul de mai jos ilustrează beneficiile folosirii API.

	<i>Scăzut</i>	<i>Mediu</i>	<i>Înalt</i>	<i>Foarte înalt</i>
Funcționalitate			*	
Cerințe tehnice			*	
Ciclu de dezvoltare			*	
Pericol				*
Încărcarea CPU	*			

Ca și în cazul tehnologiei CGI, implementarea programelor API se face folosind unele limbaje standarde de programare (ex C sau C++), extinzând semnificativ capacitățile serverului.

Plug-In-uri

Plug-in-urile permit realizarea unor taskuri mult mai ușor decât ne putem noi imagina. Aceste taskuri pot fi: rularea diferitelor formate audio sau viziualizarea diferitelor formate de documente. Plug-in-urile sunt doar niște extensii care lucrează cu tipuri de fișiere noi sau nonstandard.

Un exemplu ar fi *Real Audio plug-in*.

Common Gateway Interface (CGI)

CGI (Common Gateway Interface) [4] reprezintă un standard de realizare a legăturii dintre formularele (din partea client) cu serverele HTTP și cu alte aplicații server. Tabelul de mai jos face o analiză a avantajelor folosirii CGI-urilor. Spre deosebire de documentele HTML standard, care sunt statice (nu pot fi schimbate), o aplicație CGI va permite generarea unui fișier HTML dinamic care poate manipula date dintr-o bază de date sau poate realiza alte operații complexe.

	Scăzut	Mediu	Înalt
Funcționalitate			*
Cerințe tehnice			*
Ciclu de dezvoltare		*	
Pericol			*
Încărcarea CPU			*

Manipularea datelor text

Se realiza mai mult sub UNIX, însă acum se realizeaza și sub Windows NT.

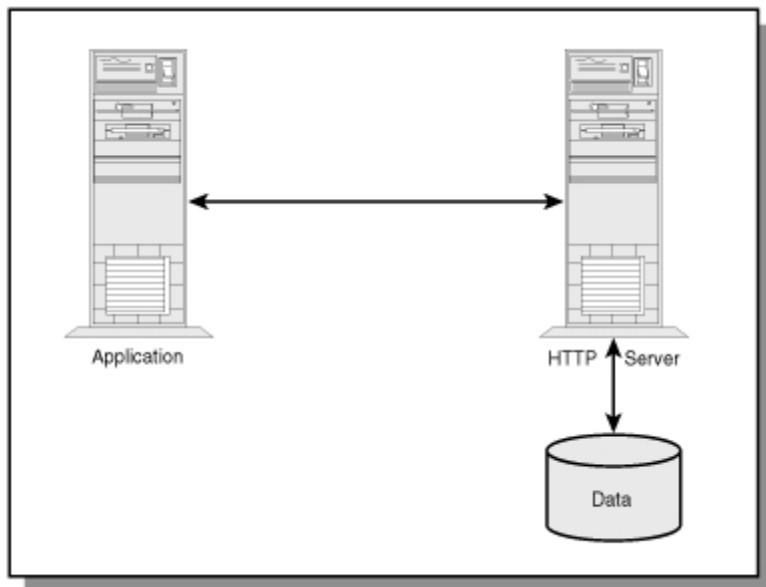
Lucru cu date nontextuale

Este un domeniu în care platformele Windows excelează. Acest domeniu cuprinde editare, calcul tabelar și manipularea bazelor de date.

Interacțiunea cu bazele de date

Urmatoarea listă cuprinde limbajele de programare cele mai folosite pentru lucru cu baze de date.

- ✓ C++
- ✓ C
- ✓ PL/SQL
- ✓ Perl
- ✓ Visual Basic
- ✓ Java



Opțiunile de întreținere a bazelor de date sunt

- *La nivel înalt*
 - Oracle
 - Sybase
- *La nivel mediu*
 - Microsoft SQL Server
 - Microsoft Access
 - Paradox
- *La nivel scăzut*
 - Fișiere text

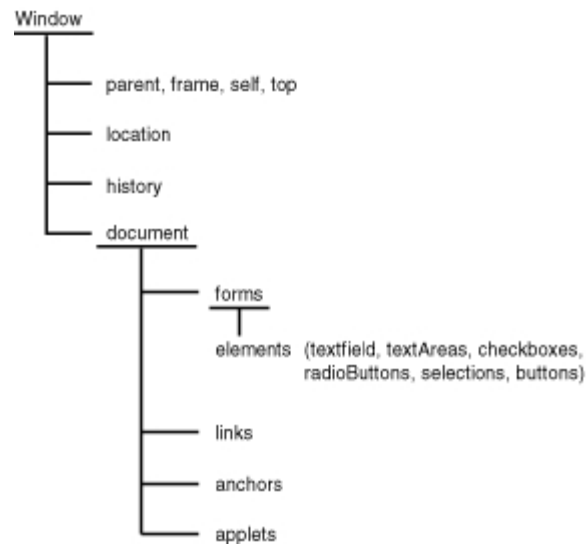
2.2. Limbaje de programare

Pentru scrierea aplicațiilor INTRANET se pot folosi mai multe limbaje de programare.

JavaScript

JavaScript este limbaj scriptual compact orientat obiect. Se folosește pentru crearea unor pagini HTML “îstete” atât în INTERNET cât și în INTRANET.

JavaScript are acces la elementele ale browserului sau elementele dintr-un formular.



Prin combinarea codului JavaScript cu formularele HTML se pot face tot felul de preprocesări. Datorită faptului că instrucțiunile JavaScript sunt incluse în documentul HTML, acestea sunt interpretate de către browser. JavaScript este ușor de implementat și de întreținut.

Una dintre cele mai valoroase proprietăți ale limbajului JavaScript este că poate “simți” unele acțiuni ale utilizatorului (ex: *mouse click*, introducerea datelor în formulare sau navigarea în pagină).

Se pot scrie funcții JavaScript care să verifice, spre exemplu, validitatea datelor introduse în câmpurile unui formular. JavaScript se mai poate folosi și pentru a executa unele acțiuni (ex: play-erea unui fișier audio, executarea unui applet sau comunicarea cu un plug-in) ca răspuns la unele acțiuni ale utilizatorului.

JavaScript suportă majoritatea sintaxei expresiilor din Java și principalele structuri de control.

Ca descendent al unor limbaje ca HyperTalk și dBASE, JavaScript este accesibil unei mai largi audiențe decât fratele lui mai mare, Java. Următorul tabel ne prezintă diferențele dintre JavaScript și Java.

<i>JavaScript</i>	<i>Java</i>
Interpretat de client (browser, plug-in, etc)	Compilat pe server înainte de a fi executat de client
Cod integrat în HTML	Cod este scris într-un fișier separat, însă este apelat din HTML
Generează cod obiectual care folosește obiecte predefinite și extensibile, însă nu folosește clase și moștenire	Obiectele applet construite din obiecte predefinite și extensibile
Limbaj slab tipizat	Limbaj puternic tipizat
Legare dinamică; referință obiectuală la rulare	Legare statică; referința obiectelor se face la compilare

Nu are acces la HD	Nu are acces HD, deși browserul poate rula aplicații native
--------------------	---

VBScript

Microsoft Visual Basic Scripting Edition (VBScript) este un subset al tradiționalului limbaj de programare MS Visual Basic. Deși VBScript tocmai și-a făcut intrarea pe scena dezvoltării programelor pentru INTRANET, acest limbaj a devenit destul de folosit. Avantajele folosirii VBScript sunt numeroase. Programatorii stau în cumpănă între a îmbrățișa VBScript sau limbajul scriptual al Sun Microsystems, JavaScript. Ca și limbajul scriptual JavaScript, VBScript este un limbaj interpretativ, instrucțiunile sale fiind citite în memorie în momentul în care pagina Web care le conține este transferată clientului browser. Codul VBScript este scris direct în pagina HTML.

```
<SCRIPT LANGUAGE="VBS">

Sub BtnAlert_OnClick

MsgBox "Mesaj"

End Sub

</SCRIPT>
```

Când browserul întâlnește un tag <SCRIPT>, apelează (în cazul în care limbajul este VBScript) motorul de compilare a codului VBScript.

Una dintre cele mai impresionante trăsături ale VBScript este introducerea elementelor OLE în dezvoltarea aplicațiilor INTRANET. Având în vedere această capacitate, VBScript va putea manipula atât browser-ul și celelalte aplicații *client-side* bazate pe evenimente predeterminate cât și alte variabile ale mașinii gazdă. În același timp, VBScript are capacitatea de a lucra cu proprietăți și metode ale controalelor ActiveX (.OCX-uri) și applete Java, ceea ce extinde funcționalitatea în INTRANET.

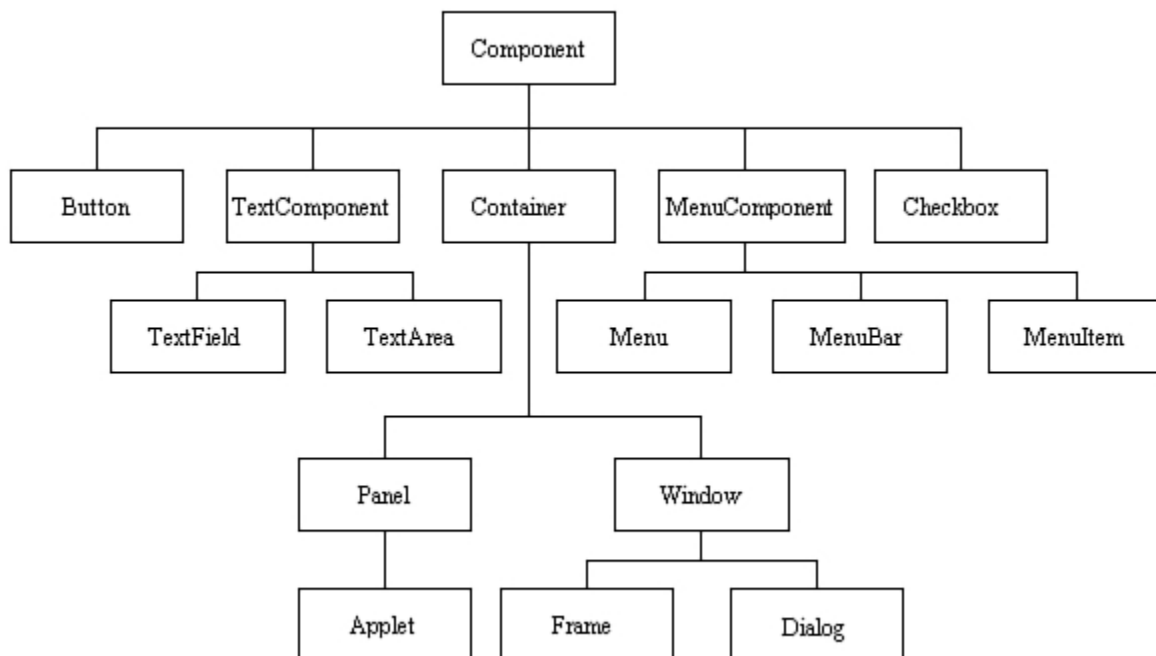
VRML

La prima conferință World Wide Web care s-a ținut la Geneva, în Elveția, în primavara anului 1994, Tim Berners-Lee și Dave Raggett au discutat noua direcție a modelărilor tridimensionale folosind ca mediu de transmitere INTERNET-ul. În timpul acestor discuții s-a adoptat termenul de "Virtual Reality Modeling Language", iar dezvoltarea VRML a început la scurt timp după acest eveniment, devenind un instrument puternic în ciuda faptului că este greu de folosit. VRML reprezintă un mediu pentru specificarea poligoanelor și a peisajelor tridimensionale și un utilitar principal de transmitere online a conținutului tridimensional.

Versiunea 1.0 a limbajului are trei scopuri principale: primul este faptul că, asemeni HTML și altor tehnologii de rețea, este independent de platformă; în al doilea rând este extensibil, iar în al treilea rând este optimizat pentru transmiterea datelor în conexiunile cu lungime de banda scăzută, lucru care este foarte important dacă se dorește transmiterea datelor VRML la mai multe stații dintr-o rețea INTRANET.

Java

Este un mediu de dezvoltare a aplicațiilor orientate obiecte. Dintre toate limbajele în care se scriu aplicații pentru INTERNET sau INTRANET, Java este cel mai revoluționar. El permite scrierea de aplicații functionale care rulează atât pe Web prin protocolul HTTP (aplicații *applet*) cât și local pe mașina client (aplicații *stand-alone*). În anii în care Web-ul a luat amploare era necesară existența unor utilitare care să faciliteze dezvoltarea unor sisteme interactive și care să conțină și elemente multimedia. În acest scop, programatorii de la Sun Microsystems au scris limbajul Java, care este un mediu de programare orientat-obiect, independent de platforma și care permite scrierea unor sisteme sau aplicații INTRANET robuste, sigure și performante. Principalele elemente folosite într-un applet sunt:



Cea mai remarcabilă trăsătură a limbajului Java este faptul că este independent de platformă. Așadar, orice applet utilizat într-o rețea INTRANET va rula pe orice platformă care are un interpretor Java. Interpretorul poate fi unul inclus în browser, un plug-in sau o altă aplicație. Astăzi, interpretoarele Java sunt disponibile pentru majoritatea platformelor și a sistemelor de operare.

O altă trăsătură importantă a acestui limbaj o reprezintă robustețea sa. Pentru tehnologii care trebuie să funcționeze pe mai multe platforme și mai multe sisteme de operare această proprietate este esențială.

Spre deosebire de Perl, scripturile shell sau JavaScript, Java este un limbaj orientat-obiect care permite moștenirea și re folosirea de cod. Fiind un instrument de înaltă performanță, Java suportă multithreading și folosirea de cod nativ.

Un alt element foarte important, prin care se deosebeste de C și C++ este modul de tratare a memoriei și absența pointerilor.

3. Elemente HTML

Pentru aplicațiile de informare este necesară scrierea de pagini HTML. În acest scop vom prezenta în continuare câteva elemente de bază HTML.

3.1. Elemente de structură

Aceste taguri definesc structura unui document HTML.

<html>..</html>

Scop: Conține întregul document HTML.

Versiune: HTML 2.0.

Detalii: Aceste taguri sunt folosite la începutul și la sfârșitul codului HTML dintr-o pagina.

<head>..</head>

Scop: Conține alte tag-uri care descriu particularități ale documentului.

Versiune: HTML 2.0.

Detalii: Aceste tag-uri trebuie așezate întotdeauna înaintea tag-urilor <BODY> sau <FRAMESET>.

<title>..</title>

Scop: Conține titlul documentului.

Versiune: HTML 2.0.

Detalii: Acest titlu se afișează de obicei în bara de titlu al browser-ului Web.

<base>

Scop: Conține informații referitoare la documentul curent; trebuie situat între tagurile <head>..</head>.

Versiune: HTML 2.0, atributul TARGET a apărut odată cu Frame-urile.

Atribute:

HREF="URL"	Specifică URL-ul absolut al documentului curent.
TARGET="nume_fereastră"	Specifică frame-ul în care se vor deschide link-urile din documentul curent.

Detalii: Este util în cazul unei revizuri a documentului.

<body>..</body>

Scop: Conține partea vizibilă a unui document HTML.

Versiune: HTML 3.0, atributul BACKGROUND este din versiunea HTML 3.0; attributele ALINK, BGCOLOR, LINK, TEXT, și VLINK sunt o extensie Netscape a versiunii HTML 3.0; atributul BGPROPERTIES este o extensie a Internet Explorer-ului.

