Prefață

Cartea își propune să-i inițieze pe cei care doresc să cunoască realizarea paginilor Web dinamice folosind tehnologia Microsoft. Se prezintă elementele de bază pentru realizarea paginilor statice cu HTML. Urmează limbajul Visual Basic Script, apoi, în final, tehnologia ASP și obiectele ei. Deoarece ASP poate interacționa cu bazele de date relaționale, cartea prezintă conceptele specifice bazelor de date relaționale, într-o formă simplificată, împreună cu modalitatea de manipulare a datelor unei baze de date Microsoft Access prin ADO.

Aduc mulțumiri artistului plastic clujean Marin Leschian pentru lucrările "Vacanță eternă" (copertă față) și "Armata lui Don Quijote" (coperta spate) pe care le-am folosit pentru a îmbrăca această carte.

Multe mulțumiri Ramonei pentru lectura și corectura manuscrisului.

Această ediție mai conține, față de prima, o introducere în rețele de calculatoare, explicarea conceptului de foaie de stil, o mai clară descriere a conceptelor legate de bazele de date relaționale și mai multe exemple.

http://www.east.utcluj.ro/mb/mep/antal

Universitatea Tehnica din Cluj-Napoca
Catedra Mecanica si Programare
Conf. Dr. Ing. ANTAL Tiberiu Alexandru

Celor care încă mai caută deşi, deja, au cam obosit.

http://www.east.utcluj.ro/mb/mep/antal

Universitatea Tehnica din Cluj-Napoca
Catedra Mecanica si Programare
Conf. Dr. Ing. ANTAL Tiberiu Alexandru

CUPRINS

Concepte de reţele de calculatoare şi protocoale de reţea

În domeniul rețelelor de calculatoare, termenul **dispozitiv utilizator** este folosit pentru a defini un echipament electric care este conectat la rețea (calculator, imprimantă, scanner etc.) în scopul schimbului de date cu un alt echipament al rețelei. Denumirea e dată din punctul de vedere al utilizatorului de rețea (de exemplu dacă discuția nu ar fi la nivel de rețea ci de calculator, dispozitive ar putea fi: disc, placă video, modem etc.). Termenul **dispozitiv de rețea** este folosit pentru a descrie echipamentele electrice necesare pentru interconectarea dispozitivele utilizator în vederea schimbului de date. Aceste dispozitive sunt conectate prin cabluri electrice (sau, direct, prin unde radio - wireless) şi asigură transmisia, recepția, gestionarea datelor ce se doresc a fi transferate între dispozitivele utilizator. Câteva dintre numele de dispozitive vehiculate mai des sunt prezentate, pe scurt, în continuare:

Repeater (repetor) - dispozitiv de rețea care se foloseşte pentru refacerea sau repetarea semnalelor (există repetoare care pot face retransmisia datelor între două rețele care folosesc protocoale diferite).

Hub (concentrator) - punct de conectare comun la nivelul unei rețele, se folosesc pentru interconectarea segmentelor (segmentul este o porțiune de rețea între două bridge-uri, router-e sau switch-uri) de LAN. Hub-ul are mai multe porturi (interfețe de plăci de rețea). Atunci când un pachet (mesajele de transmis sunt divizate în pachete, fiecare pachet se transmite individual şi poate parcurge o rută individuală către destinație) ajunge pe un anumit port se copiază pe toate celelte porturi pentru ca toate segmentele de LAN să-l vadă. Există şi hub-uri inteligente care citesc adresa destinație a pachetelor şi le trimit numai unui singur port, acestea mai pot avea şi diferite facilități de monitorizare şi administrare a traficului.

Bridge (punte) - dispozitiv de rețea ce conectează două LAN-uri sau două segmente de LAN ce folosesc acelaşi protocol (de exemplu, Ethernet).

Switch (comutator) - dispozitiv de rețea ce filtrează şi expediază pachete între două segmente de LAN. Switch-ul operează la nivelul legăturii de date (nivelul 2 OSI - Open System Interconnection) eventual la nivelul numit rețea (nivelul 3 OSI).

Router - un dispozitiv ce transferă pachete de date între rețele distincte. Tipic, router-ul este conectat la două LAN-uri sau WAN-uri respectiv între un LAN şi furnizorul de servicii Internet. Router-ele realizează dirijarea eficientă a pachetelor de date pe liniile cele mai convenabile.

Gateway - un calculator din rețea care se foloseşte pentru accesarea unei alte rețele, deoarece este unicul punct de interacțiune cu alte rețele, deseori, va fi şi proxy server şi firewall; dacă rețeaua este subnetată (are subrețele) este router-ul care transferă traficul în afara subrețelei din care face parte stația transmițătoare.

Proxy server - un server (un calculator care gestionează anumite resurse) care este "aşezat" între un client de aplicație, de exemplu un navigator de Web, şi un server real, de exemplu un server de Web. Proxy server-ul va intercepta cererile către server-ul real şi va încerca să le îndeplinească singur, dacă nu poate transferă cerererile server-ului real. Avantajele folosirii unui proxy stau în creşterea performanțelor (răspunsul este dat imediat, nu mai ajunge la server-ul real care este bombardat de mii de cereri şi cam ocupat cu ele) şi filtrarea cererilor.

Firewall - un sistem electric sau o aplicație ce gestionează accesul (filtrează) la sau de la o rețea privată.

Modalităţi de clasificare a reţelelor

Există multe criterii de clasificare a rețelelor, iată câteva posibilități:

- extinderea geografică: LAN, MAN, WAN etc.;
- topologie: stea, magistrală, inel etc.;
- mediul de transmisie: cupru, fibră optică, radio, microunde, satelit;
- proprietar: public, privat;
- tehnologia folosită: comuntată (switched) / conexiune permanentă (permanent links), cu legătură fizică (physical link) sau virtuală (virtual link), orientată pe conexiune (connection-oriented) sau fără conexiune (connectionless), cu difuzare (broadcast) sau punct la punct (point-to-point) etc.;
- viteză: bandă largă (broadband), bandă îngustă (narrowband).

Nu voi discuta toate tipurile posibile, voi aminti însă cele mai uzuale denumiri.

Reţele LAN (Local Area Networks)

Rețeaua locală permite interconectarea unor calculatoare aflate la distantă mică (un birou sau mai multe birouri) în scopul partajării fișierelor din calculatoare, a imprimatelor și a comunicației locale. Multe dintre LAN-urile actuale folosesc uzual tehnologia Ethernet. Alte tehnologii folosite pentru rețele LAN sunt Token Ring și FDDI.

Reţele WAN (Wide Area Networks)

Rețeaua WAN este, tipic, raspândită pe o arie geografică mai mare decât LAN-ul (distanța între procesoare este de ordinul kilometrilor sau mai mult). WAN-ul se formează prin conectarea mai multor LAN-uri și asigură accesul la calculatoarele locale, server-e, imprimate de la o distanță mare respectiv comunicația între calculatoare. Câteva dintre tehnologiile specifice WAN-urilor sunt: modem-urile, ISDN (Integrated Service Digital Network), DSL (Digital Subscriber Line), SONET (Synchronous Optical Network).

Reţele MAN (Metropolitan Area Networks)

MAN este o rețea de date ce are o răspândire geografică între LAN și WAN, cel mai des la nivelul unui oraș. Majoritatea MAN-urilor beneficiază de conexiuni private dar foarte rapide pe fibră optică sau alte medii de comexiune digitală rapidă. Se formează prin legarea mai multor LAN-uri, eventual și prin unde radio pentru a oferi aceste servicii în spații publice (aeroport, magazine, etc.)

Reţele VPN (Virtual Private Networks)

VPN este o rețea care folosește un mediu de transmisie public pentru a forma o rețea privată de calculatoare. Oricine are acces la această rețea va putea lucra cu toate facilitățile ei, în condiții de securitate bună, chiar dacă se află la distanță fizică mare de rețea și se leagă la ea folosind o conexiune clasică la Internet (care este nesecurizată). Diferite metode de criptare și mecanisme de securizare sunt folosite pentru a permite doar accesul utilizatorilor autorizați la rețea și pentru evitarea interceptării datelor.

Condiţiile de conectare a unui calculator la o reţea

Pentru ca un calculator să poată fi legat la o rețea, prin care să putem avea acces la Internet și la WWW, acesta trebuie să:

- aibă cel puțin o placă de rețea;
- folosească un sistem de operare ce știe lucra cu protocoale TCP/IP (Transmission Control Protocol/Internet Protocol);
- aibă instalate aplicațiile ce interpretează și afișează datele transferate prin rețea într-o formă inteligibilă pentru oameni.

Placa de rețea

Pentru conectarea unui calculator la o rețea de calculatoare acesta trebuie să conțină, la nivel fizic, un dispozitiv de rețea numit placa de rețea (NIC - Network Interface Card). Acesta asigură comunicația electrică într-o rețea. Placa de rețea folosește o linie de întrerupere (IRQ - Interrupt ReQuest), un spațiu de adrese I/O (Input/Output - Intrare/Ieșire) și o zonă de RAM (Random Access Memory - memorie volatilă) pentru a interacționa cu SO (Sistemul de Operare). O linie de întrerupere este o linie electrică specială în calculator prin care acesta poate fi oprit din funcționare normală de către un dispozitiv care transmite un semnal electric pe linia respectivă. Semnalul întrerupe temporar funcționarea curentă a calculatorului pentru ca acesta să poate decidă ce urmează să facă. Daca procesul care prezintă date este rapid, atunci calculatorul va prelua datele, altfel acestea riscă sa se piardă, după care își va relua activitatea din care a fost oprit. Deoarce mai multe semnale de întrerupere pot sosi pe rând sau simultan pe o singură linie de întrerupere, fiecărui dispozitiv ce poate întrerupe calculatorul trebuie să i se dea un număr unic pentru a putea fi identificat de calculator. Această valoare este specificată atunci când respectivul dispozitiv este intodus fizic în calculator și se poate face manual sau automat (asta dacă dispozitivul este PnP - Plug and Play). Placa de rețea este un astfel de dispozitiv ce trebuie să primească un număr de IRQ. Aceasta trebuie instalată pe orice echipament ce va accesa rețeaua deoarece asigură cuplarea la nivel electric cu aceasta. Există mai multe tipuri de plăci de rețea:

- notebook-urile pot avea placa de rețea încorporată sau atașata extern pe prin PCMCIA;
- PC-urile pot avea și ele placă de bază cu placă de rețea încorporată sau pot fi atașate ca plăci distincte.

Fiecare placă de rețea are un număr de identificare unic numit adresă MAC (Media Access Control). Acest număr este folosit (de exemplu, în foarte răspândita tehnologie de rețea Ethernet - o arhitectură de rețea locală dezvoltată de firma Xerox) pentru a derula comunicația de date a stației (numită host) prin rețea.

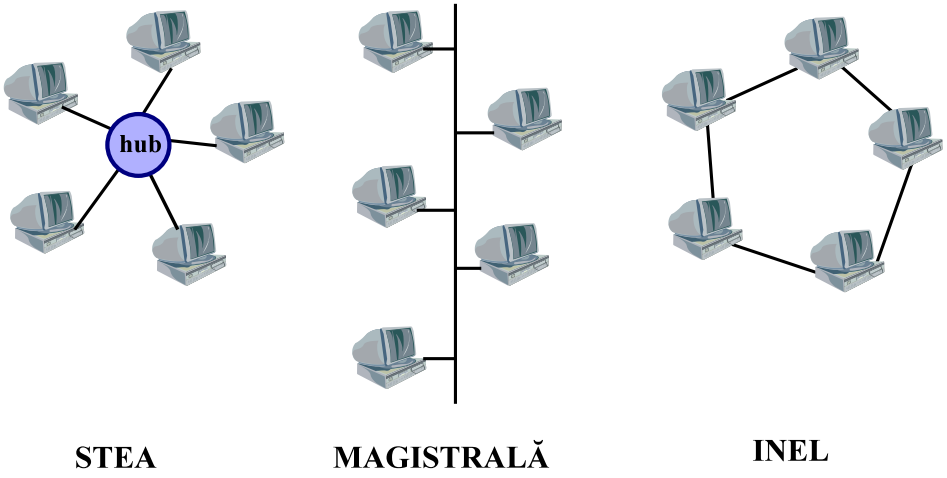
Rețeaua Ethernet apare în 1976, trecând ulterior prin mai multe revizii și standardizări. Aceasta folosea pentru conectarea fizică a plăcilor de rețea un singur cablu coaxial (cablul de rețea). Datele transmise în această rețea erau sparte în pachete. Denumirea de pachet este, azi, un termen generic, reprezentând unitatea de date ce se transmite într-o rețea. Pachetele de date erau transmise pe baza unui algoritm CSMA/CD (Carrier Sense Multiple Access / Collision Detection). În situația în care mai multe pachete de date încercau să fie transmise simulan de calculatoare diferite ale reței pe cablul de rețea apărea o coleziune (coleziune era detectată și pachetul era retransmis după un timp aleator). Implementarea unei astfel de rețele este simplă, dar ca urmare a competiției calculatoarelor în vederea accesului la cablul de rețea (mediul de transmisie), caracteristicile de transmisie nu sunt deterministe. Viteza tipică a transferului de date pe această rețea era de 10 Mbps. Ca urmare a simplității, tehnologia Ethernet a evoluat repede mărindu-și viteza transferului de date la 100 Mbps (Fast Ethernet) respectiv 1 Gbps (Gigabit Ethernet).

Topologii de rețea

Topologia se referă la modul de aranjare a legăturilor fizice între calculatoarele rețelei. Câteva dintre ele se văd în **Figura 1** (star - stea, bus - magistrală, ring - inel). Topologia stea conectează toate cablurile (pe fiecare cablu este un singur calculator) la un singur punct central (hub-ul). Topologia magistrală folosește un singur cablu pe care se leagă toate calculatoarele, cea inel leagă două câte două calculatoare și primul cu ultimul rezultând o rețea de cablu în forma inelară. Termenul de topologie logică este folosit pentru de defini modul de comunicație în rețea și poate

fi cu difuzare (broadcast) sau cu transfer de jetoane (token passing). În cazul difuzării, un calculator transmite mesaje către toate celelalte calculatoare ale rețelei, prin cablul de rețea.

Figure 1 - Topologii de rețea



Nu există o ordine specifică în care calculatoarele folosesc mediul de transmisie, din acest motiv pot să apară și conflicte (după cum am spus deja, Ethernet funcționeză după acest model). În cazul lucrului cu jeton, acesta este transmis secvențial fiecărui calculator din rețea. Calculatoul care are jetonul (token-ul) poate transmite date în rețea, dacă acesta nu are date de transmis va pasa jetonul mai departe (nu-l ține). Acest proces se repetă cu parcurgerea tuturor calculatoarelor din rețea. TokenRing este o rețea care lucrează după această procedură.

Despre protocoale de rețea

Rețelele de comunicație între calculatoare pot fi organizate ca o grupare de protocoale aproape independente, fiecare dintre acestea operând la un anumit nivel. Tipic, nivelul cel mai de jos asigură comunicația fizică între calculatoare, în timp ce nivelul cel mai înalt constă în aplicațiile utilizatorului. Fiecare nivel curent, se folosește de nivelul inferior, pentru a pune la dispoziția nivelului superior, un grup de servicii. La fiecare nivel, programe rulate pe calculatoarele rețelei folosesc protocoale corespunzătoare nivelului pentru a comunica între ele. Avantajul protocoalelor pe nivele constă în specificațiile clare ale metodelor de transmitere a informațiilor de la un nivel la altul ca parte a protocolului. Astfel, orice modificare la nivelul protocolului nu afectează celelalte nivele, această independență simplificând proiectarea și întreținerea programelor de comunicație.

Pentru cele ce urmează voi considera că suita de procoale TCP/IP este organizată pe 5 nivele de protocoale. Prezentarea este mult simplificată fiind menținute numai acele concepte care au fost deja discutate, respectiv cele ce vor fi utilizate mai departe la Web și Internet. Anumitor nivele le corespund anumite concepte și dispozitive de rețea, după cum urmează:

Nivel TCP/IP	Descriere	Dispozitive de rețea și concepte specifice nivelului
5 - Aplicație	Asigură transparența rețelei, alocarea de resurse etc. la nivelul utilizatorului rețelei.	

4 - Transport (TCP)	Asigură realizarea unei conexiuni un punct A şi unul B, pentru ca datele să ajungă fără erori şi în ordinea corectă.	
3 - Internet (IP)	Asigură determinarea rutei pachetelor de date de la transmițător la receptor.	router, subrețea, adresă IP
2 - Placă de rețea	Asigură fragmentarea datelor în pachete ce conțin informații de indentificare şi urmărire în vederea transmiterii la nivelul fizic.	switch, adresă MAC, Ethernet
1 - Fizic	Corespunde conexiunilor electrice şi mecanice.	hub

Introducere în TCP/IP

Aceast protocol a devenit interesant atunci când proprietarii de LAN-uri au înțeles că poate fi utilizat atât pentru transportul datelor într-o LAN cât şi pentru interconectarea mai multor LAN-uri. La interconectarea LAN-urilor datele pot fi transferate între calculatoarele unor LAN-uri distincte, dar numai cu ajutorul unor dispozitive de rețea de tipul router sau gateway. TCP este un protocol de comunicație sigur (garantează că toate datele ajung la destinație fără erori, în ordinea în care s-au trimis), asigură controlul fluxurilor (atunci când receptorul nu mai poate accepta date acesta va opri transmisia cu un semnal, iar când recepția devine din nou posibilă o va reporni cu un semnal), este multiplexat (mai multe semnale electrice sunt combinate pentru a partaja mediul de transmisie respectiv cablul de rețea) şi orientat pe conexiune (transportul de date între calculatoare se face într-un flux continuu, în trei faze bine definite: stabilirea conexiunii, transferul de date şi eliberarea conexiunii). Protocolul IP este fără conexiune, ce lucrează cu priorități (tipurilor de trafic li se atribuie priorități, în funcție de prioritate traficul poate sau nu tolera întârzieri, de exemplu videoconferințele trebuie să "meargă" în timp real, în timp ce e-mail-ul poate fi întârziat) şi comutare de pachete (pachetele de date sunt expediate individual între calculatoare, fără existența unei rute prestabilite). IP asigură transportul pachetelor pe diferite rute, fragmentarea (proces prin care un pachet este divizat în porțiuni mai mici numite fragmente, cu scopul adaptării acestora la cerințele rețelei fizice prin care pachetele trebuie să treacă; procesul invers se numeşte reasamblare) şi reasamblarea pachetelor.

Configurarea conexiuni TCP/IP în Windows XP şi 2000

Sistemul de operare Windows trebuie să fie configurat să lucreze cu TCP/IP pe fiecare calculator al rețelei pentru ca să poate exista o comunicație între acestea. Pentru

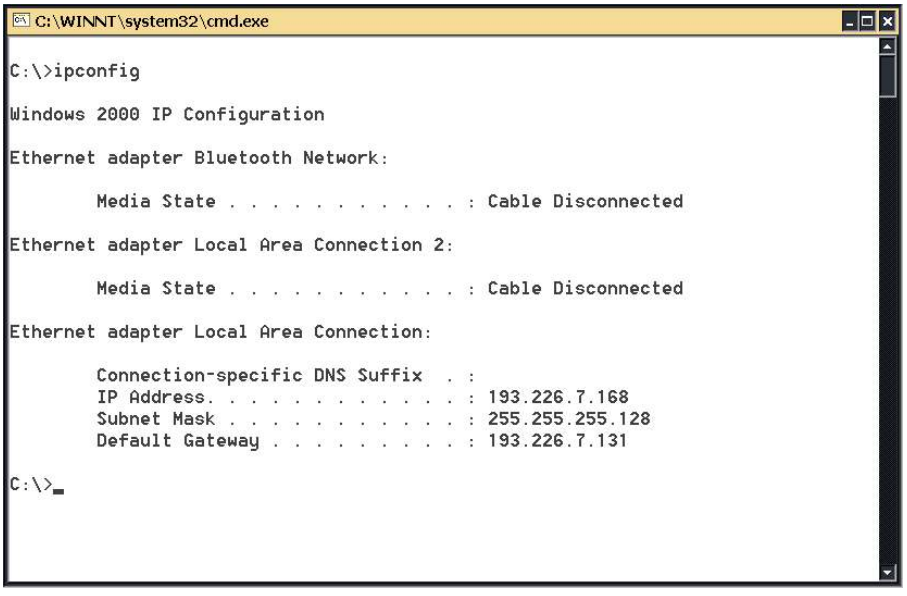


Figure 2 - Start>Run>Cmd>ipconfig

vizualizarea setărilor TCP/IP, în Windows 95, 98 şi Me se foloseşte aplicaţia **`winipcfg`**, în NT/2000 şi XP se foloseşte **`ipconfig`**.

Calculatorul trebuie să fie conectat la rețea pentru ca datele (vezi **Figura 2**) cu privire la conexiune să poată fi afişate.

Pentru ca un grup de calculatoare să fie într-o rețea locala trebuie ca la:

- **IP Address**: primele 3 grupe de numere să fie aceleaşi, iar numerele din ultima grupă se fie, toate, distincte;
- **Subnet Mask**: să fie aceeaşi;
- **Default Gateway**: să fie aceeaşi.

Funcționarea conexiunii la rețea se face cu aplicaţia **`ping`** (Packet InterNet Groper). Aceasta verifică dacă o adresă de IP există şi dacă acceptă cereri. `ping` lucrează trimiţând un pachet special numit ICMP (Internet Control Message Protocol Echo Request) către o anumită destinaţie. Fiecare pachet trimis este o cerere de răspuns, `ping` se foloseşte pentru a verifica dacă funcţiile de transmisie şi recepţie ale NIC, configurarea TCP/IP şi a rețelei sunt funcționale.

`ping 127.0.0.1` - 127.0.0.1 este o adresă rezervată pentru testare a conexiunii între client şi server (loop back address test). Aceasta permite realizarea unei conexiuni între client şi server prin TCP/IP pe acelaşi calculator (nu este nevoie de alte elemente de rețea). Fiecare calculator are adresa 127.0.0.1 asignată interfeţei de loop back care mai poartă şi numele de "local host", majoritatea NIC-urilor au implementată această buclă de testare internă. Orice transmisie pe acest port nu iese din NIC pe cablul de rețea ci este redirectată spre interfaţa de testare şi ajunge din nou în coada de intrare ip. Se emite o cerere de ecou și un răspuns la acesta prin mesaj ICMP prin care se verifică condiţiile de funcționare ale NIC-ului şi a instalării respectiv ale configurării corecte ale TCP/IP.

`ping IP_adresa_calc_local` - verifică dacă configurarea TCP/IP a rețelei este corectă, adică dacă IP-ul este adăugat corect rețelei locale şi dacă dispozitivul de rețea la care este legat calculatorul este activ (pornit).

`ping IP_adresa_default_gateway` - verifică dacă conexiunea cu router-ul (default gateway-ul) funcționează corect şi dacă se poate face legătura cu alte calculatoare din rețeaua locală.

`ping IP_adresa_calc_alta_retea` - verifică dacă se poate comunica prin router cu calculatorul aflat la distanță într-o altă rețea.

Dacă totul funcţionează bine ping întoarce adresa

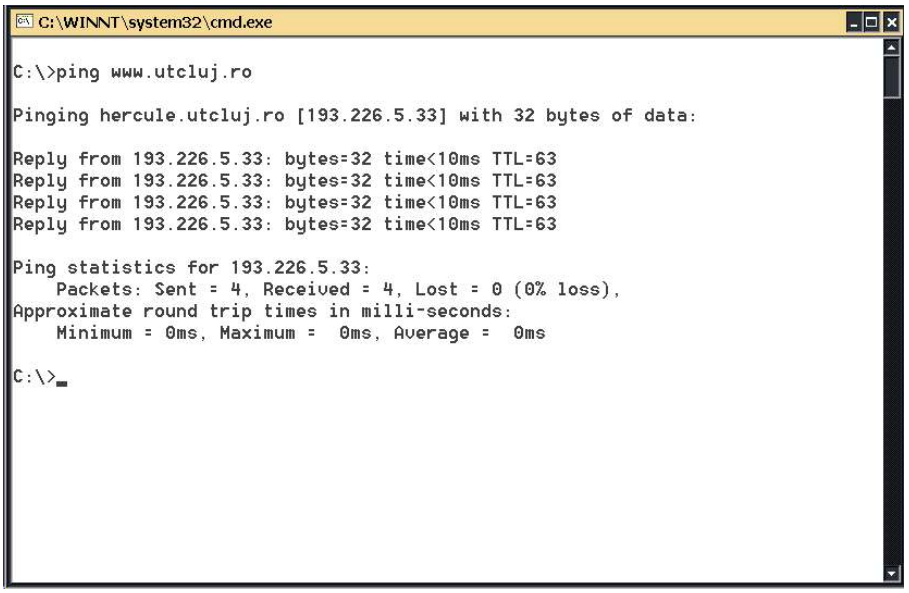


Figure 3 - ping www.utcluj.ro

verificată. Dacă răspunsul este “Request timed out” atunci nu s-a primit răspuns în timpul alocat de la calculatorul ping-uit. Durata de timp alocată aşteptării răspunsului se modifică folosind opțiunea w, de exemplu pentru aşteptare de 5 secunde scriem `ping -w 5000 adr_IP`.