Atribute:

ALINK="#rrggbb"	Specifică culoarea unui link activ.
BACKGROUND="URL"	Specifică imaginea de background.
BGCOLOR="#rrggbb"	Specifică culoarea de background dacă nu există o imagine de background.

BGPROPERTIES=FIXED	Specifică o proprietate a background-ului. Valoarea FIXED este singura care există și înseamnă ca background-ul nu va defila odata cu pagina.
LINK="#rrggbb"	Specifică culoarea unui link.
TEXT="#rrggbb"	Specifică culoarea textului.
VLINK="#rrggbb"	Specifică culoarea unui link care a fost vizitat.

Detalii: Acest tag trebuie așezat imediat după tag-ul </HEAD>. "#rrggbb" este o valoarea RGB (Red-Green-Blue) descrisă în hexazecimal.

3.2. Elemente de structură logică

Aceste taguri definesc structură logică a paginii.

<A>..

Scop: Specifică o ancoră.

Versiune: HTML 2.0, atributul TARGET a apărut odata cu frame-urile .

Attribute:

HREF="URL"	Specifică o legatură (link) către un alt document. Documentul spre care se face legatura este specificat printr-o adresă de tip URL.
NAME="nume_ancora"	Creează, în documentul curent, o ancoră la care se poate face referire prin specificarea unui URL care conține locația paginii și numele ancorei. Saltul se va face exact în locul în care s-a creat ancora.
TARGET="nume_fereastră"	Specifică frame-ul în care se va încărca noul URL.

Detalii: Există și alte attribute, însă, acestea sunt foarte puțin folosite. Textul care apare între aceste taguri apare pe ecran, și, de obicei descrie pagina la care face salt.

<address>..</address>

Scop: Conține orice fel de adresă.

Versiune: HTML 2.0.

Detalii: Se afișează, de obicei, cu caractere italice și poate fi indentat altfel decât un text obișnuit.

<blockquote>..</blockquote> și <bq>..</bq>

Scop: Conține un paragraf preluat din altă sursă (citat).

Versiune: HTML 2.0, <bq> apare în versiunea HTML 3.0.

Detalii: Se afișează cu caractere italice și se lasă margini mai mari la dreapta și la stanga textului. Ambele taguri <blockquote> și <bq> sunt recunoscute în versiunea HTML 3.0.

**
**

Scop: Specifică o linie nouă.

Versiune: HTML 2.0.

Detalii: Nu are tag de închidere.

<cite>..</cite>

Scop: Conține un citat.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere italice. <cite> se folosește pentru citate scurte, iar <blockquote> pentru citate lungi.

<code>..</code>

Scop: Conține o parte din codul unui program.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere monospațiate (ex. Courier).

..

Scop: Conține cuvinte care trebuie subliniate (întărite).

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere italice.

<fn>..</fn>

Scop: Conține o notă de subsol.

Versiune: HTML 3.0.

Attribute:

ID="identificator"	Pune o ancoră asociată notei de subsol ca să se poată face salt la ea.
--------------------	--

Detalii: Specificațiile HTML 3.0 recomandă ca aceste note să fie așezate sub forma de pop-up-uri.

<hr>

Scop: Specifică o linie orizontală.

Versiune: HTML 2.0, attributele ALIGN, NOSHADE, SIZE, și WIDTH sunt extensii ale Netscape-ului, versiunea HTML 2.0; atributul SRC apare în versiunea HTML 3.0.

Attribute:

ALIGN=LEFT RIGHT CENTER	Specifică alinierea liniei în cazul în care se folosește atributul WIDTH.
NOSHADE	Desenează o linie fără umbră.
SIZE=n	Specifică grosimea liniei.
SRC="URL"	Specifică URL-ul imaginii folosite ca și linie.
WIDTH=n n%	Specifică lungimea în pixeli sau în procente a liniei.

Detalii: Nu are tag de închidere.

<kdb>..</kdb>

Scop: Conține textul introdus de la tastatura de un utilizator.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere monospațiate (ex: Courier).

<marquee>..</marquee>

Scop: Conține un marquee.

Versiune: Extensie Internet Explorer.

Attribute:

ALIGN= BOTTOM MIDDLE TOP	Specifică alinierea textului din marquee.
BEHAVIOR=ALTERNATIVE SCROLL SLIDE	Specifică modul de mișcare al marquee-ului.
DIRECTION=LEFT RIGHT	Specifică partea ecranului prin care intră textul .
HEIGHT=n n%	Specifică înălțimea în pixeli sau în procente a marquee-ului.
HSPACE=n	Specifică marginea orizontală (în pixeli).
LOOP=n INFINITE	Specifică numărul de rotiri al marquee-ului.
SCROLLAMOUNT=n	Specifică numărul de pixeli cu care se deplasează marquee-ul la o redesenare.
SCROLLDELAY=n	Specifică numărul de milisecunde între două redesenări.
WIDTH=n n%	Specifică lățimea marquee-ului (în pixeli sau în procente din lățimea ecranului).
VSPACE=n	Specifică marginea virtuală în pixeli.

Detalii: În Java se folosesc pachetele tickertape.java or ticker.java.

<note>..</note>

Scop: Conține o notă importantă pentru cititor.

Versiune: HTML 3.0.

Attribute :

CLASS=CAUTION NOTE WARNING	Specifică o clasificare a notei .
SRC="URL"	Specifică o image care se asociază cu nota.

Detalii: În versiunea HTML 3.0 atributul CLASS are ca valoare implicită valoarea WARNING.

<p>..</p>

Scop: Conține un paragraf.

Versiune: HTML 2.0, Atributele sunt din versiunea HTML 3.0.

Attribute:

ALIGN= CENTER JUSTIFY LEFT RIGHT	Specifică alinierea paragrafului.
NOWRAP	Specifică faptul că trecerea la linie nouă nu se va face în mod automat ci numai la întâlnirea tagului .

Detalii: În versiunea HTML 1.0 nu era necesar tagul de închidere </p>. Se recomandă închiderea acestui tag.

<pre>..</pre>

Scop: Conține un text preformatat.

Versiune: HTML 2.0.

Detalii: Conține un text care nu trebuie formatat automat de către browser. Textul este afișat cu fonturi monospațiate (ex. Courier).

<q>..</q>****

Scop: Conține un citat scurt.

Versiune: HTML 3.0.

Detalii: Pentru citate lungi se folosește <blockquote>.

<samp>..</samp>****

Scop: Conține un grup de caractere.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere monospațiate (ex. Courier).

..****

Scop: Conține un text care trebuie scos în evidență.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere bold-ate.

_{..**}**

Scop: Conține un subscript.

Versiune: HTML 3.0.

^{..**}**

Scop: Conține un superscript.

Versiune: HTML 3.0.

<var>..</var>****

Scop: Conține o variabilă.

Versiune: HTML 2.0.

Detalii: Se afișează cu caractere italice.

3.3. Elemente de formatare a textului

..****

Scop: Conține un text bold-at.

Versiune: HTML 2.0.

Detalii: Dacă browserul nu recunoaște caracterele bold-ate, atunci se va folosi o altă reprezentare.

<big>..</big>****

Scop: Conține un text care se va afișa cu caractere mari.

Versiune: HTML 3.0.

Detalii: Tagul are acest atribut, dar numai pentru Netscape.

<blink>..</blink>****

Scop: Face un text să pâlpâie.

Versiune: Extensie Netscape a versiunii HTML 2.0.

Detalii: Acest tag nu se recomandă a fi folosit.

<center>..</center>

Scops: Conține obiecte care vor fi așezate centrat.

Versiune: Extensie Netscape a versiunii HTML 2.0.

Detalii: Se recomandă folosirea atributului ALIGN=center a altor taguri ca substituent al acestui tag.

Scop: Schimbă dimensiunea fontului.

Versiune: Extensie Netscape a versiunii HTML 2.0.

Attribute:

SIZE=n	Specifică dimensiunea fontului. Se accepta valori relative: -n sau +n.
---------------	--

Detalii: Intervalul dimensiunilor este de la 1 la 7. Dimensiunea implicită este 3.

<i>..</i>

Scop: Conține un text scris cu caractere italice.

Versiune: HTML 2.0.

Detalii: Dacă browserul nu recunoaște caracterele italice, se va găsi o altă reprezentare.

<nobr>..</nobr>

Scop: Conține un text care se va scrie pe o aceeași linie.

Versiune: Extensie Netscape a versiunii HTML 2.0.

<small>..</small>

Scop: Conține un text care va fi scris cu caractere mici.

Versiune: HTML 3.0.

Detalii: Tagul are acest atribut, dar numai pentru Netscape.

<tt>..</tt>

Scop: Conține text monospațiat.

Versiune: HTML 2.0.

Detalii: Dacă browserul nu recunoaște caracterele monospațiate, se va găsi o altă reprezentare.

<wbr>

Scop: Specifică locul unde se poate despărți un cuvânt.

Versiune: Extensie Netscape a versiunii HTML 2.0.

Detalii: Se folosește numai în cadrul elementelor <NOBR>.

<h1>..</h1>,<h2>..</h2>,<h3>..</h3>,<h4>..</h4>,<h5>..</h5>,<h6>..</h6>

Scop: Conține un text care reprezintă un titlu.

Versiune: HTML 2.0.

Detalii: <h1> reprezintă cea mai mare dimensiune de titlu. Acestea scad progresiv până la <h6>.

3.4. Liste

HTML suportă mai multe tipuri de liste. Se pot folosi de la simple liste ordonate pana la construirea unui întreg dicționar.

<dl>..</dl>

Scop: Conține o definiție de listă.

Versiune: HTML 2.0.

Detalii: O definiție de listă conține termeni care se specifică folosind tagul <DT>, iar definițiile se specifică în tagul <DD>.

<dd>

Scop: Dă o definiție.

Versiune: HTML 2.0.

Detalii: Urmează după tagul <DT> și un termen. <DD> nu are tag de închidere.

<dt>

Scop: Identifică un termen care va fi definit.

Versiune: HTML 2.0.

Detalii: Este urmat de tagul <DD> și de un termen de definit. Nu are tag de închidere.

<lh>..</lh>

Scop: Specifică un titlu de listă.

Versiune: HTML 3.0.

Detalii: Se așează după unul dintre tagurile sau <dl> și înainte de tagul .

Scop: Specifică începutul unui element al listei.

Versiune: HTML 2.0, Atributele TYPE și VALOARE sunt extensii ale Netscape în versiunea HTML 2.0, iar atributele SKIP și SRC apar în versiunea HTML 3.0.

Atribute:

SKIP=n	Specifică într-o lista ordonată ca se va sări în numerotare peste o secvență de n numere.
USRC="URL"	Specifică o imagine care va fi folosită ca marcator de element în listă.
TYPE=xxx	Specifică o schimbare în tipul listei. xxx poate reprezenta orice alegere din TYPE. Este atribut al tagurilor și .
VALOARE=n	Specifică elementul de început al unei liste ordonate. De exemplu, se poate începe cu 4 în loc de 1.

Detalii: nu are tag de închidere.

..

Scop: Conține o lista ordonată.

Versiune: HTML 2.0, Atributele START și TYPE sunt extensii Netscape în versiunea HTML 2.0, iar atributele CONTINUE și SEQNUM apar în versiunea HTML 3.0.

Attribute:

CONTINUE	Specifică numerotarea listei de unde a rămas ultima listă.
SEQNUM=n	Specifică secvența cu care se începe numerotarea unei liste ordonate. De exemplu, numerotarea unei liste poate să înceapă cu 4 în loc de 1.
START=n	Specifică secvența de început în numerotarea listelor ordonate.
TYPE=A a I i 1	Specifică tipul numerotărilor folosiți. 1 este implicit.

Detalii: O lista ordoată se va afișa implicit ca o lista numerotată. Tagurile sunt folosite pentru a specifica unde începe un element al listei.

..

Scop: Conține o lista neordonată.

Versiune: HTML 2.0, Atributul TYPE este o extensie Netscape a versiunii HTML 2.0.

Atributele PLAIN, SRC și WRAP apar în versiunea HTML 3.0.

Attribute:

PLAIN	Specifică lipsa elementelor de marcare a elementelor.
SRC="URL"	Specifică o imagine ca fiind marcator al unui element al listei.
TYPE=circle disc square	Specifică tipul de marcator folosit.
WRAP = horiz vert	Specifică o listă multicoloană. Valorile horiz și vert spun browser-ului dacă adăugarea elementelor se va face orizontal sau vertical. Browser-ul va fi cel care determină câte coloane sunt optime.

Detalii: O lista neordonată se afișează de obicei ca o lista marcată cu puncte. Tagurile sunt folosite pentru a specifica începutul unui element al listei.

3.5. Imagini si Sunet

HTML suportă atât imagini cât și sunet.

<area>

Scop: Definește o arie de mapare (*client-side*) a unei imagini.

Versiune: Extensie Netscape a versiunii HTML 3.0.

Attribute:

COORDS="x,y,.."	Specifică coordonatele formei.
HREF="URL"	Specifică URL-ul la care să facă referire.
NOHREF	Specifică faptul că nu face referire la nimic.
SHAPE=CIRCLE DEFAULT POLY	Specifică o formă care poate fi cerc, poligon

RECT	sau dreptunghi.
TARGET="window nume"	Specifică frame-ul în care se va deschide pagina cu URL-ul la care s-a făcut referire.

Detalii: Poate să apară numai într-un bloc marcat de tagul <map>.

<bgsound>

Scop: Specifică un sunet de background.

Versiune: Extensie a Internet Explorer-erului.

Attribute:

SRC="URL"	Specifică URL-ul fisierului de sunet.
-----------	---------------------------------------

Detalii: Formatele suportate sunt au, midi și wav.

<caption>.. </caption>

Scop: Conține ut titlu pentru o figură.

Versiune: HTML 3.0.

Attribute:

ALIGN= BOTTOM LEFT RIGHT TOP	Specifică alinierea titlului în funcție de figură.
------------------------------------	--

Detalii: Se folosește numai într-un bloc marcat de tagul <FIG>.

Scop: Specifică o imagine.

Versiune: HTML 2.0, Atributele BORDER, HEIGHT, WIDTH, HSPACE, VSPACE și a doua versiune a atributului ALIGN sunt extesii Netscape ale versiunii HTML 2.0; attributele HEIGHT și WIDTH apar în versiunea HTML 3.0; USEMAP este o extensie Netscape a versiunii HTML 3.0.

Attribute:

ALIGN=TOP MIDDLE BOTTOM LEFT RIGHT TOP	Specifică alinierea imaginii relativ la text
ALT="text"	Specifică un text care poate substitui imaginea în cazul în care acesta nu poate fi afișata.
BORDER=n	Specifică grosimea border-ului imaginii.
HEIGHT=n	Specifică înălțimea imaginii.
HSPACE=n	Specifică marginea orizontală pe care să o lase în jurul imaginii.
ISMAP	Indică dacă imaginea este mapată (server-side).
SRC="URL"	Specifică URL-ul imaginii.
USEMAP="URL"	Specifica URL-ul maparii (client-side) a imaginii.
VSPACE=n	Specifică marginea verticala pe care să o lase în jurul imaginii.
WIDTH=n	Specifică lățimea imaginii.

Detalii: GIF-ul și JPEG-ul sunt cele mai folosite formate. Pentru ca sa funcționeze o mapare de imagine, imaginea respectivă (tagul) trebuie să fie inclus într-un tag <a>..
.