Pe Windows XP şi 2000 mai puteți utiliza şi `pathping` care face mult mai multe (determină IP-ul unui calculator dintr-o altă rețea, întârzierile şi pierderile de pachete de date).

În rețelele bazate pe potocoalele TCP/IP (aici este inclus şi Internet-ul) calculatoarele se mai numesc şi gazde (host), fiecare gazdă trebuie să aibă o adresă de IP-unică. TCP permite stabilirea unei conexiuni între două host-uri cu scopul interschimbării unor fluxuri de date. TCP garantează transferul de date şi recepția pachetelor în aceeaşi ordine în care s-au transmis. Protocolul IP se ocupă numai de pachete. Aici este specificat formatul pachetelor de date (numite şi datagrame) şi schema de adresare. IP se aseamănă cu sistemul poştal. Permite adresarea şi punerea unui pachet la poştă, dar fără să facă o legătură directă între expeditor şi destinatar. TCP este cel care face această legătură virtuală între sursă şi destinație. Împreună, TCP/IP, permit stabilirea unei conexiuni între două gazde în scopul trasferului de mesaje pentru o durată limitată de timp. Versiunea curentă a protocolului IP este 4, de aceea se mai foloseşte şi scrierea IPv4. Ca urmare cerințelor de creştere a Internetului, atât în număr de gazde cât şi de trafic, a fost creat noul standard IPv6.

Adresa de IP identifică unic un calculator sau alt dispozitiv (imprimantă) al rețelei TCP/IP. Rețelele ce folosesc protocolul TCP/IP dirijează mesajele pe baza adresei IP a destinatarului. Forma unei adrese IP este un număr de 32 de biți, scrise ca 4 numere întregi separate prin puncte. Fiecare număr poate fi între 0 şi 255, de exemplu 168.212.226.204 poate fi o adresă de IP. Într-o rețea izolată de Internet atribuirea acestor adrese se poate face aleator, condiția fiind aceea ca numerele folosite sunt unice. Dacă această rețea privată va fi însă legată la Internet este obligatorile folosirea unor adrese de IP înregistrate (pentru evitarea dublurilor). Cele patru numere se folosesc pentru a identifica o anumită rețea şi o gazda particulară. 168.212.226.204 se scrie în binar 10101000.11010100.11100010.11001100. Dacă subnet mask-ul este 255.255.255.000, adică 11111111.11111111.11111111.00000000 în binar, primii 24 de biți din IP se folosesc pentru identificarea rețelei, iar ultimii 8 identifică calculatorul gazdă (host-ul) din rețea. Dimensiunea rețelei este funcție de numărul de biți folosiți la definirea gazdelor. Dacă subnet mask-ul are 8 biți pot exista cel mult 256 de adrese de gazde pentru o anumită rețea. Dacă subnet mask-ul are 16 biți (255.255.0.0) atunci sunt 65536 de adrese de gazde disponibile în rețea.

În concluzie, adresa IP este formată din două regiuni, una care identifică rețeaua, alta care identifică gazda (nodul din rețea). Clasa adresei determină care parte a adresei aparține de rețea şi care aparține de gazdă. Toate gazdele unei rețele trebuie să aibă acelaşi prefix de rețea şi trebuie să aibă o valoare de gazdă unică.

Există mai multe organizații (de exemplu, pentru Europa avem Réseaux IP Européens Network Coordination Centre.) care înregistrează şi administrează adresele de IP din următoarele clase:

Clasă	Cei mai semifica-tivi biți ai adresei	Domenii de adrese posibile	Adresa de subnet mask	Rețele pe clasă	Gazde pe clasă	Utilizare
A	0xxx	0.0.0.0 - 127.255.255.255	255.0.0.0	128	16.777.214	Rețele foarte mari, întotdeauna subnetate
B	10xx	128.0.0.0 - 191.255.255.255	255.255.0.0	16,384	65,535	Rețele mari, tipic, subnetate

C	110x	192.0.0.0 - 223.255.255.255	255.255.255.0	2,097,152	254	Rețele mici
D	1110	224.0.0.0 - 239.255.255.255	255.255.255.255	268,435,456	0[2]	Grupuri de adrese de multicast (nu există gazde)
E	1111	240.0.0.0 - 255.255.255.255	nedefinită	nedefinite	nedefinite	Rezervate în scopuri experimentale

Este posibil ca administratorul unei rețele să divizeze o rețea în mai multe rețele virtuale numite subrețele (procesul este numit "subnetare"). Pentru aceasta biții din subnet mask care țin de gazdă se "împrumută" adresei de rețea prin trecerea biților doriți din 0 în 1. De exemplu, fie adresa de rețea 192.168.10.0 cu subnet mask-ul 255.255.255.0. Pentru a face din această rețea două rețele subnet mask-ul se face 255.255.255.128 (adică din 11111111.1111111.11111111.00000000 acesta devine 11111111.11111111.1111111.10000000). Aceasta înseamnă ca în loc de 24 de biți se folosesc 25 de biți pentru rețea și numai 7 pentru host-uri. Rezultă două rețele distincte cu 128 de adrese de host posibile fiecare. Prima rețea va avea domeniul de adrese în 192.168.10.0 - 192.168.10.127, iar cea de-a doua 192.168.10.128 - 192.168.10.255.

Toate rețelele trebuie să rezerve orice adresă de gazdă ce are toți biții 0 respectiv 1 pentru rețea. Astfel fiecare subrețea va avea o adresă specifică de rețea și una de difuzare (broadcast). Într-o rețea cu 256 de gazde, 0 și 255 vor fi adrese rezevate, deci, vor fi numai 254 de adrese de gazde posibile. Pentru exemplul de mai sus, când numai 7 biți se folosesc pentru adresele de gazde, numărul maxim ar fi de 126 din 128 (pentru prima subrețea 192.168.10.0 este adresa de rețea iar 192.168.10.127 adresa de difuzare).

Toate gazdele care dorim **să comunice direct** în rețea trebuie să fie în aceeași rețea, adică să aibă același subnet mask. Dacă se folosesc subnet mask-uri diferite vor crede că sunt în rețele diferite și nu vor putea comunica direct, ci doar indirect, cu ajutorul unui router.

Fără a intra în prea multe detalii, protocolul TCP/IP este cel ce stă la baza Internetului. Acesta asigură trasportul datelor între aplicațiile, rulate pe calculatoarele unor rețele distincte. Ceea ce numim generic Internet, este cea mai mare (în sensul răspândirii geografice) grupare de rețele de calculatoare. Are o structură ierahică cu trei nivele: rețele de tranzit cum sunt ARPAnet, NSFNet, MILNET - asigură traficul de volum foarte mare între rețele pe care le leagă , rețele de mijloc - rețele intermediare, care conectează rețelele mici la cea de tranzit, rețele mici - la nivelul unei astfel de rețele traficul este de tipul local.

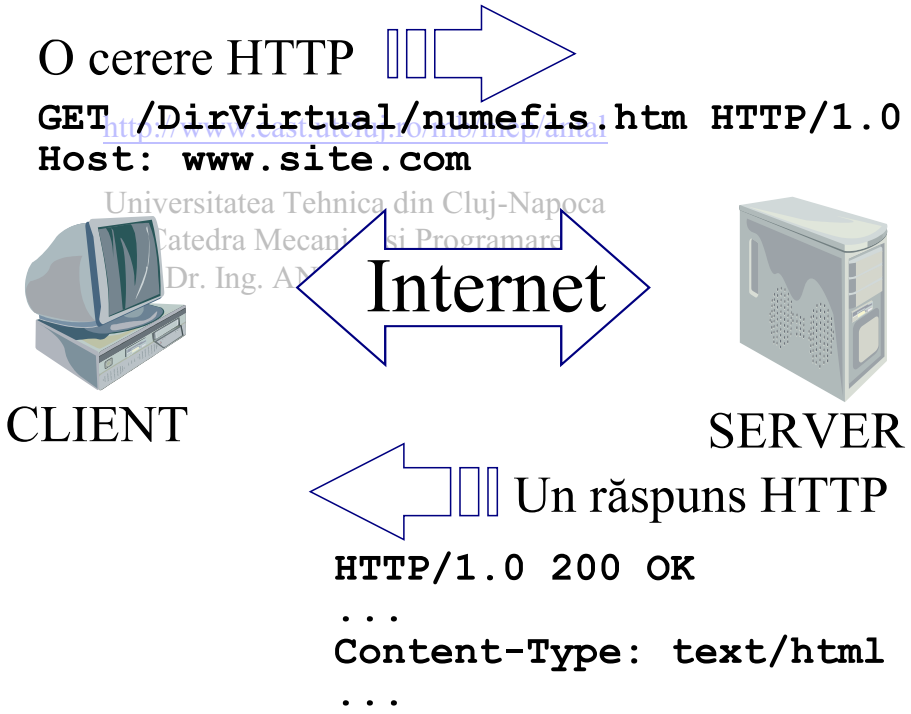
Protocolul HTTP

Protocolul HTTP este ASCII și folosește protocolul TCP/IP pentru a realiza transmisia datelor prin Internet. Definițiile care urmează vă formează cadrul necesar înțelegerii celor ce se vor prezenta în continuare.

- *Protocol* - un set de reguli formale ce descriu modul de transmitere al datelor, deseori, într-o rețea.
- *Port* - un canal logic într-un sistem de comunicație. Protocolul TCP folosește numere de port pentru demultiplexarea mesajelor. Fiecare aplicație are un număr de port unic asociat cu aceasta. Numărul de port al server-elor de Web este 80.
- *Server* - un program care furnizează un serviciu unor alte programe (clienții). Legătura între client și server se realizează prin intermediul unor mesaje transferate, deseori, printr-o rețea și folosește un protocol oarecare pentru codificarea cererilor clienților și a răspunsurilor date de server. Server-ul este rulat continuu așteptând sosirea cererilor sau poate fi pornit ca urmare a unei cereri lansate de un program special.

- *Web server* - un program server rulat pe un site de Web care trimite ca răspuns pagini Web la cererile HTTP ale navigatoarelor. Câteva nume de astfel de programe sunt: Apache, HTTPd, Personal Web Server şi IIS.
- *Site de Web* - un calculator pe Internet pe care se rulează un server de World Wide Web (WWW). Un site particular este identificat printr-un nume de gazdă (`hostname`).
- *WWW (World Wide Web)* - sistem informațional client-server, hipertext, distribuit, inițiat de Laboratorul European de Fizica Particulelor din Geneva, Elveția. Pe WWW totul (documente, meniuri, etc.) se reprezintă prin obiecte hipertext în formatul HTML. Legăturile hipertext referă documente prin URL (standard pentru specificarea locației obiectelor pe Internet). Acestea pot referi o resursă locală sau una aflată la distanță accesibilă prin FTP, Gopher etc. sau via protocolul HTTP folosit pentru transferul documentelor hipertext. Un program client, dintre cele care urmează, se rulează pe calculatorul utilizatorului (deseori, numit şi navigator de Internet) - NCSA Mosaic, Netscape Navigator, Internet Explorer, Mozila FireFox etc. - furnizând două servicii de navigare de bază: urmărirea unei legături şi transmiterea unei cereri server-ului.

Figure 4
Dialog HTTP



HTTP este un protocol de aplicație, în situația concretă, se foloseşte de către clientul de Web şi de către server-ul de Web pentru a comunica. În vederea realizării legăturii între client şi server trebuie să cunoaştem adresa IP a serverului şi portul TCP asociat aplicației server (numărul de port sau, mai pe scurt, portul, asigură identificarea aplicației sau procesului de pe calculator). Unele dintre aceste informații pot fi calculate (adresa de IP se poate determina din numele serverului, care este conținut în URL, cu ajutorul unui server de nume de domenii), altele sunt implicite (am spus deja că numărul de port implicit, pentru server-ele de Web, este 80). O tranzacție între clientul şi serverul de Web este declanşată de client prin emiterea unei comenzi de cerere (request), apoi server-ul trimite clientului un răspuns (response) la comandă (vezi **Figura 4**). De exemplu, comanda `GET` a protocolului HTTP trimite server-ului de Web numele fişierului pe care dorim să-l vizualizăm. Server-ul va răspunde cu fişierul cerut, dar îl va preceda cu un grup de informații care descriu transferul şi formatul informațiilor transferate. Astfel, `HTTP/1.0` este versiunea protocolului HTTP, `200` este codul de stare succes iar `OK` un mesaj ce

explică codul de stare. După `Content-Type`, urmează conținutul fișierul cerut. Cea mai simplă metodă de vizualizare a răspunsului transmis de server către client este utilizarea unei aplicațiie de tip terminal (de ex. `telnet` sau `netcat`). Server-ul de Web monitorizează portul 80 pentru a vedea dacă sunt clienți ce doresc să se lege la el.