<map>..</map>

Scop: Definește o mapare client a unei imagini.

Versiune: Extensie Netscape a versiunii HTML 3.0.

Attribute:

NAME="ancora"	Specifică o ancora pentru imaginea mapată.
---------------	--

Detalii: Ancora trebuie specificată în tagul .

În cazul în care două forme dintr-o mapă senzitivă se suprapun se va considera cea care are centrul mai apropiat de punctul selectat. Formele valide de mapare se afla in următorul tabel.

default	Specifică forma implicită atunci cand o imagine nu are o formă bine definită.
Circle x, y, r	Un cerc cu centru în x, y și rază r.
Polygon x1, y1, x2, y2, ...	Un poligon care are vârfurile în perechile de coordonate x,y.
Rect x, y, w, h	Un dreptunghi care are colțul din dreapta sus la x,y, iar lățimea este w și înălțimea h.

3.6. Formulare

Toate formularele HTML necesită un program CGI care să proceseze respectivul formular, ceea ce înseamnă că folosirea acestor taguri nu este suficienta pentru buna funcționare a unui formular. Formularele HTML sunt primitive deoarece nu valideaza intrările decât dupa ce acestea sunt trimise către CGI. Acestea pot fi validate folosind JavaScript sau Java.

<form>..</form>

Scop: Conține un formular.

Versiune: HTML 2.0.

Attribute:

ACTION="URL"	Specifică URL-ul programului care va procesa formularul.
METHOD=GET POST	Specifica metoda de transmitere a informațiilor din formular către programul de procesare.

Detalii: Între aceste taguri ar trebui să apară și alte tagui cum ar fi: <input> sau <textarea> care specifică elementele formularului.

<input>

Scop: Specifică un camp de intrare pentru utilizator.

Versiune: HTML 2.0 (prima versiune a atributului TYPE), HTML 3.0 (a doua versiune a atributului TYPE), toate attributele în afara de TYPE sunt extensii Netscape JavaScript.

Attribute:

onBlur="functie"	Specifică o funcție JavaScript care se apelează când
------------------	--

	câmpul respectiv nu este activ.
OnChange="functie"	Specifică o funcție JavaScript care se apelează când conținutul câmpului respectiv s-a modificat, iar câmpul nu mai este activ.
OnClick="functie"	Specifică o funcție JavaScript care se apelează când asupra câmpului respectiv se acționează un <i>mouse click</i> .
OnFocus="functie"	Specifică o funcție JavaScript care se apelează când câmpul respectiv devine activ.
OnLoad="functie"	Specifică o funcție JavaScript care se apelează când se încarcă pagina curentă.
OnMouseOver="functie"	Specifică o funcție JavaScript care se apelează când se trece cu pointerul mouse-ului peste câmpul respectiv.
onSelect="functie"	Specifică o funcție JavaScript care se apelează când userul selectează o parte de text dintr-un <i>text</i> sau <i>textarea</i> .
TYPE= CHECKBOX FILE HIDDEN IMAGE PASSWORD RADIO RANGE RESET SCRIBBLE SUBMIT TEXT	Specifică tipul câmpului.

<input type=checkbox>

Scop: Specifică un *checkbox*.

Versiune: HTML 2.0.

Atribute:

CHECKED	Specifică <i>checkbox</i> -ul care este implicit selectat.
NAME="nume"	Specifică numele <i>checkbox</i> -ului. Este obligatoriu.
VALOARE="valoare"	Specifică valoarea <i>checkbox</i> -ului. Este obligatoriu.

Detalii: Pot exista mai multe *checkbox*-uri care au un același nume specificat de atributul NAME. În acest fel se pot grupa mai multe elemente *checkbox*. Astfel se realizează o selecție multiplă.

<input type=hidden>

Scop: Permite unui document HTML să specifice câmpuri care au o valoare, dar care nu se vad în pagină (nu pot fi modificate).

Versiune: HTML 2.0.

Atribute:

NAME="nume"	Specifică numele câmpului. Este obligatoriu.
VALUE="valoare"	Specifică valoarea câmpului. Este obligatoriu.

Detalii: Un câmp ascuns nu poate fi văzut de către un utilizator.

<input type=radio>

Scop: Specifică un buton de radio.

Versiune: HTML 2.0.

Atribute:

CHECKED	Specifică butonul radio setat implicit.
NAME="nume"	Specifică numele butonului radio. Este obligatoriu.
VALUE="valoare"	Specifică valoarea butonului radio. Este obligatoriu.

Detalii: Pot exista mai multe butoane radio care să aiba un același nume specificat de atributul NAME. Numai unul dintre aceste butoane (grupate sub același nume) poate fi selectat la un moment dat. Astfel se creează o selecție 1 la n.

<input type=range>

Scop: Specifică faptul că un utilizator poate alege o valoare numerică dintr-un interval.

Versiune: HTML 3.0.

Atribute:

MAX=n	Specifică limita superioară a intervalului.
MIN=n	Specifică limita inferioară a intervalului.
NAME="nume"	Specifică numele câmpului. Este obligatoriu.

Detalii: Dacă MAX sau MIN sunt numere reale se vor accepta numai numere reale, altfel se vor accepta numai întregi.

<input type=reset>

Scop: Specifică un buton care resetează toate câmpurile formularului la valorile lor inițiale.

Versiune: HTML 2.0, atributul SRC apare în versiunea HTML 3.0.

Atribute:

SRC="URL"	Specifică o imagine care poate fi folosită ca și buton de reset.
VALUE="valoare"	Specifică o etichetă pentru butonul de reset.

<input type=submit>

Scop: Specifică un buton de submit.

Versiune: HTML 2.0, atributul SRC apare în versiunea HTML 3.0.

Atribute:

NAME="nume"	Specifică numele butonului.
SRC="URL"	Specifică o imagine care poate fi folosită ca și buton de submit.
VALUE="valoare"	Specifică o etichetă pentru butonul de submit.

Detalii: Atributelor NAME și VALUE li se atribuie o valoare numai în cazul în care există mai multe butoane de submit, iar programul specificat de URL prin atributul ACTION poate să-și dea seama care buton de submit a fost apăsător.

<input type=text> și <input type=password>

Scop: Specifică un câmp editabil (în care se poate introduce text).

Versiune: HTML 2.0.

Atribute:

MAXLENGTH=n	Specifică lungimea maximă a câmpului. Valoarea implicită este infinit.
-------------	--

NAME="nume"	Specifică numele câmpului.
SIZE=n	Specifică dimensiunea câmpului pe ecran. Dacă această dimensiune este mai mare decât valoarea atributului SIZE și mai mică decât MAXLENGTH câmpul va defila. Când nu specifică o valoare, browser-ul va alege o dimensiune optimă.
VALUE="valoare"	Specifică valoarea implicită a câmpului.

Detalii: Atributul NAME este singurul atribut obligatoriu, celelalte fiind opționale. Singura diferență între câmpurile text și password este afișarea, respectiv neafișarea caracterelor introduse.

<option>

Scop: Specifică un element al unei liste de selecții (o opțiune).

Versiune: HTML 2.0.

Atribute:

SELECTED	Specifică opțiunea implicit selectată.
VALUE="valoare"	Specifică valoarea opțiunii.

Detalii: Tagurile <OPTION> pot apărea doar într-un bloc determinat de tagul <select>. Fiecare tag <option> este urmat de un text care va apare în lista de selecții și va fi folosit ca valoarea a atributului VALUE în cazul în care acesta nu a fost specificat.

<select>..</select>

Scop: Specifică o listă de selecții.

Versiune: HTML 2.0, atributul DISABLED apare în versiunea HTML 3.0.

Atribute:

DISABLED	Specifică o listă de selecții care va fi afișată dar nu se pot face modificări în ea.
MULTIPLE	Specifică selectarea mai multor opțiuni în același timp. Dacă nu este specificat, se va selecta o singură opțiune la un moment dat.
NAME="nume"	Specifică numele listei de selecții.
SIZE=n	Specifică numărul opțiunilor care pot fi vizibile la un moment dat. Dacă nu este specificat, browser-ul va alege o valoare optimă.

Detalii: Se afișează sub forma unei liste care se defilează. Elementele listei se definesc folosind tagul <OPTION>.

<textarea>..</textarea>

Scop: Specifică un câmp multilinie de introducere a textului.

Versiune: HTML 2.0, attributele DISABLED apare în versiunea HTML 3.0, atributul WRAP este o extensie Netscape a versiunii HTML 3.0, iar primul set de attribute este o extensie Netscape JavaScript.

Atribute:

onBlur, onChange, onClick	A se vedea explicațiile date la tagul <input>.
COLS=n	Numărul coloanelor care vor fi afișate pe ecran.
DISABLED	Specifică un <i>text area</i> care va fi afișat pe ecran, dar în care nu se vor putea face modificări.

NAME="nume"	Specifică numele câmpului.
ROWS=n	Specifică numărul de rânduri care vor fi afișate pe ecran.
WRAP=OFF PHYSICAL VIRTUAL	Specifică modul de trecere automată la rândul următor. PHYSICAL înseamnă că trecerea la rând nou se va face atât la afișare cât și la transmitere. VIRTUAL înseamnă că trecerea la rând nou se va face doar la afișare și nu la transmitere.

Detalii: Textul care va apare între cele două taguri nu va fi valoarea implicită a câmpului. Atributele COLS și ROWS sunt folosite doar pentru a determina dimensiunile de afișare. Dacă textul introdus va fi mai mare decât fereastra *textarea*, atunci se vor folosi bare de defilare.

3.7. Introducerea elementelor de Java si JavaScript

Limbajele Java și JavaScript fac ca paginile Web să fie mai interactive. Firmele Sun și Netscape au introdus propriile taguri pentru aceste limbaje. Se discută înlocuirea acestor taguri prin unul singur: <insert>, însă nu s-a făcut nimic în acest sens.

<app>..</app>

Scop: Include un Alpha Java applet într-un document.

Versiune: Extensii Sun HTML.

Detalii: Se va folosi tagul <applet>.

<applet>..</applet>

Scop: Include un applet Java într-un document.

Versiune: Extensii Sun HTML.

Attribute:

CODEBASE	Specifică URL-ul sau directorul care conține appletul.
CODE	Atribut obligatoriu care conține numele fișierului cu clasa applet.
ALT	În cazul în care browserul nu poate interpreta tagul <applet>, se va înlocui cu textul introdus aici.
NAME	Un nume sau o ancoră pentru applet astfel ca celelalte applete din pagină să poată comunica cu appletul numit.
WIDTH, HEIGHT	Lățimea și înălțimea zonei în care se va desena appletul.
ALIGN	Alinierea appletului în același mod în care se face în tagul .
VSPACE, HSPACE	Spațiu vertical și orizontal lăsat liber în jurul appletului. Aceste attribute sunt asemănătoare celor din tagul .

<param>

Scop: Parametrii definiți de utilizator. Se scriu într-un bloc delimitat de tagurile <applet>..</applet>.

Versiune: Extensie Sun HTML.

Atribute:

NAME="atribut definit in applet"	Atribut definit în cadrul appletului.
VALUE="valoare"	Valoarea atributului.

<script>..</script>

Scop: Conține un script.

Versiune: Extensie Netscape JavaScript.

Atribute:

LANGUAGE=JAVASCRIPT	Specifică limbajul în care este scris scriptul. JavaScript poate fi un astfel de limbaj.
---------------------	--

3.8. Tabele

Tabelele reprezintă un instrument foarte puternic și extrem de util. Ele au apărut doar în versiunea HTML 3.0. Partea negativă este faptul ca tabelele arată foarte urât pe browserele care nu le suportă.

<table>..</table>

Scop: Definește un tabel și conține toate celelalte taguri care se referă la organizarea în tabele.

Versiune: HTML 3.0, atributele BORDER=n, CELLSPACING, CELLPADDING, și WIDTH=procent sunt extensii Netscape a versiunii HTML 3.0.

Atribute:

ALIGN= BLEEDLEFT BLEEDRIGHT CENTER JUSTIFY LEFT RIGHT	Specifică alinierea în cadrul tabelului.
BORDER	Specifică dacă tabelul are border.
BORDER=n	Specifică grosimea borderului tabelului.
CELLPADDING=n	Specifică spațiul dintre celulă și conținutul ei.
CELLSPACING=n	Specifică spațiul dintre celulele tabelului.
COLSPEC="specificarea coloanei"	Specifică lățimea și alinierea coloanelor.
NOFLOW	Nu se va putea scrie text în jurul tabelului.
NOWRAP	Specifică faptul că browserul nu va face trecere automată la rând nou.
UNITS= EN PIXELS RELATIVE	Specifică unitățile folosite pentru COLSPEC și WIDTH. EN reprezintă jumătate dintr-un punct. PIXELS este numărul pixelilor. RELATIVE este când fiecare coloana este relativă la celelalte.
WIDTH=n	Specifică lățimea tabelului. Netscape permite exprimarea lățimii în procente din lățimea documentului.

Detalii: Specificarea coloanei se face prin scrierea unei litere care specifică alinierea coloanei (C pentru centrat, D pentru aliniere zecimală pentru numerele reale, J pentru *justify*, L pentru aliniere la stânga și R pentru aliniere la dreapta) și un număr care reprezintă lățimea în UNITS.

<td>..</td> și <th>..</th>

Scop: Specifică celulele care conțin date, respectiv celulele de titlu.

Versiune: HTML 3.0.

Attribute:

ALIGN=CENTER DECIMAL JUSTIFY LEFT RIGHT	Specifică alinierea orizontală în cadrul celulei. DECIMAL asigură alinierea zecimalelor.
COLSPAN=n	Specifică o celulă care cuprinde n coloane.
NOWRAP	Nu se face trecere automată la rând nou.
ROWSPAN=n	Specifică o celulă care cuprinde n linii.
VALIGN= BASELINE BOTTOM MIDDLE TOP	Specifică alinierea verticală în cadrul celulei. BASELINE asigură o aceeași <i>baseline</i> pentru toate celulele din linie.

Detalii: <td> și <th> trebuie să apară doar într-un bloc marcat de tagurile <tr>..</tr>. Principala diferență dintre <td> și <th> este că celulele <th> au conținutul scris cu un font mai îngroșat.

<tr>..</tr>

Scop: Specifică o linie într-un tabel.

Versiune: HTML 3.0.

Attribute:

ALIGN=CENTER DECIMAL JUSTIFY LEFT RIGHT	Specifică alinierea orizontală în cadrul liniei. DECIMAL asigură aranjarea zecimalelor.
VALIGN=BASELINE BOTTOM MIDDLE TOP	Specifică alinierea verticală în cadrul celulei.

Detalii: Se întâlnește în cadrul unui bloc <table>.

3.9. Frame-uri

Frame-urile sunt o invenție Netscape. Ele împart fereastra browserului în mai multe ferestre, fiecare putând conține un document HTML.

<frame>

Scop: Defițește un frame din mulțimea de frame-uri.

Versiune: Extensie Netscape.

Attribute:

MARGINHEIGHT=n	Specifică marginile de sus și jos pentru frame.
MARGINWIDTH=n	Specifică marginile din dreapta și stanga pentru frame.
NAME="numele_ferestrei"	Specifică numele frame-ului. Cu acest nume se

	poate face referire la frame-ul respectiv.
NORESIZE	Frame-ul nu se poate redimensiona.
SCROLLING= YES NO AUTO	Specifică existența sau lipsa unei bare de defilare pentru frame. Valoarea AUTO indică faptul că browserul va decide dacă se existe sau nu bară de defilare.
SRC="URL"	Specifică URL-ul documentului care va fi deschis în frame-ul curent.