În **Figura 5**, se inițiază o conexiune cu server-ul de Web `mare`, ce are adresa `IP 127.0.0.1`. Aplicația netcat (`nc`) este gratuită şi permite conectarea de la distanță la un calculator, folosind TCP/IP. Modul de utilizare este cel de emulare de terminal respectiv acest calculator apare la nivelul celui la care s-a conectat asemenea unui simplu terminal de la care se pot introduce caractere, se poate poziția cursorul, se poate şterge ecranul etc. Pentru a realiza această conectare la server-ul de Web cu netcat se introduce linia **`nc -v -v IP port`**. Utilizarea lui -v asigură afişarea detailată a tuturor informațiilor legate de conexiune. După realizarea conexiunii, tot ceea ce se scrie pe ecranul terminalului va fi trimis exact server-ului. Deoarce ne legăm la un server de Web, trebuie să–i trimitem o comandă a protocolului HTTP. Prin **`GET /poze/t.htm HTTP/1.0`** i se cere server-ului să trimită clientului conținutul paginii cu numele `t.htm` stocată în directorul virtual `poze`. Tasta **<Enter>** trebui apăsată de două ori, după care acesta va trimite răspunsul lui clientului. Deoarece clientul nostru nu este un navigator, ci doar un emulator de terminal (nu ştie interpreta limbajul HTML în care s-a scris pagina), pe ecran, răspunsul apare afişat în ASCII.

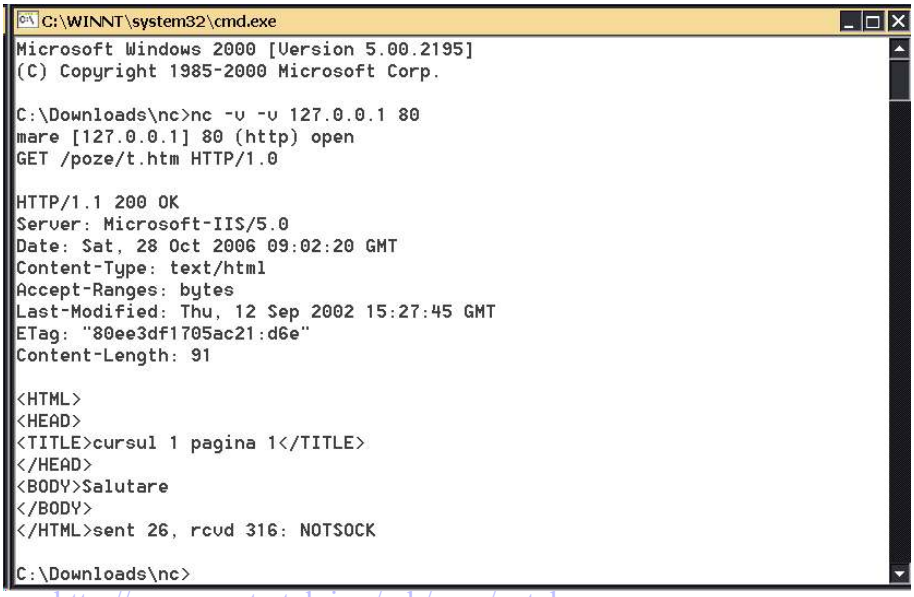


Figure 5 - Utilizarea netcat-ului pentru descărcarea unei pagini de Web

Se observă că răspunsul are trei părți distincte:

- prima linie care ne spune starea răspunsului (HTTP/1.1 200 OK);
- antetul (header): o regiune formată din perechi nume: valoare terminată cu o linie vidă;
- corpul (body): conținului scris în limbajul HTML al paginii cerute.

Această descriere a dialogului între sever-ul de Web şi client a fost prezentat prin prisma protocolului HTTP, deoarece învățarea se face mai comod dacă cititorul "vede" măcar o parte a descrierii. Etapele acestui dialog vor fi însă reluate, într-un context mai general şi mai puțin tehnic, pentru asigurarea percepției golbale asupra procesului.

Ce este ASP?

Active Server Pages (ASP) este o combinație de tehnologii ce permit crearea unor documente care conțin porțiuni de HTML şi de program. Utilizarea ASP obligă la învățarea unor colecții de instrumente de dezvoltare, limbaje de programare, tehnici şi tehnologii de programare. ASP, de asemenea, necesită folosirea unui Web server care prelucrează cererile clienților şi formulează răspunsurile în mod dinamic, individualizate pe baza unui proces logic, a unui fişier, a unor date dintr-o bază de date şi prelucrează datele individuale ale clientului utilizator. ASP permite tratarea fiecărui utilizator ca o entitate unică, deşi toți utilizatorii rulează acelaşi program pe un singur calculator care este Web server-ul.

De la HTML la ASP

HyperText Markup Language (HTML) este un format de fişiere text folosit pentru definirea modului de afişare a conținutului unei pagini de Web, de obicei într-un navigator. Acesta este motivul pentru care HTML a devenit atât de popular. Până la apariția HTML singura modalitate prin care un novice în calculatoare putea să afişeze text şi grafică pe ecran erau editoarele de texte care puteau lucra în modul WYSIWYG (What-You-See-Is-What-You-Get). Însă şi aici erau probleme. Pentru păstrarea compatibilității între documente toți utilizatorii trebuiau să folosească acelaşi sistem de operare şi acelaşi editor de texte. HTML a schimbat acest calvar, prin simplitatea lui oferind în plus posibilitatea de navigare între fişiere fără a şti locul în care se află acestea. Acest mod de navigare se numeşte hipertext, iar conceptul a fost inventat de Ted Nelson cu 30 de ani înainte de apariția HTML. Datorită simplității HTML, în scurt timp de la scoaterea lui pe piață, acesta a devenit limbajul folosit pentru afişarea informațiilor pe WWW. După ce milioane de pagini au fost publicate pe Internet, unii dintre cititori au început să se plângă de faptul că informațiile nu erau suficient de specifice. Ei doreau o posibilitate de personalizare a paginilor, interactivitate, posibilitatea de memorare a paginilor sau, mai pe scurt, doreau aplicații şi nu doar simple pagini de informație.

CGI

HTML, fiind un limbaj foarte simplu, nu putea îndeplini aceste condiții fără a avea în spate un motor de aplicații.

Terminologie

- *CGI (Common Gateway Interface)* - un standard pentru rularea unor programe externe pe un server WWW HTTP. CGI specifică modalitatea de transfer pentru argument către programul în execuție ca o parte a unei cereri HTML. De asemenea, defineşte o mulțime de variabile de mediu. Deseori, programul va avea ca rezultat text HTML care va fi retrimis către navigatorul care a transmis cererea, dar este posibilă şi redirectarea către un alt URL. CGI permite ca textul HTML întors să depindă într-un mod arbitrar de cerere. Programele CGI acceptă argumentele prin linia de comandă. Perl este un limbaj de script des folosit la scrierea programelor CGI. Când server-ul primeşte o cerere de execuție CGI creează un nou proces pentru execuția programului extern. Dacă dintr-un oarecare motiv programul "crapă" sau cererile către server vin într-un număr mult prea mare, acesta nu va putea retrimite răspunsurile în timp util şi se va "împotmoli" în procese, performanțele lui scăzând mult.
- *URL (Uniform Resource Locator)* - un standard pentru specificarea unui obiect pe Internet. Se utilizează în documentele HTML pentru specificarea sursei unei hiperlegături. Iată câteva exemple:

```
ftp://wuarchive.wustl.edu/mirrors/msdos/graphics/gifkit.zip
ftp://spy:secret@ftp.acme.com/pub/topsecret/weapon.tgz
```

```
http://www.w3.org/default.html
news:alt.hypertext
telnet://dra.com
mailto:dbh@doc.ic.ac.uk
http://wombat.doc.ic.ac.uk/?Uniform+Resource+Locator
http://www.w3.org/default.html#Introduction
```

Partea din fața caracterului `:` specifică schema de accesare sau protocolul. Partea de după `:` se interpretează în funcție de schema de acces. În general, cele două caractere `//` descriu un nume de host (host - un nume care identifică unic un calculator pe Internet; alte forme folosite mai sunt: `host:port` sau pentru FTP utilizator: `parolă@host` sau `utilizator@host`). Schemele includ ftp, http, gopher sau WAIS. Schema fișierului poate fi folosită numai pentru referirea unui fișier aflat pe host. Scheme mai puțin folosite sunt news, telnet sau mailto (e-mail). Numarul de port poate fi omis din URL, caz în care acesta ia valoarea implicită de 80. Ultima parte, opțională, din URL poate fi un șir de caractere precedat de `?` sau un fragment de identificare precedat de `#` pentru specificarea unei poziții particulare în document.

- *hiperlegătură* - o referință dintr-un punct al unui document hipertext către un alt document sau o altă poziție din același document. Navigatoarele afișează hiperlegăturile dintr-un document într-o formă și culoare distinctă de textul obișnuit al acestuia. Atunci când utilizatorul activează o hiperlegătură (printr-un clic de mouse pe aceasta) navigatorul va afișa destinația stocată în hiperlegătură.
- *limbaj de script (script language)* - un limbaj de programare relativ mic, deseori interpretat (nu compilat).

HTML a fost ajutat de CGI să poată răspunde cerințelor tot mai dure formulate de utilizatori. O aplicație CGI se rulează pe server. Când navigatorul contactează server-ul și formulează cererea, aplicația CGI va întoarce text HTML navigatorului. La nivelul aplicației CGI se pot realiza prelucrări ale informațiilor trimise prin cererea navigatorului și se poate genera un răspuns care diferă în funcție de anumite condiții. Deși aplicațiile CGI pot fi scrise în aproape orice limbaj de programare, majoritatea programatorilor au folosit limbajul Perl (Practical Extraction and Report Language, deseori numit în lumea bună a programatorilor Pathologically Eclectic Rubbish Lister) datorită posibilităților avansate a lui de prelucrare a textelor. Programele CGI folosesc formulare pentru a interacționa cu utilizatorul în vederea realizării unor căutări și pentru a întoarce răspunsuri personalizate. Din păcate CGI avea câteva probleme. Primele versiuni de CGI trebuiau să încarce în memorie aplicația CGI pentru fiecare cerere. În cazul unor site-uri cu trafic mai mare aceasta însemna aproximativ 100 de cereri pe secundă. Încărcarea și descărcarea aplicațiilor lua însă mult din timpul de lucru al Web server-ului. Acesta este motivul pentru care firmele serioase de software de Web au căutat alte soluții. Noile concepte au fost standardizate sub numele de Internet Server Application Programming Interface (ISAPI) sau, în cazul server-elor Netscape, Netscape Server Application Programming Interface (NSAPI). În acest caz aplicațiile CGI puteau fi rulate ca o parte a server-ului de Web (nu mai erau aplicații externe lui) fără a mai trece prin încărcarea și descărcarea lor cu fiecare cerere.

Avantaje ASP

Câteva dintre avantajele principiale ale ASP sunt prezentate în continuare.

Independenţa de limbajul de programare

Motorul ASP nu depinde de un anumit limbaj de programare. Motorul ASP nu rulează codul scris de programator. El este o gazdă independentă de limbajul de script folosit. În acest fel, el poate folosi orice limbaj care este compatibil cu cerințele descrise în Microsoft Scripting Host. Dacă este cazul, cod în mai multe limbaje de programare poate fi scris pe aceeași pagină. Motorul ASP

va separa codul script de HTML, apoi va cere motorului corespunzător să-l ruleze. Majoritatea codului care urmează va fi scris în limbajul **VBScript**, cu excepţia unei părţi scurte care va fi scrisă în **JavaScript** (în secvenţele de cod următoare caracterul ✄ are semnifică o linie întreruptă din motive de spaţiu).

ASP este simplu de învăţat

Una dintre problemele calculatorişţilor este găsirea unui limbaj de programare pe care ne-programatorii să-l poată înţelege în vederea implementării unor operaţii tipice anumitor munci. Ideea reutilizării codului scris de cineva în altă parte permite crearea unor aplicaţii funcţionale chiar şi de către neprofesionişţii în programare. ASP face legătura între mai multe tehnologii. Astfel, componente software scrise de programatori profesionişţi pot fi integrate în orice aplicaţie fără mult efort de către programatorii ne-profesionişţi.