Detalii: Nu are tag de închidere. Poate să apară numai într-un bloc <frameset>.

<frameset>..</frameset>

Scop: Definește un document care este o mulțime de frame-uri și nu un document HTML.

Versiune: Extensie Netscape.

Atribute:

COLS="specific_coloanei"	Specifică lățimea coloanelor din <i>frameset</i> . Specificarea este o listă de elemente despărțite prin virgulă. Fiecare element al listei poate fi un procent, o valoare în pixeli sau o valoare relativă. Valoarea relativă se specifică prin * (însemnând 1), 2* (însemnând 2), s.a.m.d.
ROWS=" specific_liniei"	Specifică înălțimea liniei într-un <i>frameset</i> . Asemănătoare cu sintaxa COLS.

Detalii: Nu se folosește în același fișier cu tagul <body>.

<noframes>..</noframes>

Scop: Conține o alternativă pentru browserele care nu suportă *frame*-uri.

Versiune: Extensie Netscape.

Detalii: Trebuie folosit alături de un bloc <frameset>.

3.10. Comentarii

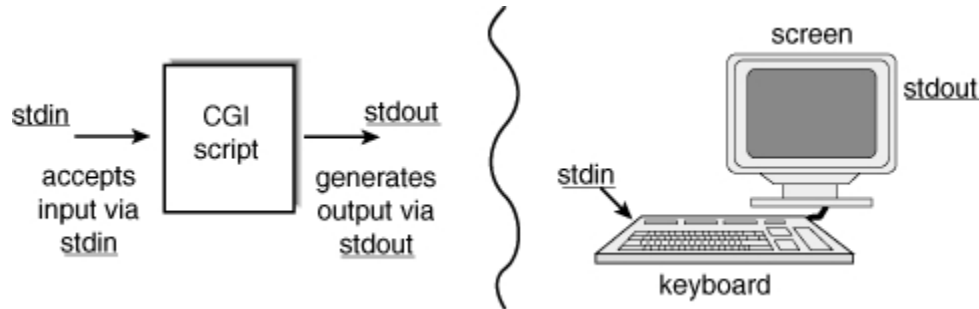
Un comentariu începe cu <!-- și se termină cu -->. Nu toate browserele (ex Lynx) recunosc comentariile.

4. Common Gateway Interface (CGI)

4.1. Ce este CGI?

Common Gateway Interface (CGI) [9] reprezintă o interfață a unui server Web care permite extinderea funcționalității acestuia. Folosind tehnologia CGI se poate realiza interacțiunea cu utilizatorii care accesează site-ul respectiv. Teoretic, CGI permite extinderea capacităților serverului interpretând intrările de la nivelul browserului și returnând informații în funcție de acele intrări. Practic, CGI este o interfață care permite programatorilor să scrie programe prin care se poate comunica ușor cu server-ul. Fără a fi nevoie de cunoștințe speciale, se poate scrie un program într-un limbaj de programare care să realizeze o interfață cu serverul Web. Acest program va rula pe orice server Web care înțelege protocolul CGI.

Comunicarea CGI se realizează prin intrările și ieșirile standard, ceea ce înseamnă că dacă știm cum să scriem sau să citim într-un limbaj de programare, putem scrie o aplicație Web la nivel de server.



În afară de realizarea intrărilor și ieșirilor, o aplicație CGI este aproape echivalentă cu orice altă aplicație.

Datorită faptului că CGI este o “interfață obișnuită”, nu există o restricție în ceea ce privește alegerea limbajului de programare. Cea mai frecventă întrebare pe care oamenii o pun este: “*Ce limbaj de programare trebuie folosit pentru a scrie o aplicație CGI?*” Răspunsul la aceasta întrebare este că se poate folosi orice limbaj care îndeplinește următoarele condiții:

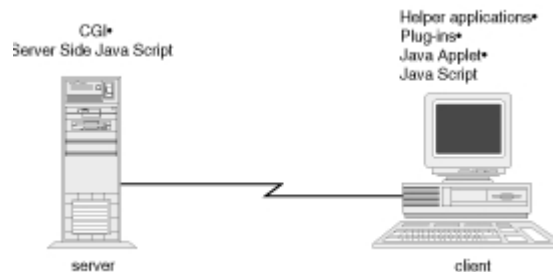
- ◆ Are instrucțiuni de tipărire la ieșirea standard
- ◆ Are instrucțiuni de citire de la intrarea standard
- ◆ Poate citi variabilele de mediu

Aproape toate limbajele de programare și multe dintre limbajele scriptuale pot realiza cele trei cerințe menționate mai sus, așadar se poate folosi oricare dintre aceste limbaje.

Limbajele pot fi clasificate în două mari categorii: compilate sau interpretate. Un limbaj compilat ca C sau C++ poate fi mic și rapid, în timp ce limbajele interpretative ca Perl sau Rexx necesită încărcarea unui interpretor mare.

Cele mai folosite limbaje în programarea CGI sunt C și Perl. Perl este un limbaj de nivel înalt foarte puternic. Deși este ușor de folosit, flexibil și puternic, ceea ce îl face să fie atractiv pentru programarea CGI, produce aplicații mari și încete, ceea ce nu este de dorit în anumite situații. Programele C sunt mai mici, mai eficiente și oferă un control mai bun asupra sistemului, deși este mai greu de programat și de depanat. După cum se vede, fiecare dintre cele două limbaje de programare au avantajele și dezavantajele lor.

Alternativa programării CGI este programarea API. Multe servere includ această alternativă care face ca programare să fie mai ușoară datorită faptului că se creează extensii directe ale serverului în comparație cu aplicațiile CGI, care sunt aplicații separate de server. API-urile tind să fie mai eficiente decât CGI-urile. Unele servere au funcționalități predefinite care permit efectuarea unor acțiuni cum ar fi accesul la o bază de date, fără ajutorul CGI-urilor. Alte aplicații pot rula la nivel de client (și nu la nivel de server), ca de exemplu Java.



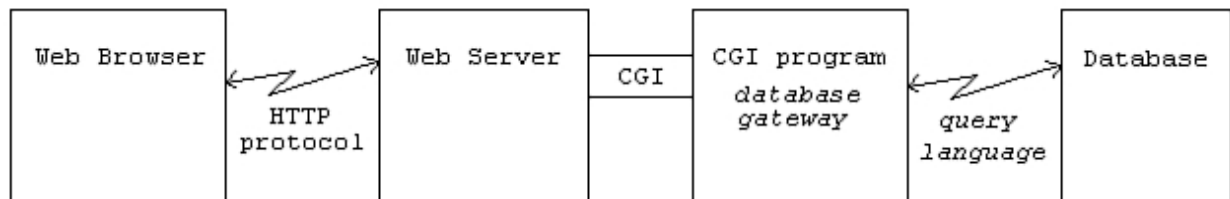
Pot deveni CGI-urile o tehnologie învechită, odată cu apariția acestor noi tehnologii?

Răspunsul este că CGI-urile au unele avantaje pe care noile tehnologii nu le au:

- Sunt portabile. Se poate scrie o aplicație CGI folosind aproape orice limbaj de programare, iar aplicația respectivă poate rula pe orice platformă. Alternativa oferită de API se poate realiza numai folosind anumite limbaje de programare care sunt mai greu de învățat.
- Tehnologiile client-server ca Java nu pot înlocui cu totul CGI-urile deoarece există anumite aplicații care se pot realiza numai la nivel de server.
- Multe dintre limitările CGI sunt limitări HTML sau HTTP.

4.2. Elemente de bază

Programarea CGI se poate reduce la două *task-uri* importante: obținerea informațiilor de la browserul Web și trimiterea altor informații înapoi la browser. Această este descrierea intuitivă a modului în care se creează o aplicație CGI. În mod concret, trebuie să existe un formular care trebuie să fie completat de către un utilizator. Odată completat acest formular, conținutul lui va fi transmis programului CGI prin apăsarea butonului de *submit*. Programul CGI trebuie să traducă informația pe care a primit-o în ceva ce poate fi înțeles de către program și care poate fi procesat ca apoi să se înoarca înapoi la browser ca o simplă înștiințare sau ca rezultat al unei căutări complexe într-o bază de date.



Programarea CGI necesită înțelegerea preluării intrărilor și trimiterea informațiilor spre ieșirea din browserul Web. Ceea ce se va întâmpla între aceste două etape dintr-un program CGI, depinde de ceea ce vrea programatorul să realizeze. Partea cea mai complexă dintr-un CGI constă întocmai în această etapă de mijloc.

Cel mai simplu program : Hello, World!

Cel mai simplu program este tradiționalul “*Hello world!*”. Un program echivalent (scris în Shell) pentru programarea CGI ar fi următorul:

```
#!/bin/sh

echo -e "Content-Type: text/html\n\n"
echo "<html> <head>\n"
echo "<title>Hello, world!</title>"
echo "</head>\n"
echo "<body>\n"
echo "<h1>Hello, world!</h1>\n"
echo "</body> </html>\n"
```

Salvând acest program sub numele de *hello.cgi*, el poate fi apelat dintr-un browser folosind o adresă de tip URL(Uniform Resource Locator)-ul:

http://hostname/directoryname/hello.cgi

unde *hostname*-ul este numele serverului Web, iar *directoryname* este directorul în care se află fișierul *hello.cgi* (de obicei acest director este **cgi-bin**).

După cum se poate observa, în programul de mai sus nu se folosesc decât instrucțiuni de tipărire. Trimiterea spre ieșirea browserului înseamnă trimiterea spre ieșirea standard. O a doua observație ar fi că prima instrucțiune de tipărire dintr-un CGI trebuie să fie specificarea tipului conținutului transmis browserului (*Content-Type: text/html*); acest lucru spune browserului că ceea ce va primi este o pagină HTML. Pot fi transmise și alte elemente cum ar fi imagini sau sunete, dar înainte de a fi transmise trebuie specificat tipul lor.

O a treia observație este că fișierul trebuie să aibă extensia **.cgi**. Nu în toate cazurile această extensie este necesară, dar ea le poate diferenția de fișierele HTML sau de fișierele text. Unele servere sunt configurate să execute numai acele fișiere care au extensia *.cgi* și să afișeze conținutul tuturor celorlalte fișiere.

Pe scurt, programul *hello.cgi* are două părți principale:

- Transmite browserului tipul de informație pe care acesta îl va aștepta (*Content-Type: text/html*)
- Transmite browserului informația propriuzisă (Hello, world!)

Hello, World! in C

Pentru a demonstra independența de limbaj a programelor CGI vom lista același program, *hello.cgi* scris în C și în Perl. Varianta în C este următoarea:

```
/* hello.cgi.c - Hello, world CGI */

#include <stdio.h>

int main() {
    printf("Content-Type: text/html\r\n\r\n");
    printf("<html> <head>\n");
    printf("<title>Hello, World!</title>\n");
```

```
printf("</head>\n");
printf("<body>\n");
printf("<h1>Hello, World!</h1>\n");
printf("</body> </html>\n");
}
```

Nici pentru serverului Web și nici pentru browser nu contează în ce limbaj de programare este scris CGI-ul. Deși fiecare limbaj de programare are avantajele și dezavantajele sale ca limbaj de programare al CGI-urilor, cel mai bun limbaj este acela care este cel mai bine stăpânit de către programator.

Îesirile CGI

Browserul așteaptă să primească două seturi de date: tipul informației pe care îl va afișa (conținut în linia *Content-Type*;) și informația propriu-zisă (care va fi afișată în browserul Web). Aceste două blocuri de informații sunt despărțite de o linie goală.

Prima linie care conține tipul informației se numește header HTTP. Acesta oferă o serie de informații asupra datelor pe care browserul le va primi. Există câteva tipuri diferite de header-e HTTP. Se pot folosi mai multe combinații de headere HTTP separate prin *carriage return* și linie nouă (*\r\n*). Linia goală are rolul de a separa headerul de restul datelor. Aceasta conține un *carriage return* și trecerea la o linie nouă.

Formatul unui header este următorul:

Content-Type: subtip/tip

unde *subtip/tip* reprezintă un tip MIME (Multipurpose Internet Mail Extensions). Cele mai cunoscute tipuri MIME sunt următoarele:

Tipuri MIME	Descriere
text/html	HyperText Markup Language (HTML)
text/plain	Simplu fișier text
image/gif	Imagini GIF
image/jpeg	Imagini compresate JPEG
audio/basic	Fișiere audio Sun *.au
audio/x-wav	Fișiere Windows *.wav

Ceea ce se va scrie după header și linia despărțitoare se va scrie în formatul în care se dorește să apară în browserul Web. În cazul în care se dorește transmitere de cod HTML se vor scrie taguri HTML. Dacă se dorește transmiterea de imagini sau sunete sau alte fișiere binare, se va scrie conținutul lor la ieșirea standard.

4.3. Instalarea și rularea programelor CGI

Orice server are nevoie de spațiu pe disc în care să-și țină fișierele și documentele HTML. Acest spațiu ocupat de server se numește *ServerRoot* (rădăcina serverului) și se află de obicei în directorul */etc/httpd/*, iar spațiul în care sunt documentele se numește *DocumentRoot* și se află în directorul */etc/httpd/htdocs/*.

Accesarea fișierelor prin browserul Web se face folosind o adreasare de tip URL, relativă la *DocumentRoot*.

Configurarea serverului pentru CGI

Majoritatea serverelor Web sunt preconfigurate pentru a folosi programe CGI. Există două lucruri care semnalează serverul de existența unei aplicații CGI:

- Un director predefinit. Unele servere permit specificarea a tuturor fișierelor dintr-un director ca fiind CGI-uri. Directorul respectiv se numește, de obicei, **cgi-bin**.
- Extensia din numele fișierului. Multe servere sunt preconfigurate să recunoască toate fișierele care se termină cu extensia **.cgi** ca fiind CGI-uri.

Instalarea CGI-urilor pe servere UNIX

Apelate din browser, aplicațiile CGI vor fi rulate pe serverul Web de către un user care nu există (nobody), nu are drepturi de acces și nu se poate loga pe server. În consecință, aplicațiile CGI trebuie să aibă drepturi de citire și execuție pentru oricine. Acest lucru se realizează, în UNIX, prin comanda:

```
chmod 755 nume_de_fisier_CGI
```

Pentru alte setări se va edita fișierul *conf/srm.conf*. Pentru a schimba locația directorului cgi-bin se va modifica linia:

```
ScriptAlias numele_fals_al_directorului numele_real_al_directorului
```

unde *numele_fals_al_directorului* este un nume fals al directorului (/cgi-bin), iar *numele_real_al_directorului* este calea completa spre directorul în care se afla programele CGI. Se pot configura mai multe *ScriptAlias*-uri prin adaugarea mai multor linii *ScriptAlias*.

Pentru configurarea serverului (Apache) astfel încât să recunoască orice fișier cu extensia *.cgi* trebuie editate două fișiere. Primul fișier este *srm.conf* în care trebuie decommentată linia:

```
AddType application/x-httpd-cgi .cgi
```

Aceasta va asocia tipul CGI MIME cu extensia *.cgi*. Al doilea fișier care trebuie editat este *access.conf* care permite fișierelor CGI să fie executate din orice director. Pentru aceasta se va adăuga opțiunea *ExecCGI* în linia:

```
Option Indexes FollowSymLinks ExecCGI
```

Acum, orice fișier cu extensia *.cgi* va fi considerat o aplicație CGI.