Alte metode pentru crearea de pagini Web dinamic

ASP este cea mai nouă modalitate de creare a paginilor Web dinamice, dar nu este singura modalitate de lucru. Prin anii '90 cea mai utilizată tehnologie Web era programarea CGI.

Programe CGI

Programele CGI sunt aplicaţii executabile pe care server-ul de Web le rulează atunci când navigatorul lansează o astfel de cerere. Programul CGI prelucrează cererea prin codul scris şi întoarce text HTML navigatorului. Diferenţa dintre un program CGI şi o pagină de Web statică este aceea că textul HTML întors navigatorului poate fi diferit în funcţie de utilizatorul care face cererea, a locului de unde o face şi a momentului ales. Microsoft a observat că programele CGI nu sunt cel mai eficient mod de prelucrare a cererilor pentru generarea unui conţinut activ. Motivul ineficienţei este necesitatea încărcării şi descărcării programului CGI la fiecare cerere. Explicaţia folosirii acestui mod de lucru constă în protocolul de transfer hipertext (HTTP - HyperText Transfer Protocol). O cerere HTTP este o tranzacţie scurtă între un client şi un server. Atât server-ul cât şi clientul vor uita că tranzacţia a avut loc după derularea ei. La fiecare tranzacţie HTTP clientul şi server-ul, pe baza unui protocol fix, vor derula aceeaşi comunicaţie. Apoi, întrucât server-ul uită cine a fost clientul va descărca din memorie programul CGI încărcat ca urmare a cererii navigatorului. Dacă apare o nouă cerere programul CGI trebuie reîncărcat, iar această reîncărcare este deseori mai înceată decât execuţia programului în sine. Din punctul de vedere al server-ului de Web activitatea de încărcare a programelor CGI este neproductivă. Soluţia la această problemă s-a numit ISAPI.

Ce este ISAPI?

Pentru eliminarea acestei probleme legate de aplicaţiile CGI, Microsoft a introdus o nouă interfaţă numită ISAPI (Internet Server Application Programming Interface). Aici, timpul necesar pentru încărcarea programului CGI se păstrează, dar acesta nu mai este descărcat din memorie la terminarea rezolvării cererii. El va rămâne încărcat în memorie atât timp cât server-ul funcţionează sau pe o durată de timp predefinită. Există două tipuri de programe ISAPI: aplicaţii ISAPI şi filtre ISAPI.

Aplicaţii ISAPI

Aplicaţiile ISAPI se rulează după ce server-ul de Web a primit cererea. Aplicaţiile ISAPI permit adăugarea de noi funcţii server-ului de Web cu păstrarea facilităţilor deja existente ale server-ului. Motorul ASP este o aplicaţie ISAPI. IIS (Internet Information Server) transferă cererile de fişiere ASP motorului ASP care prelucrează cererea pentru a întoarce conţinutul dinamic.

Filtre ISAPI

Filtrele ASP sunt aplicaţii care se rulează înainte ca server-ul de Web să primească o cerere în

acest scop. Acestea, deseori, permit monitorizarea sau interceptarea unor cereri de prelucrări speciale.

Când se folosesc ASP şi HTML împreună?

HTML este un limbaj simplu şi flexibil pentru organizare şi formatare, dar nu are integrat un limbaj de programare. Dacă se doreşte afişarea de texte statice, HTML este perfect. Dacă însă conținutul de afişat se schimbă des sau acesta trebuie personalizat pentru grupe de utilizatori HTML nu poate face față întrucât la nivelul lui nu se pot lua decizii. Pentru acest scop este nevoie de un limbaj de programare. Scripturile ASP oferă mecanisme care permit lucrul cu baze de date, luarea unor decizii, accesul la resursele hard ale calculatorului server etc. pentru generarea de pagini HTML cu un conținut actualizat în mod dinamic.

Comparaţie între ASP şi alte tehnologii de dezvoltare a aplicaţiilor Web

Terminologie

- *componentă ActiveX* - o unitate de cod executabil (.exe, .dll sau .ocx) care respectă specificațiile ActiveX pentru crearea obiectelor. Tehnologia ActiveX permite ca programatorii să asambleze componente software reutilizabile într-o aplicație sau într-un serviciu software.
- *COM (Component Object Model*) - tehnologie inventată de Microsoft pentru dezvoltarea aplicațiilor prin componente software. COM defineşte principiile pentru implementarea cu succes a încapsulării, polimorfismului şi a moştenirii împreună cu o structură pentru distribuirea acestor componente
- *DLL (Dynamic Link Library)* - o grupare de rutine dintr-o bibliotecă ce pot fi apelate din proceduri şi care se încarcă şi se leagă în aplicație, în momentul execuției acesteia.

ASP are câteva avantaje majore în comparație cu celelalte tehnologii şi medii folosite pentru dezvoltarea aplicațiilor de Web:

1. **codul ASP este stocat în fişiere text** - fişierele text sunt uşor de modificat inclusiv după depunerea pe server. Au uriaşul avantaj de a putea rezolva orice problemă folosind un editor de texte simplu. Aplicațiile Web ce depind de cod compilat sau de obiecte ActiveX sunt mult mai greu de întreținut şi de actualizat.

2. **codul ASP expiră în timp** - IIS (Web server-ul firmei Microsoft) opreşte execuția codului ASP automat după 90 de secunde (durata de timp se poate modifica). Dacă dintr-o eroare scriem un cod care are o buclă infinită sau dacă cineva cere milioane de înregistrări, server-ul nu va fi blocat decât în perioada celor 90 de secunde. Mulți furnizori de internet nu folosesc aplicații compilate pentru motivul că cele ASP expiră după un anumit timp.

3. **codul ASP nu blochează server-ul** - codul ASP are limitări dure, de exemplu nu veți putea citi sau scrie fişirere binare direct cu ASP. Este foarte dificil ca sever-ul IIS să fie blocat, acesta este încă unul dintre motivele pentru care mulți furnizori de servicii internet preferă să lucreze cu tehnologia ASP.

4. **codul ASP nu necesită înregistrare** - Programul de instalare al IIS va instala DLL-urile motorului ASP, cele de script, Microsoft ActiveX Data Objects DLL (ADO) şi Microsoft Scripting Runtime DLL. Acestea sunt toate componentele necesare rulării unei aplicații ASP. Celelalte instrumente de dezvoltare necesită instalări de componente adiționale pe server şi operații cu registrul de Windows.

5. **aplicațiile ASP sunt mici** - întrucât majoritatea DLL-urilor sunt deja instalate pe server este necesară doar copierea fişierelor cu cod, a imaginilor şi a fişierelor de suport pentru ca aplicația ASP să devină funcțională. Pe măsură ce aplicației i se adaugă tot mai multe componente ActiveX compilate ne îndepărtăm tot mai mult de o aplicație mică.

6. **aplicațiile ASP pot fi actualizate fără oprirea IIS** - la prima vedere nu pare să fie un avantaj mare atunci când se rulează o singură aplicație pe server. Atunci însă când se rulează zeci de aplicații nimeni nu doreşte oprirea server-ului numai pentru că una dintre ele va fi actualizată.

Etapele desfăşurării dialogului, între client şi server, pe Web

Cu fiecare introducere a unui URL nou în cutia de adrese a navigatorului, selectarea unei legături, sau a transferului unui formular, navigatorul va împacheta informații despre el însuşi, URL şi uneori despre utilizator şi le va trimite server-ului de Web sub forma unei cereri. Motorul ASP este o colecție de obiecte care conțin informații despre cerere, tehnologia de script folosită pentru luarea deciziilor, despre modul de tratare a cererii cât şi despre server-ul de Web. Înainte de a detalia modul de lucru a motorului ASP să vedem anatomia unei cereri.

Orice cerere pe Web necesită două componente, clientul şi server-ul de Web. Clientul este, deseori, un navigator dar poate fi şi un spider (program care parcurge legăturile de pe Web pentru a culege informații) sau un agent (un program însărcinat cu găsirea unor informații specifice care foloseşte de regulă motoare de căutare). De obicei, server-ul şi clientul sunt pe două calculatoare diferite, dar se poate ca acestea să fie rulate şi pe acelaşi calculator. Atât server-ul cât şi clientul trebuie să folosească acelaşi protocol pentru a comunica. Protocolul este o convenție cu privire la desfăşurarea unei sesiuni de comunicație (inițializare, transfer de informație şi terminare). Există mai multe protocoale folosite la transferul de informație pe Web; cele mai comune sunt HTTP şi FTP (File Transfer Protocol). Indiferent de protocolul folosit pentru cererile pe Web acestea se rezolvă printr-un serviciu de transfer de bază numit, TCP/IP (Transmission Control Protocol/Internet Protocol). Acesta este standardul de comunicație globală pentru două calculatoare ce schimbă informație prin Internet. Server-ul rulează o buclă infinită în care verifică inițializarea comunicației. Clientul, trimite o secvență de inițializare pentru a marca începutul unei sesiuni. Când server-ul primeşte cererea de inițializare validează transmisia printr-un răspuns către client. Apoi, conversația între client şi server continuă în ambele sensuri până când răspunsul dat la cerere se termină.

Modul în care clientul trimite cererea

Clientul trimite toate cererile unui server de nume (naming server). Acesta are o bază de date cu nume ce sunt asociate unor adrese de IP. Acest server traduce numele adresei de accesat în cifre. O adresă de IP este formată din patru numere în domeniul 0-255 separate prin caracterele punct. De exemplu: 207.225.123.37.

Fiecare adresă de IP identifică în mod unic un calculator dintr-o rețea. Dacă primul server de numer nu cunoaşte adresa dorită atunci o va transmite următorului dintr-o ierarhie prestabilită. Dacă nici un server de nume nu poate traduce numele adresei într-o adresă de IP cererea va ajunge la un server de nume mai puternic care stochează lista tuturor adreselor publice de IP. Dacă nici aici traducerea nu se poate realiza, atunci răspunsul negativ va fi transmis pe baza ierarhiei server-elor de nume până când ajunge înapoi la navigator. Acesta este momentul în care navigatorul va afişa un mesaj de eroare. În caz de succes, server-ul de nume întoarce adresa de IP navigatorului, iar acesta va folosi adresa de IP pentru a contacta server-ul de Web asociat numelui respectiv. Multe pagini conțin referințe la alte pagini sau imagini pe care server-ul de Web trebuie să le ceară din altă parte pentru ca pagina să poată fi completă. Astfel, procesul de afişare a unei pagini Web se face prin mai multe conversații scurte între navigator şi server. Tipic, navigatorul primeşte pagina principală, apoi caută referințele către alte fişiere, continuă cu afişarea paginii principale în timp ce îşi formulează cererile cu privire la referințele către fişierele necunoscute.

Prelucrarea cererii de către server

Din punctul de vedere al server-ului de Web fiecare conversație este un contact nou. Server-ele de Web nu rețin un anumit navigator între două cereri ale acestuia (protocolul HTTP 1.1 are deja

posibilitatea menținerii unei conexiuni cu cereri multiple).

Părţile componente ale unui URL

Linia pe care o introducem ca adresă în navigator este un URL. Server-ul descompune acest şir în mai multe componente care sunt separate prin două puncte, slash, punct etc. Fie URL-ul:

```
http://www.unsite.ro/Programe/default.htm?Pagina=1&Paragraf=2
```

Elementele componente ale URL-ului de mai sus se prezintă în tabelul următor:

Compomentă URL	Funcție	Descriere
`http`	Protocol	Spune server-ului protocolul care îl va folosi pentru a răspunde la cerere.
`www.unsite.ro`	Numele domeniului	Această parte din URL se traduce într-o adresă de IP. Numele de domeniu este format din mai multe părți separate prin caracterele punct.
`Programe`	Director virtual	Server-ul transformă acest nume într-o cale fizică pe hard disc.
`default.htm`	Nume fişier	Server-ul va întoarce conținutul acestui fişier. Dacă fişierul este unul executabil (de exemplu fişier ASP) server-ul îl va executa în loc să întoarcă conținutul fişierului.
`? (semul de întrebare)`	Separator	Acest caracter separă cererea de fişier de parametrii adiționali trimişi împreună cu cererea. În exemplul prezentat URL-ul are doi parametri `Pagina=1&Paragraf=2`.
`Pagina`	Nume parametru	Programele scrise, cum sunt şi paginile ASP, pot citi parametrii şi-i pot folosi pentru a obține informații adiționale.
`= (semnul egal)`	Separator	Separă numele parametrului de valoarea dată acestuia.
`1`	Valoare parametru	Parametrul cu numele `Pagina` ia valoarea `1`. Toate valorile de parametri se transferă ca şiruri de caractere. Programele scrise pot interpreta aceste valori ca numere numai în urma conversiei lor de la şir la număr.
`&`	Separator	Separă perechile `parametru=valoare` între ele.
`Paragraf=2`		Al doilea parametru cu valoarea lui.