Instalarea CGI-urilor pe Windows

Majoritatea serverelor pentru Windows 3.1, Windows 95 și Windows NT sunt configurate să folosească metoda de recunoaștere a CGI-urilor după extensie. Reconfigurarea serverelor pe Windows necesită rularea programului de reconfigurare a serverului. Acest program va face schimbările necesare.

Pentru scripturile scrise în Perl nu se poate specifica interpretorul (în prima linie ca sub UNIX). Unele servere sunt preconfigurate să asocieze fișierele cu o anumită extensie cu un anumit interpretor. Multe servere Windows consideră toate fișierele cu extensia *.pl* ca fiind scripturi Perl. Dacă nu există o asemenea asociere se poate scrie un fișier bat care să apeleze odata cu scripturile și interpretoarele corespunzătoare.

Instalarea CGI-urilor pe Macintosh

Pentru Macintosh există exact două opțiuni de servere Web: *StarNine's WebStar* și predecesorul acestuia, *MacHTTP*. Ambele recunosc CGI-urile după extensia fișierului.

MacHTTP înțelege două extensii: *.cgi* și *.acgi* (CGI asincron). Rularea programelor CGI obișnuite, instalate pe Macintosh (cele cu extensia *.cgi*), vor ține serverul Web ocupat până la terminarea acestora, forțând serverul să pună toate celelalte cereri să aștepte. CGI-urile asincrone fac ca serverul să poată accepta și alte cereri în timpul rulării lor.

Rularea unui CGI

Există mai multe posibilități de rulare a CGI-urilor. Dacă CGI-ul este unul de informare, cum este programul *Hello, World!*, atunci el se poate rula prin simpla apelare prin intermediul URL-ului. Majoritatea programelor CGI sunt apelate din cadrul unui formular HTML.

4.4. Preluarea intrărilor de la un Browser

Cea mai importantă capacitate a unei aplicații CGI este aceea de a prelua informații de la un browser. Această trăsătură realizează natura interactivă a Web-ului.

Un program CGI primește două tipuri de informații:

- Informații furnizate de server: despre browser (tipul, ce date poate să vadă, numele mașinii la distanță, s.a.m.d.), despre server (numele și versiunea, portul pe care rulează, s.a.m.d.) și despre programul CGI (numele și locul unde se află). Serverul furnizează toate aceste informații (pentru programele CGI) prin intermediul variabilelor de mediu.
- Informații introduse de un utilizator. Aceste informații sunt transmise fie prin intermediul variabilelor de mediu (cu metoda GET) sau de la intrarea standard (cu metoda POST).

Varibilele de mediu

Cunoașterea variabilelor de mediu care sunt disponibile unui program CGI poate fi utilă putând fi folosită în procesul de debug-are. Tabelul de mai jos cuprinde toate variabilele de mediu pe care un program CGI le poate accesa.

Variabile de mediu	Scop
REMOTE_ADDR	Adresa IP a mașinii client.
REMOTE_HOST	Numele de host al mașinii client.
HTTP_ACCEPT	Tipurile MIME ale datelor pe care browserul le știe interpreta.
HTTP_USER_AGENT	Informații despre browser (numele, numărul versiunii, sistemul de operare, etc).
REQUEST_METHOD	GET sau POST.
CONTENT_LENGTH	Dimensiunea lungimii intrărilor în cazul în care acestea sunt transmise cu metoda POST. Dacă sunt transmise cu metoda GET, aceasta variabilă nu este definită.
QUERY_STRING	Conține informațiile transmise din browserul Web în cazul în care acestea au fost transmise cu metoda GET.
PATH_INFO	Permite utilizatorului să specifice o cale din linia de comandă CGI (de exemplu, http://hostname/cgi-bin/programname/path).
PATH_TRANSLATED	Traduce calea relativă din PATH_INFO în calea curentă din

sistem.

Pentru a scrie un program CGI care să afișeze variabilele de mediu trebuie să știm două lucruri:

- Să determinăm toate variabilele de mediu și valorile lor corespunzătoare.
- Să se tipărească rezultatul la ieșirea standard.

Programul CGI scris în Perl, *env.cgi*, care afișează toate variabilele de mediu este următorul:

```
#!/usr/local/bin/perl

print "Content-type: text/html\n\n";

print "<html> <head>\n";
print "<title>CGI Environment</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>CGI Environment</h1>\n";

foreach $env_var (keys %ENV) {
print "<B>$env_var</B> = $ENV{$env_var}<BR>\n";
}

print "</body> </html>\n";
```

Programul similar scris în C este:

```
/* env.cgi.c */

#include <stdio.h>

extern char **environ;

int main()
{
char **p = environ;

printf("Content-Type: text/html\r\n\r\n");
printf("<html> <head>\n");
printf("<title>CGI Environment</title>\n");
printf("</head>\n");
printf("<body>\n");
printf("<h1>CGI Environment</h1>\n");

while(*p != NULL)
printf("%s<br>\n", *p++);

printf("</body> </html>\n");
}
```

GET Versus POST

Care este diferența dintre metodele GET și POST? Folosind metoda GET intrările se transmit prin variabila de mediu QUERY_STRING, în timp ce folosind metoda POST intrările se transmit prin intrarea standard. Metoda POST este cea mai folosită datorită faptului că se pot trimite mult mai multe date decât folosind metoda GET, aceasta fiind limitată de dimensiunea care poate fi stocată în variabila de mediu.

Pentru a determina ce metodă s-a folosit se va verifica variabila de mediu REQUEST_METHOD, care poate avea una din valorile GET sau POST. Dacă va avea valoarea POST, lungimea informației va fi stocată în variabila de mediu CONTENT_LENGTH.

Codificarea intrărilor

Când un utilizator submitează un formular, browserul codifică informația înainte de a o transmite serverului (programului CGI). Fiecarui tag <input> îi este asociat un nume simbolic, care face ca acest element al unui formular să poată fi referit ca o variabilă. Valoarea introdusă de utilizator poate fi considerată valoarea variabilei asociată elementului <input>.

Codificarea intrărilor se face folosind specificările codificării URL. Aceste specificări pot fi descrise folosind următoarele reguli:

- Câmpurile sunt despărțite prin ampersand (&).
- Numele și valorile sunt separate prin semnul egal (=), cu numele în stanga și valoarea în partea dreaptă.
- Înlocuiește spațiile cu semnul plus (+).
- Înlocuiește toate caracterele "abnormale" cu semnul procent (%) urmat de două cifre hexadecimale care reprezintă codul caracterului.

În urma codificării va rezulta un șir de forma:

name1=value1&name2=value2&name3=value3 ..

Presupunând că avem un formular ca cel descris în pagina de mai jos:

```
<html> <head>
<title>Name and Age</title>
</head>
<body>
<form action="/cgi-bin/nume_varsta.cgi" method=POST>
Numele: <input type=text name="nume"><p>
Varsta: <input type=text name="varsta"><p>
<input type=submit>
</form>
</body> </html>
```

Dacă un utilizator va introduce valorile *Ion Vasile* pentru câmpul *nume* și valoarea *21* pentru câmpul *varsta*, intrarea va fi codificată sub forma:

nume=Ion+Vasile&varsta=21

Traducerea intrărilor

Ca informațiile transmise dintr-un formular HTML să poată fi înțelese, acestea trebuie traduse în ceva care poate fi folosit de un program CGI. Având în vedere că există librării disponibile care fac o asemenea traducere se pot folosi aceste librării.

Vom da în continuare două exemple de astfel de librării:

cgi-lib.pl

În *cgi-lib.pl*, se folosește funcția *ReadParse* pentru a pune perechile nume/valoare într-un tablou. Codul funcției *ReadParse* este următorul:

ReadParse (from Steve Brenner's cgi-lib.pl).

```
sub ReadParse {
    local (*in) = @_ if @_;
    local ($i, $key, $val);

    # Read in text
    if (&MethGet) {
        $in = $ENV{'QUERY_STRING'};
    } elsif (&MethPost) {
        read(STDIN, $in, $ENV{'CONTENT_LENGTH'});
    }

    @in = split(/[&;]/, $in);

    foreach $i (0 .. $#in) {
        # Convert plus's to spaces
        $in[$i] =~ s/\+/ /g;

        # Split into key and value.
        ($key, $val) = split(=/, $in[$i], 2); # splits on the first =.

        # Convert %XX from hex numbers to alphanumeric
        $key =~ s/%(..)/pack("c", hex($1))/ge;
        $val =~ s/%(..)/pack("c", hex($1))/ge;

        # Associate key and value
        $in{$key} .= "\0" if (defined($in{$key})); # \0 is the multiple
        separator
        $in{$key} .= $val;
    }

    return scalar(@in);
}
```

Dacă există mai multe perechi nume/valoare care au același nume, *ReadParse* pune toate aceste valori în aceeași intrare a tabloului separate de caracterul null.

Codul minimal al traducerii intrărilor este descris în următoarea secvență:

Codul Perl minimal folosind cgi-lib.pl.

```
#!/usr/local/bin/perl

require 'cgi-lib.pl';

&ReadParse(*input);
```

Dacă avem o variabilă cu numele *telefon*, valoarea acesteia poate fi accesată folosind sintaxa `$input{'phone'}`.

cghtml

Procesarea intrărilor CGI în C este mult mai complexă decât în Perl; în consecință, programul *cghtml* este mult mai complex.

Înainte de a descrie biblioteca *cghtml* trebuie definite niște liste înlănțuite:

Definirea listelor înlănțuite în llist.h (from Eugene Kim's cghtml).

```
typedef struct {
    char *name;
    char *value;
} entrytype;

typedef struct _node {
    entrytype entry;
    struct _node* next;
} node;

typedef struct {
    node* head;
} llist;
```

Fiecare intrare într-o listă înlănțuită stochează separat numele și valoarea unei variabile. Pentru accesarea valorii unei variabile trebuie parcursă toată lista, de la început și se va compara numele variabilei respective până se va găsi elementul corespunzător din listă. Funcția *read_cgi_input()* este echivalentă funcției *ReadParse* din *cgi-lib.pl*, exceptând faptul că perechile nume/valoare sunt așezate într-o listă înlănțuită. *read_cgi_input()* folosește funcțiile *x2c()* și *unescape_url()* pentru a decoda caracterele codificate după principiile URL.

read_cgi_input().

```
/* x2c() and unescape_url() stolen from NCSA code */
char x2c(char *what)
{
    register char digit;

    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A') + 10 : (what[0]
- '0'));
    digit *= 16;
```

```

digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1]
- '0'));
return(digit);
}

void unescape_url(char *url)
{
    register int x,y;

    for (x=0,y=0; url[y]; ++x,++y) {
        if((url[x] = url[y]) == '%') {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
        url[x] = '\\0';
    }

    int read_cgi_input(llist* entries)
    {
        int i,j,content_length;
        short NM = 1;
        char *input;
        entrytype entry;
        node* window;

        list_create(entries);
        window = (*entries).head;

        /* get the input */
        if (REQUEST_METHOD == NULL) {
            /* perhaps add an HTML error message here for robustness sake;
            don't know whether CGI is running from command line or from
            web server. In fact, maybe a general CGI error routine might
            be nice, sort of a generalization of die(). */
            fprintf(stderr,"caught by cgihtml: REQUEST_METHOD is null\n");
            exit(1);
        }
        if (!strcmp(REQUEST_METHOD,"POST")) {
            if (CONTENT_LENGTH != NULL) {
                content_length = atoi(CONTENT_LENGTH);
                input = malloc(sizeof(char) * content_length + 1);
                if (fread(input,sizeof(char),content_length,stdin) !=
                content_length) {
                    /* consistency error. */
                    fprintf(stderr,"caught by cgihtml: input length <
                    CONTENT_LENGTH\n");
                    exit(1);
                }
            }
            else { /* null content length */
                /* again, perhaps more detailed, robust error message here */
                fprintf(stderr,"caught by cgihtml: CONTENT_LENGTH is null\n");
                exit(1);
            }
        }
    }
}

```

```

else if (!strcmp(REQUEST_METHOD, "GET")) {
    if (QUERY_STRING == NULL) {
        fprintf(stderr, "caught by cgihtml: QUERY_STRING is null\n");
        exit(1);
    }
    input = newstr(QUERY_STRING);
    content_length = strlen(input);
}
else { /* error: invalid request method */
    fprintf(stderr, "caught by cgihtml: REQUEST_METHOD invalid\n");
    exit(1);
}
/* parsing starts here */
if (content_length == 0)
    return 0;
else {
    j = 0;
    entry.name = malloc(sizeof(char) * content_length + 1);
    entry.value = malloc(sizeof(char) * content_length + 1);
    for (i = 0; i < content_length; i++) {
        if (input[i] == '=') {
            entry.name[j] = '\\0';
            unescape_url(entry.name);
            if (i == content_length - 1) {
                strcpy(entry.value, "");
                window = list_insafter(entries, window, entry);
            }
            j = 0;
            NM = 0;
        }
        else if ( (input[i] == '&') || (i == content_length - 1) ) {
            if (i == content_length - 1) {
                entry.value[j] = input[i];
                j++;
            }
            entry.value[j] = '\\0';
            unescape_url(entry.value);
            window = list_insafter(entries, window, entry);
            j = 0;
            NM = 1;
        }
        else if (NM) {
            if (input[i] == '+')
                entry.name[j] = ' ';
            else
                entry.name[j] = input[i];
            j++;
        }
        else if (!NM) {
            if (input[i] == '+')
                entry.value[j] = ' ';
            else
                entry.value[j] = input[i];
            j++;
        }
    }
}

```

```
return 1;
}
}
```

4.5. Un program CGI simplu

Pentru formularul descris în fișierul HTML listat mai sus se vor descrie câte un program CGI (scris atât în Perl cât și în C).

Fișierul CGI-ului scris în Perl cuprinde următoarele instrucțiuni:

```
#!/usr/local/bin/perl
# nume_varsta.cgi
require 'cgi-lib.pl'

&ReadParse(*input);
print "Content-Type: text/html\r\n\r\n";
print "<html> <head>\n";
print "<title>Nume si varsta </title>\n";
print "</head>\n";
print "<body>\n";
print "Hello, " . $input{'nume'} . ". Ai\n";
print $input{'varsta'} . " ani.<p>\n";
print "</body> </html>\n";
```

Fișierul echivalent scris în C este următorul:

```
/* nume_varsta.cgi.c */

#include <stdio.h>
#include "cgi-lib.h"

int main()
{
    llist input;

    read_cgi_input(&input);
    printf("Content-Type: text/html\r\n\r\n");
    printf("<html> <head>\n");
    printf("<title> Nume si varsta </title>\n");
    printf("</head>\n");
    printf("<body>\n");
    printf("Hello, %s. Ai\n", cgi_val(input, "nume"));
    printf("%s ani.<p>\n", cgi_val(input, "varsta"));
    printf("</body> </html>\n");
}
```

Aceste două programe sunt aproape echivalente. Amândouă folosesc rutinele de traducere a intrărilor din librăriile descrise mai sus.

5. Aplicații INTRANET

Aplicațiile INTRANET au cunoscut în ultima perioadă o dezvoltare tot mai mare. Din nevoia de comunicare o oamenilor, aplicațiile au devenit tot mai complexe.

Căutarea studenților

(<http://www.scs.ubbcluj.ro/scs/cautare/index.html>)



User	Nume	Pagina Web
ar3170	Ardelean Raluca Dana	
ba27863	Baciu Adriana Minodora	
ba27875	Bunea Ana Rodica	
ba3174	Banarescu Adrian	
ba33	Borz Ana Gabriela	
ba369	Banabic Anca Claudia	
bi27657	Brad Ioana Maria	
bi3172	Baciu Ioana	
bi3175	Banc Ioana Maria	
bi38177	Besa Iuliana Corina	
bi883	Bordas Ioana	
ca2481	Chira Anamaria	
ca27662	Catana Adrian Ioan	http://www.scs.ubbcluj/~ca27662
ca27879	Chereches Alina Oana	http://www.scs.ubbcluj/~ca27879
ca28271	Caciula Ana Maria	
ca8112	Chereches Ana Maria	http://www.scs.ubbcluj/~ca8112
ca28274	Candea Cosmina Ioana	

Această primă aplicație este o mașină de căutare care caută, în funcție de sirul introdus în *textbox*, utilizatorii existenți pe serverul studenților (disco.scs.ubbcluj.ro). Aplicația este o aplicație CGI scrisă în shell.

Fișierele care cuprind această aplicație sunt în număr de trei și vor fi prezentate în continuare.

Primul fișier este cel care cuprinde pagina HTML. El conține un formular care va apela un program CGI scris în shell.

search.html

[illegible]

Formularul cuprinde două elemente:

- un *textbox* în care se va introduce, de către utilizator, șirul după care se va face căutarea;
- o imagine care substituie butonul de *submit*.

Al doilea fișier conține sursa scrisă în shell a CGI-ului.

cauta.cgi

```
#!/bin/sh
echo -e "Content-type: text/html\n\n"
echo "<html><head><base target=_top></head>"
echo "<body bgcolor=#FDFAC6><pre>"
read intrare
for name in `echo $intrare |sed 's/^\&/=/' |awk -F= '{print $2}'|sed 's/+^\&/g'`
do
cut /etc/passwd -d : -f 1,5,6|grep $name --ignore-case>cautare1

if [ -s cautare1 ]
    then echo "<b>User   Nume                               Pagina Web</b><br>"
    else echo " Nu exista<b> $name </b>!"
fi
```

```

    fi
done
for linie in `cut cautare1 -d : -f 3`
do
    if test -d $linie/public_html
    then
        grep $linie cautare1|sed -e "s/$/:x/1">cautare2
        #echo $linie are
    else
        grep $linie cautare1|sed -e "s/$/:1">cautare2
    #echo $linie
    fi
done

cut cautare2 -d : -f 1,2,4 | awk -F : -f cauta.awk

done

echo "</pre></body></html>"

```

Se preia în variabila *name* sirul introdus de utilizator și se va căuta o corespondență a acestui șir în fișierul */etc/passwd/*, care conține date despre toți utilizatorii din sistem. Dacă exista măcar un utilizator care să îndeplinească condițiile, atunci se va continua prelucrarea detelor. În caz contrar de va afișa mesajul: *'Nu există utilizator!'* Din rezultatele acestei căutări se vor extrage trei informații pentru fiecare utilizator care corespunde căutării. Aceste informații sunt:

- ❑ Numele de login;
- ❑ Numele și prenumele real;
- ❑ Pagina Web (dacăutilizatorul are o astfel de pagină).

Pentru verificarea paginii de Web a unui utilizator se va apela un program *awk*.

cauta.awk

```

#!/bin/awk
{
if($3 != x) print "<a href='mailto:" $1 "@scs.ubbcluj.ro">" $1 "</a>" $2 " <a
href='../../~" $1 ">http://www.scs.ubbcluj/~" $1 "</a>" "
else print "<a href='mailto:" $1 "@scs.ubbcluj.ro">" $1 "</a>" $2
}

```

Acest program va realiza și afișarea la ieșirea standar.

Citirea notelor (<http://www.scs.ubbcluj.ro/~sanda>)

Cea de a doua aplicație este tot o aplicație CGI care întreține o bază de date a notelor obținute de studenți la examene. Există trei tipuri de persoane care pot accesa această bază de date:

- ◆ SUPERUSER
- ◆ GESTIONAR
- ◆ STUDENT

Aceste persoane vor avea acces diferențiat la baza de date.

La nivel de **SUPERUSER** se pot efectua următoarele operații:

- ✓ Creare BD
- ✓ Setare BD curenta

Pentru CREARE unei baze de date trebuie setate de către administratorul rețelei drepturi de scriere, citire și execuție (777) asupra directorului în care se află fișierele care conțin bazele de date.

Prin SETAREA B.D. CURENTE se înțelege scrierea numelui fișierului care conține respectiva bază de date în fișierul *BD_cur.h* de unde toate celelalte CGI-uri se vor informa.

La nivel de **GESTIONAR** se pot efectua două tipuri de operații:

- la nivel de înregistrarea (student)
 - Adăugare student (înregistrare)
 - Ștergere student (înregistrare)
 - Sortare BD
- la nivel de materie
 - Adăugare materie
 - Modificare materie
 - Ștergere materie

Toate aceste operații se vor efectua asupra bazei de date curente.

La nivel de **STUDENT** se poate efectua a singură operație

- Consultare BD (Citire note)

Această consultare se realizează pe baza unei parole și a unui nume de login.

CREARE B.D.

Crearea unei baze de date înseamnă crearea unui fișier text, care are pentru început structura:

Nume Prenume Nr_matr Login Parola

CREARE BAZA DE DATE



create.html

```
<html>
<head>
<title>Citire note</title>
<link rel='STYLESHEET' type='text/css' href='general.css'>
</head>
<body>
<h1>Creare baza de date</h1>

  <form method=post action="cgi-bin/create.cgi">
<blockquote>
  Nume BD  <input type="text" name="NumeBD"><br>
<br><br>
  <input type="submit" value="OK" width=50>
  </blockquote>
</form>
</body>
</html>
```

Fișierul care conține CGI-ul care procesează formularul de mai sus este:

create.c

```
#include <stdio.h>
#include <stdlib.h>
#include "html.h"
#include "util.h"

struct {
    char *val;
    char *name;
}e[20];
int cl, x, m;

int nr_linii;
```

```

void creare(char *datafile){
    FILE *data;

    if((data = fopen(datafile, "w+"))==NULL) {
        printf("Nu pot crea fisierul baza de date\n");
        perror("<br>");
        exit(0);
    }
    else printf("Sa creat fisierul %s\n", datafile);

    fprintf(data, "Nume Prenume      Nr_matr      Login Parola");
    fclose(data);
}

main(int argc, char **argv){
    int i, n;

    printf("Content-type: text/html\n\n");

    begin_html();

    cl = atoi (getenv ("CONTENT_LENGTH"));
    for(x = 0; cl && (!feof(stdin)); x++){
        m = x;
        e[x].val = fmakeword(stdin, '&', &cl);
        plustospace (e[x].val);
        unescape_url (e[x].val);
        e[x].name = makeword (e[x].val, '=');
        //      printf(" %d %s %s<br>", x, e[x].name, e[x].val);
    }
    printf("<h2>Crearea BD %s</h2>", e[0].val);

    creare(e[0].val);
    /*  n=atoi(e[1].val);
    printf("%d\n",n);

    printf("<form action=preia_struct.cgi method=POST>");
    printf("formular");
    printf("</form>");*/
    end_html();
}

```

SETARE B.D. CURENTĂ

Se iau toate fişierele care conţin structura cu care se creează bazele de date şi se construieşte o pagină cu o listă de selecţie care le conţine:

CGI-ul care generează această pagină este:

BD_curenta.cgi

```
#!/bin/sh
echo -e "Content-type: text/html\n\n"
echo "<html><head><title>Setarea Bazei de Date</title><head>"
echo "<link rel='STYLESHEET' type='text/css' href='../general.css'>"
echo "<body><h1>Setare Baza de Date curenta</h1>"
echo "<form action='setare_BD.cgi' method='POST'>"
echo "<select name='baza_date'>"
grep "Nume Prenume Nr_matr Login Parola" -l *|grep -v
".c"|grep -v temp|awk '{print "<option>", $0}'
echo "</select><br><br>"
echo "<input type='submit' value='Seteaza'>"
echo "<input type='reset' value='Anuleaza'>"
echo "</form>"
echo "<blockquote><a href='javascript:history.back()'><img
src='../back.gif' alt='Back' border=0></a>"
echo "</blockquote></body></html>"
```

În continuare vom prezenta sursa CGI-ului care face setarea bazei de date curente.

setare_BD.c

```
#include "util.h"
#include "html.h"
#include <ctype.h>
#include <time.h>

#define bd_cur "BD_cur.h"

struct {
    char *val;
    char *name;
}e[20];
int cl, x, m;
```

```

main() {
    FILE *bd;

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();

    cl = atoi (getenv ("CONTENT_LENGTH"));
    for(x = 0; cl && (!feof(stdin)); x++){
        m = x;
        e[x].val = fmakeword(stdin, '&', &cl);
        plustospace (e[x].val);
        unescape_url (e[x].val);
        e[x].name = makeword (e[x].val, '=');
        //      printf(" %d %s %s<br>", x, e[x].name, e[x].val);
    }
    printf("<h2>Setare fisierului %s ca Baza de Date curenta</h2>",
e[0].val);
    if((bd=fopen(bd_cur, "w+"))==NULL){
        printf("Nu pot deschide fisierul de setare!<br>");
        exit(0);
    }
    fprintf(bd, "%s", e[0].val);
    printf("S-a setat ca <i>Baza de Date</i> curenta fisierul: %s",
e[0].val);
    fclose(bd);

    end_html();
}

```

ADĂUGARE STUDENT (ÎNREGISTRARE)

Adăugarea unui student înseamnă adăugarea unei înregistrări în baza de date.

ADAUGARE INREGISTRARE STUDENT

Nume:

Prenume:

Nr. matricol:

*

*

*

Se vor introduce *Numele Prenumele* și *Nr_matr*. Pe baza acestor informații se vor genera un *Login* și o *Parola*. Pentru materiile introduse în baza de date se va completa cu nota 0.

adaug_stud.html

[illegible]

Sursa CGI-ului care tratează formularul de introducere a unui nou student este:

adaug_stud.c

```
#include "util.h"
#include "html.h"

#define bd_cur "BD_cur.h"
#define MAXCAMPURI 100
#define sep "      "

int pozitie=0;
int nr_campuri, ind;
char campuri[MAXCAMPURI][50];

struct {
    char *val;
    char *name;
```

```

    }e[20];
    int cl, x, m;

    void cit_campuri(FILE *dat){
        char *c, sir[320];
        int i;
        fseek(dat, 0L, SEEK_SET);
        fgets(sir, 320, dat);
        c=strtok(sir, sep);
        nr_campuri=0;

        while (c!=NULL){
            strcpy(campuri[++nr_campuri], c);
            c=strtok(NULL, sep);
        }

        fseek(dat, 0L, SEEK_END);
    }

    char *fara_spatii(char *s){
        char sir[320], *c;
        int i=0;
        c=strtok(s, sep);

        while (c!=NULL){
            if (!i) {sprintf(sir, "%s", c); i++;}
            else sprintf(sir, "%s_%s", sir, c);
            c=strtok(NULL, sep);
        }
        strcpy(s,sir);
        return (s);
    }

    main(){
        int i, nota[MAXCAMPURI];
        FILE *data, *bd;
        char ssir[320], *c, tot[320], pass[12], nlog[12], studfile[320];
        char litere[35]="abcdefghijklmnopqrstuvwxyz1234567890";

        printf("Content-type: text/html%c%c", 10, 10);
        begin_html();
        printf("\n<h1>Adaugare inregistrare <i>student</i></h1>\n");

        cl = atoi (getenv ("CONTENT_LENGTH"));
        for(x = 0; cl && (!feof(stdin)); x++){
            m = x;
            e[x].val = fmakeword(stdin, '&', &cl);
            plustospace (e[x].val);
            unescape_url (e[x].val);
            e[x].name = makeword (e[x].val, '=');
            // printf(" %d %s %s<br>", x, e[x].name, e[x].val);
        }
        if
        ((strlen(e[0].val)==0)|| (strlen(e[1].val)==0)|| (strlen(e[2].val)==0))
            printf("Unul dintre cele trei campuri nu au fost completate!");
        else{

```

```

    if((bd=fopen(bd_cur, "r"))==NULL){
        printf("Nu pot deschide fisierul de setare al Bazei de Date
curente");
        exit(0);
    }
    fscanf(bd, "%s", studfile);
    fclose(bd);
    printf("<h3>In Baza de Date <i>%s</i></h3>", studfile);
    if((data=fopen(studfile, "a+")==NULL){
        printf("Nu pot deschide fisierul <i>%s</i>", studfile);
        exit(0);
    }

    cit_campuri(data);

    sprintf(nlog, "%c%c%s", tolower(e[0].val[0]), tolower(e[1].val[0]),
e[2].val);
    srand(time(NULL));
    for(i=0; i<8; i++) pass[i]=litere[random()%35];
    pass[8]=0;

    sprintf(tot, "%s\t%s\t%s\t%s\t%s", fara_spatii(e[0].val),
fara_spatii(e[1].val), e[2].val, nlog, pass);

    for(i=1; i<=nr_campuri-5; i++)
        sprintf(tot, "%s\t0", tot);
    fprintf(data, "\n%s", tot);
    printf("<p align='justify'>Am adaugat inregistrarea: <i>%s</i>",
tot);

    printf("</p>");

    printf("</form>\n");
    printf("<br><br><a href='listare.cgi?%s'>LISTARE</a>", studfile);
    fclose(data);
}
end_html();
}

```

ȘTERGERE STUDENT (ÎNREGISTRARE)

Pentru ștergerea unei înregistrări se va genera o pagină care să aibă o listă de selectie din care se va selecta studentul care se dorește a fi șters.