Pe Web, referirea la fişiere nu se face printr-o cale fizică, ci prin una virtuală. După transferarea

URL-ului, server-ul transformă calea virtuală în una fizică. De exemplu, calea `numeCale` din URL-ul `http://numeSever/numeCale/fisier.asp`, este una virtuală. Directorul virtual `numeCale` va fi transformat într-o cale către un director local de exemplu de forma `c:\inetpub\wwwroot\asp\fisier.asp`.

Server-ul verifică existența fişierului cerut. În cazul în care acesta nu există el va întoarce mesajul de eroare `HTTP 404 - File not found`.

După localizarea resursei cerute server-ul verifică dacă contul care cere informațiile are drepturile de accesare a resursei. Dacă de exemplu, contul de pe care s-a făcut cererea este unul anonim şi utilizatorul a cerut un fişier pentru care acel cont nu are drepturi de citire, server-ul va întoarce mesajul de eroare `HTTP 403 - Access Denied`. Mesajele de eroare 403 sunt organizate pe subnivele. Se poate, deci ca textul mesajului să fie diferit de cel de sus în funcție de motivul exact care a condus la imposibilitatea deservirii cererii (conținutul la majoritatea mesajelor de eroare poate fi personalizat).

Modul în care server-ul răspunde la o cerere

Server-ele diferențiază tipurile de fişiere cerute prin mai multe metode. IIS face această diferențiere pe baza extensiei fişierului (de exemplu: asp, htm, exe etc.) la fel cu Windows Explorer. Atunci când se face clic dublu pe un fişier, Windows Explorer caută extensia fişierului în registru (o bază de date specială în care se stochează informații legate de sistem şi de aplicații). Registrul conține câte o intrare pentru fiecare extensie de fişier înregistrată. Fiecare extensie are asociată o intrare de tip de fişier. Fiecare tip de fişier are asociat un fişier executabil sau o modalitate prin care se prelucrează. Server-ul extrage extensia de fişier din numele lui apoi caută programul asociat acesteia şi lansează în execuție acel program pentru a întoarce fişierul.

Majoritatea server-elor de Web folosesc extensia fişierelor pentru a determina modul de prelucrare a cererii, dar nu folosesc asociații din registru ci au o listă proprie de perechi *(extensie, aplicație)*, în acest scop. Intrările în aceste liste poartă denumirea de tipuri MIME (Multipurpose Internet Mail Extensions) întrucât programele de e-mail trebuie să cunoască tipul conținutului fişierului inclus în mesaj. Fiecare tip MIME, asemenea asocierilor din registru, este asociat unui program sau unei acțiuni. Server-ul de Web caută lista pentru a găsi extensia corespunzătoare fişierului cerut. Majoritatea server-elor care nu găsesc extensia cerută tratează problema permițând descărcarea fişierului în cauză pe client. Unele server-e au predefinite acțiuni în cazul în care se cere un URL care nu are nume de fişier. Deseori, un fişier implicit, cu numele `default.htm` sau `index.htm`, va fi întors. Numele fişierelor implicite de întors este configurabil, la nivel de server sau, la nivel de director virtual.

Server-ul poate trimite informația cerută pe măsură ce generează răspunsul sau îl poate stoca într-un buffer pentru a-l trimite tot, într-o singură bucată, atunci când acesta este formulat complet. Răspunsul este format din două părți: antetul (header) şi corpul (body). Antetul conține informații legate de tipul răspunsului. Aici apar: codul răspunsului, tipul de MIME, data şi ora după care răpunsul devine invalid, un URL pentru redirectare, orice valori de `cookie` (un şir de caractere pe care navigatorul îl salvează pe discul clientului) pe care server-ul doreşte să le stocheze. Un cookie poate exista pe durata unei sesiuni de navigare, până la o anumită dată de expirare sau poate fi permanent. Navigatorul trimite server-ului un cookie asociat unui site pentru fiecare cerere ulterioară adresată pentru acel site.

Modul în care clientul prelucrează răspunsul

Clientul, deseori un navigator, trebuie să cunoască tipul conținutului transmis ca răspuns de către server. El citeşte tipul de MIME din antet pentru a determina tipul conținutului. La majoritatea

cererilor tipul de MIME este `text/html` sau `image/gif`, dar ar putea fi şi fişier editor de texte, fişier video, fişier audio, o animaţie sau orice alt tip de fişier. Asemenea server-ului, navigatorul foloseşte şi el registru şi liste de tipuri de MIME pentru a determina modul de afişare a fişierului. Pentru HTML şi imagini navigatorul va folosi motorul lui intern. Pentru alte tipuri de fişiere el va apela serviciile de interfaţare cu aplicaţiile sau plug-ins-urile care pot afişa informaţiile corect. Navigatorul va asigna toată fereastra lui sau o parte din ea aplicaţiei sau plug-ins-ului pentru ca aceasta să-i genereze conţinutul. Atunci când corpul fişierului conţine HTML navigatorul va analiza fişierul pentru a extrage marcajele din conţinut. Apoi, marcajele vor fi folosite pentru a determina modul de afişare a conţinutului pe ecran. Noile fişierele HTML pot include mai multe tipuri de conţinuturi în plus faţa de marcaje, text şi imagini. Navigatorul le va trata pe fiecare diferit, în funcţie de tip, după cum urmează:

Cascading Style Sheets (CSS) - CSS conţin informaţii despre modul de formatare a documentului. Navigatoarele moderne folosesc CSS pentru specificarea culorilor, marginilor, vizibilităţii, poziţiei etc. elementelor de pe pagină.

Script - toate navigatoarele moderne pot rula cod JavaScript, deşi, deseori, nu-l rulează în acelaşi mod. Termenul de **JavaScript** este folosit pentru secvenţele de cod scrise în limbajul de script **JavaScript** a lui Nestcape, deşi el are două variante în esenţă identice (aceeaşi sintaxă şi suport), **JScript** a lui Microsoft şi **ECMAScript**. În plus, faţă de **JScript**, Internet Explorer permite programarea şi în **VBScript** care este un subset a lui **Visual Basic for Application** (un subset a limbajului **Visual Basic**).

Componente ActiveX sau *Java Applets* - acestea sunt programe mici care se rulează pe calculatorul clientului şi nu pe server. Componentele ActiveX rulează numai pe Internet Explorer şi sub Windows, în timp ce Java Applets rulează aproape pe toate navigatoarele şi platformele.

XML (eXtensible Markup Language) - este un limbaj asemănător cu HTML, ambele au marcaje şi conţinut, fiind derivate din SMGL (Standard Generalized Markup Language). Marcajele HTML descriu modul de afişare a conţinutului şi într-o măsura mai limitată, funcţiile conţinutului. Marcajele XML descriu funcţiile conţinutului. HTML este un limbaj de formatare şi afişare în timp ce XML este un limbaj pentru descrierea conţinuturilor.

Prelucrarea cererilor ASP

O cerere de fişier ASP parcurge aceleaşi etape cu un fişier HTML obişnuit până când ajunge la server. Mai departe însă server-ul va dirija cererea către motorul ASP în locul motorului implicit IIS. Motorul ASP va citi fişierul cerut fie de pe disc, fie din cache-ul server-ului de IIS după care va analiza fişierul. Trei dintre etapele prelucrării codului ASP de către server sunt importante din punctul de vedere al programatorului ASP:

- motorul ASP va insera toate fişierele `include`. Aceste fişiere sunt separate de cel cerut iniţial şi IIS poate insera conţinutul lor în fişierul cerut. După inserare, IIS prelucrează fişierul ca şi când fişierele inserate ar fi parte a celui cerut. Inserarea fişierelor cu `include` se face înainte ca motorul ASP să prelucreze codul.

- motorul ASP începe să interpreteze codul. Codul este prelucrat secvenţial în ordinea scrierii instrucţiunilor din fişier, cu excepţia secţiunilor marcate cu `Function` sau `Sub`.

- motorul ASP întoarce răspunsul. Se poate controla dacă răspunsul întors de server

este imediat (răspuns nebuffer-at) sau dacă este stocat într-un şir până când el este complet (răspuns buffer-at) prin setările IIS.

Navigatorul nu ştie nimic despre tehnologia folosită de server pentru a răspunde cererii. Din punctul lui de vedere toate răspunsurile sunt şiruri de caractere sau numere. Tipul de MIME din răspuns determină modul în care navigatorul va trata şirul.

Ce este un Script?

Terminologie

- *Limbaj maşină* - reprezentarea unui program care poate fi citită şi interpretată de un calculator.
- *Limbaj de programare* - un limbaj formal folosit pentru programarea calculatoarelor. Definiția unui limbaj particular se face prin sintaxă (modul în care simbolurile limbajului pot fi combinate) şi semantică (semnificația construcțiilor din limbaj).
- *Compilator* - un program care converteşte un program dintr-un limbaj sursă (sau limbaj de programare) în limbaj maşină (cod obiect).
- *Interpretor* - un program care rulează (execută) un alt program. Compilatorul nu rulează programul de intrare (programul sursă) ci îl traduce în cod executabil al maşinii. Acest cod constituie ieşirea compilatorului (rezultatul compilării), el fiind stocat într-un fişier de unde poate fi ulterior rulat. Acelaşi program sursă poate fi rulat direct de un interpretor sau poate fi compilat, după care codul rezultat va fi rulat. Un program rulat pe un interpretor este mai încet decât unul compilat, dar se poate ca timpul de interpretare să fie mai scurt decât cel de compilare. Interpretarea codului este mai înceată decât rularea codului compilat deoarece interpretorul trebuie să analizeze fiecare linie de program la fiecare reluare a rulării şi trebuie să regenereze acțiunile dorite, în timp ce codul compilat, face acțiunea imediat.

În cele discutate până acum am folosit de mai multe ori termenul de script. Din păcate nu există un răspuns foarte clar pentru acest termen el depinzând de documentația consultată. Pentru Microsoft, un script este orice limbaj de programare ActiveX care expune o interfață compatibilă cu **Windows Scripting Host**. Două dintre cele mai comune limbaje de script ale lui Microsoft sunt **VBScript** şi **JScript** sau **JavaScript** în cazul lui Netscape. Pentru Sun Microsystems, script înseamnă **JavaScript** rulat pe server ca şi pagină **JavaScript** (JSP - JavaScript Pages), această tehnologie fiind cel mai serios concurent al ASP-ului. Este general acceptat, că sub denumirea de script se înțelege un limbaj de programare restrâns şi interpretat. Limbajele de script, în comparație cu limbajele obişnuite, au un set restrâns de instrucțiuni şi de tipuri date. În acest sens **VBScript** este un subset al limbajului **Visual Basic**, însă **JavaScript** nu este un subset al limbajului Java, deşi cele două limbaje seamănă mult la nivel sintactic. **JavaScript** a fost inventat de Netscape întrucât aveau nevoie de un limbaj al cărui cod să fie rulat în siguranță la nivelul navigatorului. **JScript** şi **JavaScript** nu sunt identice, dar probabil că în timp vor evolua către un singur limbaj ca urmare a standardelor ECMA.

Majoritatea navigatoarelor pot rula **JavaScript**. IE este singurul navigator care ştie rula **VBScript**. În cele ce urmează limbajul folosit pentru cod este **VBScript** deoarece codul ASP se rulează pe server şi nu pe navigator. Limbajul folosit pe server nu afectează navigatorul în nici un fel. **VBScript** a fost introdus de Microsoft ca fiind limbajul implicit pentru pagini ASP, din acest motiv foarte multe exemple de programe din documentație sunt scrise în **VBScript**.

Sigur că aceste limbaje nu sunt singurele variante, practic se poate folosi orice limbaj de script compatibil ActiveX (câteva dintre variante ar fi Java, C sau Perl).

Mai trebuie reținut că programele ASP pot fi rulate atât pe server, înainte ca răspunsul să fie complet, cât şi pe client (navigator), după ce răspunsul a început să ajungă înapoi la navigator. Deoarece majoritatea clienților sunt încă produse Netscape este bine ca la crearea unor aplicații publice de Internet să se folosească **JavaScript** pentru întreg codul care se rulează pe client.