STERGERE INREGISTRARE STUDENT

Nume:



CGI-ul care realizează pagina cu lista de selecție este:

stergere Stud.c

```
#include <stdio.h>
#include <stdlib.h>
#include "html.h"

#define bd_cur "BD_cur.h"

char studfile[320];

void cit_studenti() {
    FILE *data;
    char nume[36], prenume[36], linie[320];
    int pozitie;

    data = fopen(studfile, "r");
    printf("<font color=#FFFFCC>Nume:</font>\n<select name='nume'>\n");
    fgets(linie, 320, data);
    while (!feof(data)) {
        fgets(linie, 320, data);
        pozitie++;
        sscanf(linie, "%s\t%s", nume, prenume);
        printf("<option >%s\t%s\n", nume, prenume);
    }
    printf("</select><br>\n");
    fclose(data);
}

main() {
    FILE *bd;

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();
    if ((bd=fopen(bd_cur, "r"))==NULL) {
        printf("Nu pot deschide fisierul de setare al Bazei de Date
curente");
        exit(0);
    }
    fscanf(bd, "%s", studfile);
}
```

```

fclose(bd);
printf("\n<h1>Stergere inregistrare <i>student</i></h1>\n");
printf("<form action='sterg_stud.cgi' method='POST'>\n");
printf("<blockquote>\n<table>\n<tr class=head><td>\n");
cit_studenti();
printf("<br><input type='submit' value='Sterge
student'>&nbsp;&nbsp;&nbsp;<input type='reset' value='Anuleaza'>\n");
printf("</td></tr>\n</table>\n</blockquote>\n");
printf("</form>\n");
end_html();
}

```

CGI-ul care realizează ștergerea studentului selectat din baza de date este:

sterg stud.c

```

#include "util.h"
#include "html.h"
#include <ctype.h>

#define bd_cur "BD_cur.h"
#define tempfile "temp.txt"
char studfile[320];

struct {
    char *val;
    char *name;
}e[20];
int cl, x, m;

int cauta(char * nume){
    FILE *data;
    char linie[320], num[36], pren[36];
    int i, ind, pozitie;

    if((data = fopen(studfile, "r"))==NULL){
        printf("Nu pot deschide fisierul Baza de Date <i>%s</i>",
studfile);
        return (-1);
    }
    ind=0;
    pozitie=0;
    fgets(linie, 320, data);
    while (!feof(data)){
        pozitie++;
        sscanf(nume,"%s\t%s", num, pren);
        if((strstr(linie, num)!=NULL)&&(strstr(linie, pren)!=NULL))
ind=pozitie;
        fgets(linie, 320, data);
    }
    fclose(data);
    return (ind);
}

```

```

void sterge(char *fisier, int pozitie){
    char l[320];
    int p;

    FILE *data, *temp;

    printf("\n<h1>Stergere</h1>\n");
    if ((data=fopen(fisier, "r"))==NULL){
        printf("Nu pot deschide Baza de Date");
        exit(0);
    }
    if((temp=fopen(tempfile, "w"))==NULL){
        printf("Nu pot folosi fisierul temporar");
        exit(0);
    }
    p=0;
    fgets(l, 320, data);

    while(!feof(data)){
        p++;
        l[strlen(l)-1]=0;
        if (p!=pozitie) {
            if(p!=1) fprintf(temp, "\n");
            fprintf(temp, "%s", l);
            printf("%s<br>\n", l);
        }
        fgets(l, 320, data);
    }
    fclose (temp);
    fclose (data);

    if ((rename(tempfile, fisier))===-1) printf("Nu pot redenumi
    fisierul");
}

main(){
    FILE *bd;

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();

    cl = atoi (getenv ("CONTENT_LENGTH"));
    for(x = 0; cl && (!feof(stdin)); x++){
        m = x;
        e[x].val = fmakeword(stdin, '&', &cl);
        plustospace (e[x].val);
        unescape_url (e[x].val);
        e[x].name = makeword (e[x].val, '=');
        printf(" %d %s %s<br>", x, e[x].name, e[x].val);
    }

    if((bd=fopen(bd_cur, "r"))==NULL){
        printf("Nu pot deschide fisierul de setare al Bazei de Date
        curente");
        exit(0);
    }
}

```

```

        fscanf(bd, "%s", studfile);
        fclose(bd);

        sterge(studfile, cauta(e[0].val));
//    printf("%d", cauta(e[0].val));
    end_html();
}

```

ADĂUGARE MATERIE

Dacă adăugarea unui student înseamnă adăugarea unei linii în baza de date, atunci adăugarea unei materii înseamnă adăugarea unei coloane (atribut) în baza de date.

ADAugARE MATERIE

Nume materie:

Adauga materieAnuleaza



Numele materiei este descris în formularul descris în pagina de mai jos:

adaug_mat.html

[illegible]

</html>

Fișierul C care face adăugarea unei materii în baza de date este:

adaug_mat.c

```
#include "util.h"
#include "html.h"

#define tempfile "temp.txt"
#define bd_cur "BD_cur.h"
#define MAXCAMPURI 100
#define sep "      "

char studfile[320];
int nr_campuri;
char campuri[MAXCAMPURI][50];

struct {
    char *val;
    char *name;
}e[20];
int cl, x, m;

void cit_campuri(FILE *dat){
    char *c, sir[320];
    int i;
    fseek(dat, 0L, SEEK_SET);
    fgets(sir, 320, dat);
    c=strtok(sir, sep);
    nr_campuri=0;

    while (c!=NULL){
        strcpy(campuri[++nr_campuri], c);
        c=strtok(NULL, sep);
    }

    fseek(dat, 0L, SEEK_END);
}

char *fara_spatii(char *s){
    char sir[320], *c;
    int i=0;
    c=strtok(s, sep);

    while (c!=NULL){
        if (!i) {sprintf(sir, "%s", c); i++;}
        else sprintf(sir, "%s_%s", sir, c);
        c=strtok(NULL, sep);
    }
    strcpy(s, sir);
    return (s);
}
```

```

void madaug(char *fisier, char *s){
    char l[320];
    FILE *data, *temp;

    if((data=fopen(fisier, "r"))==NULL){
        printf("Nu pot deschide fisierul Baza de Date");
        exit(0);
    }

    if((temp=fopen(tempfile, "w"))==NULL){
        printf("Nu pot folosi fisierul temporar");
        exit(0);
    }
    fgets(l, 320, data);
    l[strlen(l)-1]=0;
    fprintf(temp, "%s\t%s", l, s);
    while(!feof(data)){
        fgets(l, 320, data);
        if(!feof(data)) l[strlen(l)-1]=0;
        fprintf(temp, "\n%s\t0", l);
    }
    fclose(temp);
    fclose(data);

    if ((rename(tempfile, fisier))== -1) printf("Nu pot redenumi
    fisierul");
}

main(){
    FILE *bd, *data;

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();
    printf("\n<h1>Adaugare inregistrare <i>student</i></h1>\n");

    cl = atoi (getenv ("CONTENT_LENGTH"));
    for(x = 0; cl && (!feof(stdin)); x++){
        m = x;
        e[x].val = fmakeword(stdin, '&', &cl);
        plustospace (e[x].val);
        unescape_url (e[x].val);
        e[x].name = makeword (e[x].val, '=');
        // printf(" %d %s %s<br>", x, e[x].name, e[x].val);
    }

    if((bd=fopen(bd_cur, "r"))==NULL){
        printf("Nu pot deschide fisierul de setare al Bazei de Date
        curente");
        exit(0);
    }
    fscanff(bd, "%s", studfile);
    fclose(bd);

    if((data=fopen(studfile, "r"))==NULL){
        printf("Nu pot deschide fisierul Baza de Date");
    }
}

```

```

        exit(0);
    }
    cit_campuri(data);
    fclose(data);

    madaug(studfile, fara_spatii(e[0].val));

    printf("In BD %s adaug materia %s (sunt %d campuri)", studfile,
fara_spatii(e[0].val), nr_campuri);

    end_html();
}

```

MODIFICARE MATERIE

Modificarea unei materii se face tot pe baza unei liste de selecții care este generată în funcție de materiile introduse în baza de date.

Fișierul sursă al CGI-ului care generează pagina cu lista de selecție care conține materiile existente în baza de date este:

modificare_mat.c

```

#include <stdio.h>
#include <stdlib.h>
#include "html.h"

#define sep "      "
#define bd_cur "BD_cur.h"

char studfile[320];

void cit_materii(){
    FILE *data;
    char *c, linie[320];
    int nr_camp;

    data = fopen(studfile, "r");
    printf("<font color=#FFFFCC>Nume materie:</font>\n<select
name='nume'>\n");

```

```

nr_camp=0;
fgets(linie, 320, data);
c=strtok(linie, sep);

while (c!=NULL){
    nr_camp++;
    if ((nr_camp-5)>0) printf("<option >%s\n", c);
    c=strtok(NULL, sep);
}
printf("</select><br>\n");
fclose(data);
}

main(){
    FILE *bd;

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();
    if ((bd=fopen(bd_cur, "r"))==NULL){
        printf("Nu pot deschide fisierul de setare al Bazei de Date
curente");
        exit(0);
    }
    fscanf(bd, "%s", studfile);
    fclose(bd);
    printf("\n<h1>Modificare materie</h1>\n");
    printf("<form action='modif_mat.cgi' method='POST'>\n");
    printf("<blockquote>\n<table>\n<tr class=head><td>\n");
    cit_materii();
    printf("<br><input type='submit' value='Modifica
materie'>&nbsp;&nbsp;&nbsp;<input type='reset' value='Anuleaza'>\n");
    printf("</td></tr>\n</table>\n</blockquote>\n");
    printf("</form>\n");
    end_html();
}

```

Aceasta partea este încă în lucru și din acest motiv nu se vor prezenta în detaliu programele.

CONSULTARE BD (CITIRE NOTE)

Partea de consultare a bazei de date functionează pe pagina studenților de la Învățământ la Distanță.

CITIREA NOTELOR

Nume:	na3456
Parola:	*****
Login	Anuleaza



Pe baza unei parole și a unui nume de login, fiecare student poate intra să-și vadă notele obținute la materiile la care au dat examen.

NEGREANU-MAIOR ADRIAN

Bazele_matematice_ale_calculatoarelor 10
Algoritmica_si_programare 4
Birotica_si_tehnici_multimedia 10
Retele_de_calculatoare 0
Inteligenta_artificiala 0
Metode_numerice 0



Fișierul sursă al paginii Web care cuprinde formularul de logare este:

valid.html

```
<html>
<head>
<title>Citire note</title>
<link rel='STYLESHEET' type='text/css' href='general.css'>
</head>
<body>
<h1>Citirea notelor</h1>
<h2>Seria II</h2>
<form action='cgi-bin/valid.cgi?seria_2.txt' method='POST'>
  <blockquote>
    <table>
```

```
<tr class=head><td align='right'><font color='#FFFFCC'>Login:  
</font></td>  
    <td><input name='Login' type='text' width=25></td>  
</tr>  
<tr class=head><td align='right'><font color='#FFFFCC'>Parola:  
</font></td>  
    <td><input type='password' name='Parola' width=25></td>  
</tr>  
</table><br>  
<input type='submit' value='Login'>&nbsp&nbsp&nbsp  
<input type='reset' value='Anuleaza'>  
</blockquote>  
</form>  
<blockquote>  
    <a href='javascript:history.back()'>  
        <img src='back.gif' alt='Back' border=0>  
    </a>  
</blockquote>  
<p align="right">Creat de: <b>Sanda AVRAM</b><br>Coordonator:  
<b>Florian  
Mircea BOIAN</b></p>  
</body>  
</html>
```

Fișierul sursă CGI care tratează datele din formularul de mai sus este:

valid.c

```
#include "util.h"
#include "html.h"

#define MAXMATERII 20

int pozitie=0;
int nr_materii, ind;
char materii[MAXMATERII][50];

struct {
    char *val;
    char *name;
}e[20];
int cl, x, m;

void citire_studenti (char *datafile, char *login, char *parola){
    FILE *data;
    char nume[36], pren[36], nrmatr[12], nlog[12], pass[12], not,
nota[MAXMATERII][6];
    char linie[320], camp[64], a;
    int i, ind, nc, nl;

    if ((data = fopen(datafile, "r"))==NULL){
        printf("Nu pot deschide fisierul studenti!");
        exit(1);
    }
}
```

```

    ind=0;

    /// avem 5 campuri pe care le cunoastem: Nume, Prenume, Nr_matricol,
    Login, Parola
    nc=1; //numarul de campuri
    do{
        a=fgetc(data);
        if (a=='\t') {
            if (nc>5) materii[nc-5][nl]=0;
            nc++;
            nl=0; //numarul de litere
        }
        else if (nc>5) materii[nc-5][nl++]=a;
    }while(a!='\n');
    nr_materii=nc-5;

    fscanf(data, "%s", nume);
    fscanf(data, "%s", pren);
    fscanf(data, "%s", nrmatr);
    fscanf(data, "%s", nlog);
    fscanf(data, "%s", pass);
    for (i=1 ; i<=nr_materii; i++)
        fscanf (data, "%s", nota[i]);

    while (!(feof(data)||ind)){
        if((strcmp(nlog, login)==0)&&(strcasecmp(pass, parola)==0)){
            ind=1;
            printf("<h1>%s %s</h1>", nume, pren);
            for(i=1;i<=nr_materii;i++)
                printf("<br><i>%s</i> %s", materii[i], nota[i]);

        }
        fscanf(data, "%s", nume);
        fscanf(data, "%s", pren);
        fscanf(data, "%s", nrmatr);
        fscanf(data, "%s", nlog);
        fscanf(data, "%s", pass);
        for (i=1 ; i<=nr_materii; i++)
            fscanf (data, "%s", nota[i]);

    }
    if(!ind) printf("Nu exista utilizatorul <b>%s</b> sau parola e
    gresita!", login);
    fclose(data);
}

main(int argc, char **argv){
    int i, nota[MAXMATERII];

    printf("Content-type: text/html%c%c", 10, 10);
    begin_html();

    cl = atoi (getenv ("CONTENT_LENGTH"));
    for(x = 0; cl && (!feof(stdin)); x++){
        m = x;
        e[x].val = fmakeword(stdin, '&', &cl);
    }
}

```

```
        plustospace (e[x].val);
        unescape_url (e[x].val);
        e[x].name = makeword (e[x].val, '=');
        ///printf(" %d %s %s<br>", x, e[x].name, e[x].val);
    }
    if ((strlen(e[0].val)==0)|| (strlen(e[1].val)==0)) printf("Login
incorrect!");
    else{
        citire_studenti(argv[1], e[0].val, e[1].val);
    }
    end_html();
}
```

Anexa A

În această secțiune vom introduce câteva coduri folosite în scrierea paginilor HTML.

Caractere Speciale

&	&
<	<
>	>
"	"

Codurile caracterelor HTML

COD	DESCRIERE	SEMN	DESCRIERE IN ENGLEZA
�-	NEFOLOSIT		UNUSED
		TAB ORIZONTAL		HORIZONTAL TAB

	LINIE NOUA		LINE FEED
-	NEFOLOSIT		UNUSED
	CARRIAGE RETURN		CARRIAGE RETURN
-	NEFOLOSIT		UNUSED
 	SPATIU		SPACE
!	SEMN DE EXCLAMARE	!	EXCLAMATION MARK
"	GHILIMELE	“	QUOTATION MARK
#	DIEZ	#	NUMBER SIGN
$	DOLLAR	\$	DOLLAR SIGN
%	PROCENT	%	PERCENT SIGN
&	AMPERSAND	&	AMPERSAND
'	APOSTROF	‘	APOSTROPHE
(	PARANTEZA STANGA	(	LEFT PARENTHESIS
)	PARANTEZA DREAPTA	)	RIGHT PARENTHESIS
*	ASTERISC	*	ASTERISK
+	PLUS	+	PLUS SIGN
,	VIRGULA	,	COMMA
-	CRATIMA	-	HYPHEN
.	PUNCT	.	PERIOD (FULLSTOP)
/	SLASH	/	SOLIDUS (SLASH)
0-9	CIFRELE 0-9	0-9	DIGITS 0-9
:	DOUA PUNCTE	:	COLON
;	PUNCT SI VIRGULA	;	SEMI-COLON
<	MAI MIC	<	LESS THAN
=	EGAL	=	EQUALS SIGN
>	MAI MARE	>	GREATER THAN
?	SEMNUL INTREBARII	?	QUESTION MARK
@	SEMNUL COMERCIAL AT	@	COMMERCIAL AT
A-Z	LITERELE A-Z	A-Z	LETTERS A-Z
[	PARANTEZA PATRATA STANGA	[	LEFT SQUARE BRACKET
\	BACKSLASH	/	REVERSE SOLIDUS (BACKSLASH)
]	PARANTEZA PATRATA DREAPTA	]	RIGHT SQUARE BRACKET
^	CACIULITA	^	CARET