Modul în care server-ul separă script-ul de conţinutul HTML

Când motorul ASP parcurge fişierul, el separă script-ul de pe server de conținut în două moduri. Script-ul ASP este separat de conținut prin folosirea delimitatorilor. Unul se foloseşte pentru marcarea începutului de cod, "`<%`", altul pentru marcarea terminării codului, "`%>`". Există şi o metodă mai tradițională, în stil HTML pentru scrierea codului.

```
<%                      <script language="JScript" RunAt="Server">
'cod ASP                'cod ASP
%>                      </script>
```

În cea de a două variantă textul `RunAt="Server"` poartă denumirea de atribut şi spune server-ului să ruleze codul. În lipsa lui, server-ul va ignora codul, în schimb navigatorul va fi cel care va încerca să-l ruleze.

Modul în care server-ul prelucrează script-ul

Când server-ul apelează motorul de script ASP acesta prima oară parcurge fişierul şi inserează toate fişierele `include`. Apoi, va colecta toate porțiunile de script pe baza delimitatorilor. Motorul ASP tratează tot conținutul ne-script ca un şir pe care îl trimite fără modificări clientului. În continuare va crea variabilele şi componentele externe script-ului. În final va interpreta script-ul executând fiecare comandă şi trimițând răspunsul rezultat clientului (răspunsul, în funcție de setări, poate fi şi buffer-at). După terminarea prelucrării script-ului, motorul va distruge variabilele şi componentele externe create pentru rularea script-ului. Dacă rezultatul a fost buffer-at, server-ul întoarce clientului conținutul buffer-ului.

Navigatorul şi codul ASP

Navigatorul nu face diferența între pagini ASP sau HTML şi nici nu cunoaşte modul în care server-ul prelucrează cererile. Tot ceea ce contează din punctul lui de vedere este ca transmisia cu server-ul să se desfăşoare cu respectarea unuia dintre protocoalele de comunicație.

Introducere în HTML

Acest capitol îşi propune să expună bazele HTML parcurgând structurile elementare şi metodele de editare.

Ce este HTML?

HTML este un limbaj pentru specificarea informaţiilor de afişare şi de aranjare a conţinutului unui document. Aceste informaţii, numite marcaje, sunt comenzi pe care le au majoritatea editoarelor de text. De exemplu, prin HTML se poate spune calculatorului că documentul conţine paragrafe, liste, tabele sau imagini. Motorul de afişare HTML este responsabil de aşezarea pe ecran a informaţiilor de afişat. Diferenţa între un editor de texte şi HTML este că editorul de text este proprietarul formatului de afişare. Din acest motiv, când se încearcă transferul fişierelor între două editoare de texte diferite, este necesară conversia de format prin folosirea funcţiilor import sau export. HTML este un standard deschis, astfel, dacă un document este creat în HTML 3.2, majoritatea regiunilor lui vor fi afişate de orice navigator şi sub orice sistem de operare.

Un document HTML este un fişier text ASCII care conţine două tipuri de informaţii - *marcajele*, texte cuprinse între paranteze unghiulare(`< >`) şi *conţinutul*, textul în afara perechilor de paranteze unghiulare. Navigatorul nu afişează marcajele, dar le interpretează pentru afişarea conţinutului. Părţile individuale cuprinse între parantezele unghiulare se numesc marcaje (`tags`) sau comenzi. Există două tipuri de marcaje - de început şi de terminare - iar ele apar întotdeauna în pereche. Singura diferenţă între ele este aceea că marcajul de terminare are la început un caracter slash "`/`", de exemplu, `</B>`.

Sintaxa HTML - marcaje şi atribute

Un fişier HTML valid are câteva cerinţe de bază. Fie exemplul:

```
<HTML>
   <HEAD>
      <TITLE>cursul 1 pagina 1</TITLE>
   </HEAD>
   <BODY>Salutare

         </BODY>
</HTML>
```

Fişierul `c1p1.htm`, din **Figura 6**, conţine atât marcaje cât şi conţinut fiind vizualizat cu IE. Orice document HTML este structurat în două părţi: antetul (`header` - care nu se afişează) şi corpul (`body` - textul care este afişat de navigator). Structura generală a unui document este:

```
<HTML>                  -- începutul documentului HTML
   <HEAD>               -- începutul antetului documentului
      ...               -- conţinutul antetului
   </HEAD>              -- terminarea antetului
   <BODY>               -- începutul corpului documentului
      ...               -- conţinutul corpului
   </BODY>              -- terminarea corpului
</HTML>                 - terminarea documentului HTML
```

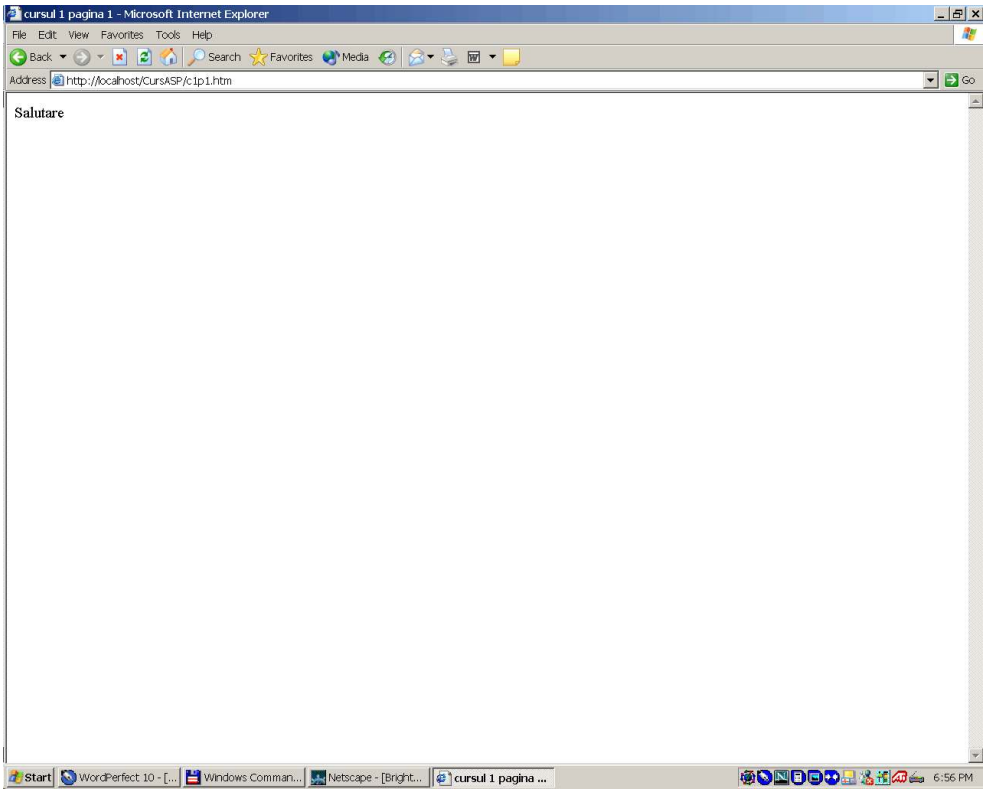


Figure 6
Modul de afişare de către IE a fişierului c1p1.htm

Fişierele HTML sunt formate din *tipuri de elemente* cum sunt paragraf, hiperlegături, liste, tabele etc. Un *tip de element* reprezintă o structură sau implementează un comportament predefinit. Fiecare tip de element este format din trei părţi: un *marcaj de început*, *conţinutul* şi un *marcaj de terminare*. Numele unui element apare în marcajul de început şi în cel de terminare precedat de caracterul `slash`. De exemplu, orice fişier HTML începe cu marcajul `<HTML>` şi se termină cu marcajul `</HTML>`. Între aceste marcaje pot exista alte marcaje şi conţinut. În documentaţia Microsoft, marcajele care conţin alte marcaje se numesc *elemente bloc* (`block elements`). *Elementele text* (`text elements`) se folosesc pentru a marca textul din interiorul elementelor bloc.

Un marcaj reprezintă o singură comandă, dar aceasta mai poate conţine şi un număr nelimitat de valori asociate comenzii, denumite *atribute*. Fiecare atribut are un *nume* şi o *valoare*. Atributul trebuie separat de comandă sau de alte atribute, prin spaţii albe (`white space` - unul sau mai multe spaţii, tab-uri sau întreruperi de linie). Folosirea mai multor spaţii albe este opţională, astfel `"Te salut!"` şi `"Te salut!"`, vor fi afişate identic de navigator. Scrierea cu litere mari sau mici, a marcajelor sau atributelor, este din nou nerelevantă pentru navigator. În acest sens `<font>` sau `<FONT>` vor avea aceeaşi semnificaţie. Iată câteva exemple în continuare:

```
<FONT FACE="Arial" SIZE=12>
```

Mai sus, marcajul `<FONT>` are două atribute - `FACE` şi `SIZE`, fiecare dintre ele luînd o valoare.

```
<INPUT TYPE="hidden" NAME="txtParam" value='doar un alt exemplu.'>
```

Pentru delimitarea unei valori de atribut se pot folosi atât ghilimelele (") cât şi apostroafele ('). Nu toate navigatoarele au nevoie de ghilimele sau apostroafe, în jurul valorilor de atribute, însă

folosirea lor devine utilă atunci când se trece la scrierea codului ASP.

Dacă veți încărca codurile de mai sus în navigator nu se va vedea nimic pe ecran întrucât acestea sunt marcaje şi nu conținut.

Ordinea marcajelor este importantă! Un marcaj deschis trebuie închis înainte de scrierea unui alt marcaj sau aşa încât să-l cuprindă pe primul. Intercalarea marcajelor este interzisă. De exemplu, `<FONT SIZE=12><I>Text inclinat</FONT></I>` este o construcție HTML greşită întrucât marcajul `<I>` apare înaintea marcajului `<FONT>`. Scrierea HTML corectă este: `<I><FONT SIZE=12>Text inclinat</FONT></I>`.

Pentru înțelegerea cât mai uşoară a codului HTML se pot folosi comentarii. Acesta are scrierea: `<!--Un comentariu-->`.

Caracterele `< >` şi alte caractere, cu semnificație specială în HTML, trebuie codificate pentru a putea fi afişate de navigator. Câteva sunt prezentate în tabelul care urmează:

Pentru afişarea lui	Se va scrie
>	>
<	<
"	"
&	&

Structura unui document HTML

Documentele HTML au o structură predefinită, iar cel care scrie un astfel de document este obligat să o respecte. Pe de o parte aceasta este o limitare dar, pe de altă parte, documentul devine mai uşor de parcurs şi de modificat, astfel întreținerea unui site devine mai simplă.

```
<HTML>
<HEAD>
<TITLE>Aici se scrie titlul documentului</TITLE>
<META NAME="descriere" CONTENT="Acesta este un document HTML">
<LINK REV="made" HREF="mailto:antaltiberiu@pcnet.ro">
</HEAD>
<BODY>
 ... corpul documentului
</BODY>
</HTML>
```

Marcajul <HTML>

Acest marcaj cuprinde întregul document. El nu este obligatoriu şi indică faptul că textul cuprins este HTML (atributul `VERSION` poate fi folosit pentru a specifica versiunea). Dacă toate informațiile despre document se scriu la început şi urmează imediat toate marcajele legate de corp se poate deduce unde anume se termină header-ul şi unde începe corpul (`body`) documentului.

Marcajul <TITLE>

Este singurul marcaj obligatoriu în secțiunea de `HEAD`. Fiecarea document HTML trebuie să aibă un singur titlu. Acesta este afişat în fereastra de titlu a navigatorului fiind folosit în fişiere bookmark şi în listele motoarelor de căutare. Un exemplu este:

```
<TITLE>O introducere in HTML</TITLE>
```

Marcajul <META>

Marcajul furnizează "meta-informații" despre document sub forma unor perechi de tipul nume/valoare, dar poate fi folosit şi pentru specificarea antetului de document. De exemplu, se poate specifica o descriere a documentului şi a programul folosit pentru generarea lui sau se pot furniza date cu privire la autorul lui. Atributele care pot să apară sunt: `HTTP-EQUIV=şir | NAME=şir, CONTENT=şir.NAME` se foloseşte pentru a specifica tipul proprietarului, iar `CONTENT` stochează valoarea corespunzătoare acestuia. `HTTP-EQUIV` se poate folosi înaintea lui `NAME` pentru a obliga server-ul să formeze un antet al răspunsului HTTP care respectă stilul de antet RFC 822. Acesta poate fi folosit de anumite chache-uri pentru a determina expirarea documentului. Câteva exemple sunt:

```
<META NAME="Autor" CONTENT="Soricel Alb">
```

Din cauza liniilor:

```
<META HTTP-EQUIV="Expira" CONTENT="Tue, 04 Dec 2003 21:29:02 GMT">
<META HTTP-EQUIV="Cuv-ch" CONTENT="Pagini Web, ASP, VBScript">
<META HTTP-EQUIV="Adr" CONTENT="antaltibi@pcnet.ro (Antal Tibi)">
```

server-ul trebuie să includă în antetul răspunsului următoarele informații:

```
Expira: Tue, 04 Dec 2003 21:29:02 GMT
Cuv-ch: Pagini Web, ASP, VBScript
Adr: antaltibi@pcnet.ro (Antal Tibi)
```


Marcajul <LINK>

Acest marcaj stochează informații de pagină relative la celelalte părți ale site-ului. De exemplu, se poate folosi un marcaj `LINK` pentru a specifica locația conținutului site-ului (`toc - table of contents`), care este următorul document (`next document`) sau care este locația foii cu stiluri (`style sheet`). Forma generală este `<LINK REL=şir HREF=URL>`, iar atributele care pot să apară sunt: `REL=şir, REV=şir, HREF=URL, TITLE=şir`. Două tipuri de relații sunt permise: `REL` indică o relație directă, adică de la documentul curent la cel specificat în `URL`; `REV` indică o relație inversă, adică de la documentul specificat în `URL` la cel curent. Atributul `TITLE` se foloseşte pentru a da titlul relației sau URL-ului referit. Unele dintre valorile pe care le ia `REL` au un efect predefinit (mai jos, acestea sunt `next` şi `previous`), iată câteva exemple:

```
<LINK REL=continut HREF=toc.html>
<LINK REL=previous HREF=doc31.html>
<LINK REL=next HREF=doc33.html>
<LINK REL=capitol REV=continut HREF=chapter2.html>
```

Marcajul <BODY>

Marcajul `BODY` cuprinde corpul documentului. Un document poate avea un singur `BODY`. Acesta poate conține următoarele elemente:

- antete (heading, H1 - H6);
- elementul ADDRESS;
- elemente bloc;
- elemente text.