_	LINIE DE SUBLINIERE	—	HORIZONTAL BAR (UNDERSCORE)
`	ACCENT ACUT	`	ACUTE ACCENT
a-z	LITERELE A-Z	A-Z	LETTERS A-Z
{	ACOLADA STANGA	{	LEFT CURLY BRACE
|	BARA VERTICALA		VERTICAL BAR
}	ACOLADA DREAPTA	}	RIGHT CURLY BRACE
~	TILDA	~	TILDE
-Ÿ	NEFOLOSIT		UNUSED
 			NON-BREAKING SPACE
¡	SEMNUL ECLAMARII INTORS	¡	INVERTED EXCLAMATION
¢	CENT	¢	CENT SIGN
£	LIRA STERLINA	£	POUND STERLING
¤	MONEDA	¤	GENERAL CURRENCY SIGN
¥	YEN	¥	YEN SIGN
¦	BARA VERTICALA DESPARTITA		BROKEN VERTICAL BAR
§	SEMN DE DELIMITARE A UNEI SECTIUNI	§	SECTION SIGN
¨	UMLAUT	¨	UMLAUT (DIERESIS)
©	COPYRIGHT	©	COPYRIGHT
ª		ª	FEMININE ORDINAL
«	GHILIMELE DREPTE STANGI	«	LEFT ANGLE QUOTE, GUILLEMOTLEFT
¬	SEMN DE NEGARE	¬	NOT SIGN
­	CRATIMA MICA	-	SOFT HYPHEN
®	MARCA INREGISTRATA	®	REGISTERED TRADEMARK
¯	ACCENT MACRON	—	MACRON ACCENT
°	GRADE	°	DEGREE SIGN
±	PLUS MINUS	±	PLUS OR MINUS
²	DOI SUPRASCRIȘ	²	SUPERSCRİPT TWO
³	TREI SUPRASCRIȘ	³	SUPERSCRİPT THREE
´	ACCENT ACUT	´	ACUTE ACCENT
µ	SEMN MICRO	µ	MICRO SIGN
¶	SEMN DE DELIMITARE A PARAGRAFULUI	¶	PARAGRAPH SIGN
·	PUNCT MILOCIU	·	MIDDLE DOT
¸	CEDILLA	,	CEDILLA
¹	UNU SUPRASCRIȘ	¹	SUPERSCRİPT ONE
º		º	MASCULINE ORDINAL
»	GHILIMELE DREPTE DREPTE	»	RIGHT ANGLE QUOTE, GUILLEMOTRIGHT
¼	O PATRIME	¼	FRACTION ONE-FOURTH
½	O JUMATATE	½	FRACTION ONE-HALF
¾	TREI PATRIMI	¾	FRACTION THREE-FOURTHS
¿	SEMNUL INTREBARII INTORS	¿	INVERTED QUESTION MARK
À	A CU ACCENT GRAV	À	CAPITAL A, GRAVE ACCENT
Á	A CU ACCENT ACUT	Á	CAPITAL A, ACUTE ACCENT
Â	A CU ACCENT CIRCUMFLEX	Â	CAPITAL A, CIRCUMFLEX ACCENT
Ã	A TILDA	Ã	CAPITAL A, TILDE
Ä	A UMLAUT	Ä	CAPITAL A, DIERESIS OR UMLAUT MARK

Å	A RING	Å	CAPITAL A, RING
Æ	DIFTONGUL AE	Æ	CAPITAL AE DIPHTHONG (LIGATURE)
Ç	C CEDILLA	Ç	CAPITAL C, CEDILLA
È	E CU ACCENT GRAV	È	CAPITAL E, GRAVE ACCENT
É	E CU ACCENT ACUT	É	CAPITAL E, ACUTE ACCENT
Ê	E CU ACCENT CIRCUMFLEX	Ê	CAPITAL E, CIRCUMFLEX ACCENT
Ë	E UMLAUT	Ë	CAPITAL E, DIERESIS OR UMLAUT MARK
Ì	I CU ACCENT GRAV	Ì	CAPITAL I, GRAVE ACCENT
Í	I CU ACCENT ACUT	Í	CAPITAL I, ACUTE ACCENT
Î	I CU ACCENT CIRCUMFLEX	Î	CAPITAL I, CIRCUMFLEX ACCENT
Ï	I UMLAUT	Ï	CAPITAL I, DIERESIS OR UMLAUT MARK
Ð	ICELANDIC	Ð	CAPITAL ETH, ICELANDIC
Ñ	N TILDA	Ñ	CAPITAL N, TILDE
Ò	O CU ACCENT GRAV	Ò	CAPITAL O, GRAVE ACCENT
Ó	O CU ACCENT ACUT	Ó	CAPITAL O, ACUTE ACCENT
Ô	O CU ACCENT CIRCUMFLEX	Ô	CAPITAL O, CIRCUMFLEX ACCENT
Õ	O TILDA	Õ	CAPITAL O, TILDE
Ö	O UMLAUT	Ö	CAPITAL O, DIERESIS OR UMLAUT MARK
×	SEMN DE MULTIPLICARE	×	MULTIPLY SIGN
Ø	MULTIMEA VIDA	Ø	CAPITAL O, SLASH
Ù	U CU ACCENT GRAV	Ù	CAPITAL U, GRAVE ACCENT
Ú	U CU ACCENT ACUT	Ú	CAPITAL U, ACUTE ACCENT
Û	U CU ACCENT CIRCUMFLEX	Û	CAPITAL U, CIRCUMFLEX ACCENT
Ü	U UMLAUT	Ü	CAPITAL U, DIERESIS OR UMLAUT MARK
Ý	Y CU ACCENT ACUT	Ý	CAPITAL Y, ACUTE ACCENT
Þ	THORN	Þ	CAPITAL THORN, ICELANDIC
ß	SZ	ß	SMALL SHARP S, GERMAN (SZ LIGATURE)
à	A CU ACCENT GRAV	À	SMALL A, GRAVE ACCENT
á	A CU ACCENT ACUT	Á	SMALL A, ACUTE ACCENT
â	A CU ACCENT CIRCUMFLEX	Â	SMALL A, CIRCUMFLEX ACCENT
ã	A TILDA	Ã	SMALL A, TILDE
ä	A UMLAUT	Ä	SMALL A, DIERESIS OR UMLAUT MARK
å	A RING	Å	SMALL A, RING
æ	AE	Æ	SMALL AE DIPHTHONG (LIGATURE)
ç	C TILDA	Ç	SMALL C, CEDILLA
è	E CU ACCENT GRAV	È	SMALL E, GRAVE ACCENT
é	E CU ACCENT ACUT	É	SMALL E, ACUTE ACCENT
ê	E CU ACCENT CIRCUMFLEX	Ê	SMALL E, CIRCUMFLEX ACCENT
ë	E UMLAUT	Ë	SMALL E, DIERESIS OR UMLAUT MARK
ì	I CU ACCENT GRAV	Ì	SMALL I, GRAVE ACCENT
í	I CU ACCENT ACUT	Í	SMALL I, ACUTE ACCENT

î	I CU ACCENT CIRCUMFLEX	î	SMALL I, CIRCUMFLEX ACCENT
ï	I UMLAUT	ï	SMALL I, DIERESIS OR UMLAUT MARK
ð	ETH	ð	SMALL ETH, ICELANDIC
ñ	N TILDA	ñ	SMALL N, TILDE
ò	O CU ACCENT GRAV	ò	SMALL O, GRAVE ACCENT
ó	O CU ACCENT ACUT	ó	SMALL O, ACUTE ACCENT
ô	O CU ACCENT CIRCUMFLEX	ô	SMALL O, CIRCUMFLEX ACCENT
õ	O TILDA	õ	SMALL O, TILDE
ö	O UMLAUT	ö	SMALL O, DIERESIS OR UMLAUT MARK
÷	SEMNUL IMPARTIRII	÷	DIVISION SIGN
ø	MULTIMEA VIDA (MICA)	ø	SMALL O, SLASH
ù	U CU ACCENT GRAV	ù	SMALL U, GRAVE ACCENT
ú	U CU ACCENT ACUT	ú	SMALL U, ACUTE ACCENT
û	U CU ACCENT CIRCUMFLEX	û	SMALL U, CIRCUMFLEX ACCENT
ü	U UMLAUT	ü	SMALL U, DIERESIS OR UMLAUT MARK
ý	Y CU ACCENT ACUT	ý	SMALL Y, ACUTE ACCENT
þ	THORN	þ	SMALL THORN, ICELANDIC
ÿ	Y UMLAUT	ÿ	SMALL Y, DIERESIS OR UMLAUT MARK

Numele culorilor

black	=	#000000	navy	=	#000080
blue	=	#0000FF	dark green	=	#006400
deep sky blue	=	#00BFFF	dark turquoise	=	#00CED1
green	=	#00FF00	spring green	=	#00FF7F
cyan	=	#00FFFF	midnight blue	=	#191970
dodger blue	=	#1E90FF	light sea green	=	#20B2AA
forest green	=	#228B22	sea green	=	#2E8B57
dark slate gray	=	#2F4F4F	lime green	=	#32CD32
medium sea green	=	#3CB371	turquoise	=	#40E0D0
dark slate blue	=	#483D8B	medium turquoise	=	#48D1CC
dark olive green	=	#556B2F	cadet blue	=	#5F9EA0
cornflower blue	=	#6495ED	medium aquamarine	=	#66CDAA
dim gray	=	#696969	slate blue	=	#6A5ACD
olive drab	=	#6B8E23	slate gray	=	#708090
light slate gray	=	#778899	medium slate blue	=	#7B68EE
lawn green	=	#7CFC00	chartreuse	=	#7FFF00
aquamarine	=	#7FFFD4	light slate blue	=	#8470FF
blue violet	=	#8A2BE2	saddle brown	=	#8B4513
dark sea green	=	#8FBC8F	pale green	=	#98FB98
yellow green	=	#9ACD32	brown	=	#A52A2A
light blue	=	#ADD8E6	green yellow	=	#ADFF2F
pale turquoise	=	#AFEEEE	maroon	=	#B0E0E6
firebrick	=	#B22222	powder blue	=	#B8860B
medium orchid	=	#BA55D3	dark goldenrod	=	#BC8F8F
dark khaki	=	#BDB76B	rosy brown	=	#BEBE8E
medium violet red	=	#C71585	gray	=	#D0D0D0
chocolate	=	#D2691E	violet red	=	#D81B60

light gray	=	#D3D3D3	tan	=	#DAA520
pale violet red	=	#DB7093	goldenrod	=	#DDA0DD
burlywood	=	#DEB887	plum	=	#E066FF
light cyan	=	#E0FFFF	lavender	=	#E6E6FA
dark salmon	=	#E9967A	violet	=	#EE82EE
light coral	=	#F08080	khaki	=	#F0E68C
alice blue	=	#F0F8FF	honeydew	=	#F0FFF0
azure	=	#F0FFFF	sandy brown	=	#F4A460
wheat	=	#F5DEB3	beige	=	#F5F5DC
white smoke	=	#F5F5F5	mint cream	=	#F5FFFA
ghost white	=	#F8F8FF	salmon	=	#FA8072
antique white	=	#FAEBD7	linen	=	#FAF0E6
old lace	=	#FDF5E6	red	=	#FF0000
magenta	=	#FF00FF	deep pink	=	#FF1493
tomato	=	#FF6347	hot pink	=	#FF69B4
coral	=	#FF7F50	orange	=	#FFA500
light pink	=	#FFB6C1	gold	=	#FFD700
moccasin	=	#FFE4B5	seashell	=	#FFF5EE
yellow	=	#FFFF00	white	=	#FFFFFF

Anexa B

Coduri HTTP

COD	INDICAȚIE
2xx	Success
200	OK; the request was fulfilled.
201	OK; following a POST command.
202	OK; accepted for processing, but processing is not completed.
203	OK; partial information-the returned information is only partial.
204	OK; no response-request received but no information exists to send back.
3xx	Redirectare
300	OK; Moved-the data requested has a new location and the change is permanent.
301	Found-the data requested has a different URL temporarily.
302	Method-under discussion, a suggestion for the client to try another location.
303	Not Modified-the document has not been modified as expected.
4xx	Erori datorate clientului
400	Bad request-syntax problem in the request or it could not be satisfied.
401	Unauthorized-the client is not authorized to access data.
402	Payment granted-indicates a changing scheme is in effect.
403	Forbidden-access not granted even with authorization.
404	Not found-server could not find the given resource.
5xx	Erori datorate serverului
500	Internal Error-the server could not fulfill the request because of an unexpected condition.
501	Not implemented-the server does not support the facility requested.
502	Server overloaded-high load (or servicing) in progress.
503	Gateway timeout-server waited for another service that did not completed in time.

Anexa C

Variabilele de mediu folosite de un CGI

SERVER_SOFTWARE	The name and the Versiune number of the server.
SERVER_NAME	The server's full host name, IP address, or an alias.
GATEWAY_INTERFACE	Revision of the CGI specification.
SERVER_PROTOCOL	The name and revision of the service protocol.
SERVER_PORT	The port where the server is running.
REQUEST_METHOD	The method of the request.
PATH_INFO	Extra information at the end of the path of the executing CGI script.
PATH_TRANSLATED	A translated Versiune of PATH_INFO that has removed any virtual mappings.
SCRIPT_NAME	The full virtual path of the executing CGI script.
QUERY_STRING	Anything that goes after the ? in the URL that referenced this CGI program.
REMOTE_HOST	The hostname of the machine making the request to the Webserver.
REMOTE_ADDR	The IP address of REMOTE_HOST or the machine making the request.
AUTH_TYPE	The authentication method used if the script was protected.
REMOTE_USER	Login name of the user from REMOTE_ADDR if the user logged in via user-authentication.
REMOTE_IDENT	Login name of the user from REMOTE_ADDR if the remote host supports user identification.
CONTENT_TYPE	Content type of the data.
CONTENT_LENGTH	Length of the content given by the client.
HTTP_ACCEPT	The MIME type that the client accepts.
HTTP_USER_AGENT	The browser from the client.
HTTP_REFERER	The page that refers the client to this CGI program.
REDIRECT_REQUEST	The request for a redirect as sent to the server.
REDIRECT_STATUS	The status the server would have sent if it hadn't been redirected.
REDIRECT_URL	The URL that caused the redirect.

Headerele HTTP folosite de programele CGI

Allowed	Lists the requests that the user is allowed to use on this URL.
Content-Encoding	Encoding method.
Content-Language	ISO3316 language code with an optional ISO639 country code.
Content-Length	The length of the returning document.
Content-Transfer-Encoding	The MIME encoding method used on the returning document.
Content-Type	MIME type of the returning document.
Date	Creation date in GMT format.
Expires	Expiration date.
Last-Modified	Last modification date.
Location	Virtual path or the URL of the returning document.
Message-Id	Message identifier.
Public	Lists all requests that anyone can use.
Status	Returned status of the request.
URI	URI of the document.
Versiune	Versiune of the document.
Title	Title of the document.

Bibliografie

- [1] Scott Zimmerman si Tim Evans, **Building an Intranet with Windows NT 4**, 1996
- [2] Jerry Ablan, **Developing Intranet Applications With Java**, 1996
- [3] Preston Gralla si Mina Reimer, **How Intranets Work**, 1996
- [4] David Garrett, **Intranets Unleashed**, 1996
- [5] Rich Casselberry, **Running a Perfect Intranet**, 1996
- [6] Shel Holtz, **The Intranet Advantage**, 1996
- [7] John December si Mark Ginsburg, **HTML 3.2 and CGI Professional Reference Edition UNLEASHED**, 1996
- [8] Lab Wab Ooi si Billy Baroon, **HTML & CGI Quick Reference**, 1997
- [9] Eugene Eric Kim, **CGI Developer's Guide**, 1996
- [10] Mark Robbin Brown si Jerry Honeycutt, **Using HTML 4 (Special Edition)**, 1997