Atributele marcajului `BODY` sunt: `BACKGROUND=URL, BGCOLOR=#RRGGBB, TEXT=#RRGGBB, LINK=#RRGGBB, VLINK=#RRGGBB` şi `ALINK=#RRGGBB`. Acestea se pot folosi pentru a seta imagini de fundal (`background`) care se repetă, culoarea de fond, cea de scris pentru text şi cea pentru hiperlegături. Culorile se formează prin specificarea componentelor de roşu (`red`), verde (`green`) şi albastru (`blue`) în notație hexazecimală, precedate de caracterul #. De exemplu, pentru a

specifica culoarea albă componentele iau valorile zecimale (maxime) 255, 255, 255, iar în hexazecimal #FFFFFF. De asemenea, este corectă folosirea următoarelor nume, în limba engleză, pentru specificarea unei culori:

Culoare	Cod	Culoare	Cod	Culoare	Cod	Culoare	Cod
black	#000000	green	#008000	silver	#C0C0C0	lime	#00FF00
gray	#808080	olive	#808000	white	#FFFFFF	yellow	#FFFF00
maroon	#800000	navy	#000080	red	#FF0000	blue	#0000FF
purple	#800080	teal	#008080	fuchsia	#FF00FF	aqua	#00FFFF

Iată un exemplu:

```
<body bgcolor=white text=black link=red vlink=maroon alink=fuchsia>
```

Marcajul BODY este opțional. Dacă toate elementele din HEAD se scriu la început, navigatorul poate să determine locul din care începe corpul (BODY) documetului.

Formatarea textului

HTML face ca formatarea textului să fie foarte simplă, cu condiția ca să nu conteze foarte mult modul lui afişare, locul în care apare trecerea la linie nouă sau poziția relativă a textului în raport cu alte elemente din pagină. Modul de afişarea a textului este controlat prin folosirea stilurilor de antet (header style), tipuri de caractere (fonts), culori (colors) paragrafe (paragraph) şi liste (lists).

Stiluri de antete

HTML recunoaşte şase nivele de antete, scrise <H1> , <H2> până la <H6>. Numărul semnifică poziția antetului dintr-o ierarhie; cu cât valoarea lui este mai mică cu atât conținutul este într-o ierarhie mai înaltă. Majoritatea editoarelor de texte lucrează în acelaşi fel cu stilurile de antete. Codul paginii HTML din **Figura 7** este prezentat în continuare:

```
<HTML>
<HEAD>
<TITLE>Stiluri de antete</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
<H1>Heading 1</H1>
	<H2>Heading 2</H2>
		<H3>Heading 3</H3>
			<H4>Heading 4</H4>
				<H5>Heading 5</H5>
					<H6>Heading 6</H6>
<HR>
<H1>Ce este ASP?</H1>
	<H2>De la HTML la ASP</H2>
	<H2>CGI</H2>
	<H2>Avantaje ASP</H2>
		<H3>Independenta de limbajul de programare</H3>
		<H3>ASP este simplu de invatat</H3>
	<H2>Alte metode pentru crearea de pagini Web dinamic</H2>
		<H3>Programe CGI</H3>
		<H3>Ce este ISAPI?</H3>
			<H4>Aplicatii ISAPI</H4>
			<H4>Filtre ISAPI</H4>
</BODY>
```

```
</HTML>
```

Figure 7
Cele 6 nivele de antet

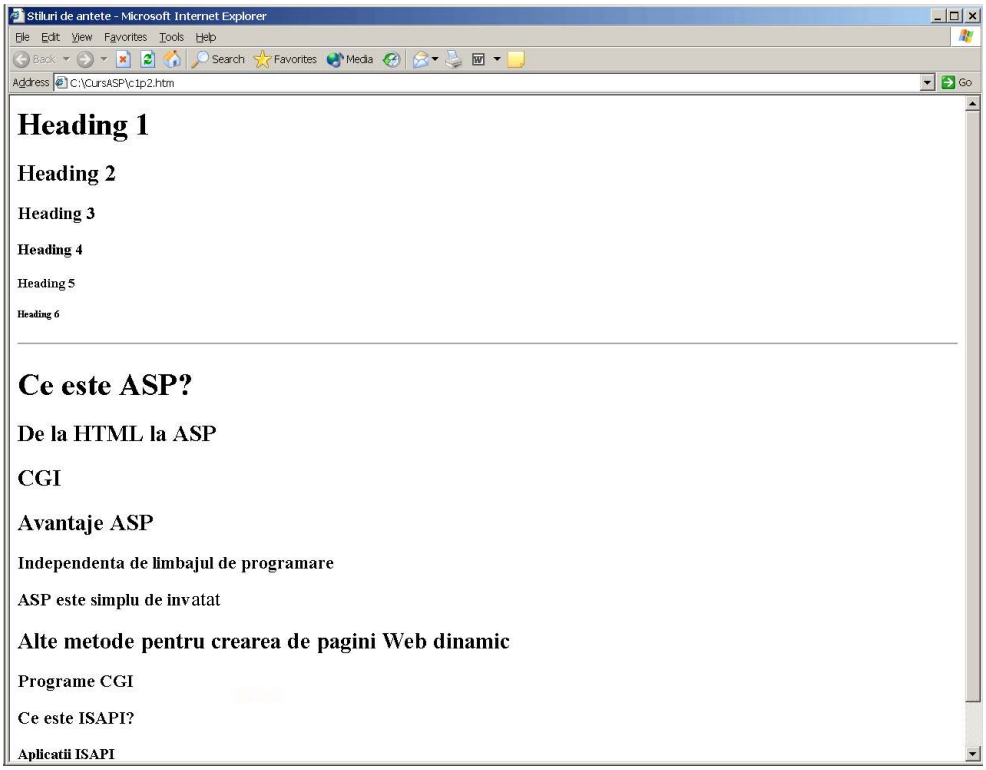


Figura7 arată modul de afişare a celor şase nivele de antete în IE6. Modul de afişare este specific navigatorului, astfel că nivelele de antete nu vor fi afişate în acelaşi mod de două navigatoare diferite.

Aliniere, tipuri de caractere

Elementele bloc (tabele, imagini, paragrafe etc.) pot fi aliniate prin folosirea atributului `ALIGN`. Deşi atributul poate fi scris în majoritatea elementelor HTML, valorile pe care le poate lua diferă în funcţie de element. În continuare se va prezenta semnificaţia lui `ALIGN` numai în cazul textului. Valorile posibile în acest caz sunt: `LEFT`, `CENTER`, `RIGHT`, `JUSTIFY` şi au efectul de aliniere la stânga, pe centru, la dreapta sau între marginile ferestrei textului marcat. Iată câteva exemple:

```
<H1 ALIGN="CENTER"> Cum sa nu pierzi timpul ... </H1>
<P ALIGN="RIGHT">...Text al unui pragaraf...
```

Elementul `FONT` modifică tipul de caractere ale textului marcat, iar `BASEFONT` realizează aceeaşi modificare la nivelul întregului document. Valoarea implicită a lui `BASEFONT` este `3`. Iată un exemplu:

```
<P><FONT SIZE=1>SIZE=1</FONT>
<FONT SIZE=2>SIZE=2</FONT>
<FONT SIZE=3>SIZE=3</FONT>
<FONT SIZE=7>SIZE=7</FONT>
```

Containere de text

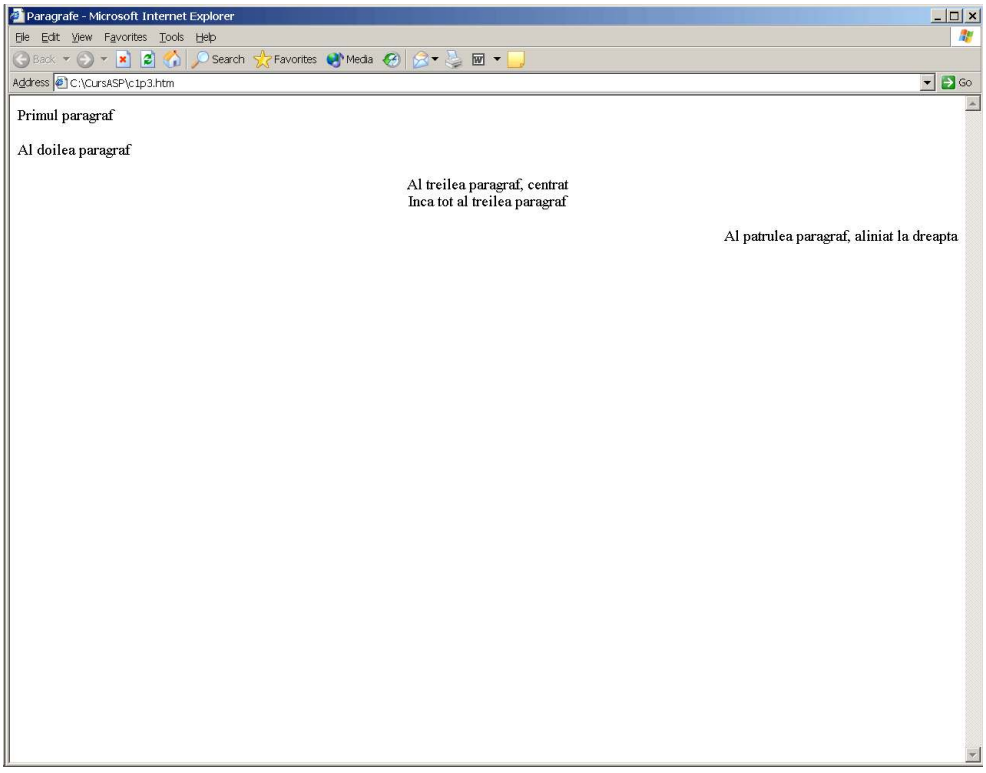
Elementele de tipul container au un marcaj de început şi unul de terminare. Ca urmare a acestei delimitări clare ele pot conţine date.

Paragraful

`P` se foloseşte pentru marcarea unui paragraf. Marcajul de terminare este opțional întrucât poate fi întotdeauna dedus de către navigator. Atributul opțional `ALIGN` permite specificarea modului de aliniere a conținutului paragrafului. Implicit, acesta este aliniat la stânga (`LEFT`). Valori de aliniere corecte sunt şi `CENTER`, `RIGHT`. Marcajele de tipul bloc pot conține alte marcaje numite, marcaje copil, cum ar fi text, imagini sau comenzi de formatare. Pagina din **Figura 8** are codul HTML următor:

```
<HTML>
<HEAD>
<TITLE>Paragrafe</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
<P>Primul paragraf
<P>Al doilea paragraf
<P ALIGN=CENTER>Al treilea paragraf, centrat
<BR>
Inca tot al treilea paragraf
<P ALIGN=RIGHT>Al patrulea paragraf, aliniat la dreapta
</BODY>
```

Figure 8 Afişarea paragrafelor



Modul de afişare a paragrafului este din nou dependent de agentul de afişare. La nivelul paragrafului al treilea s-a inclus un marcaj încă nediscutat. În lipsa lui navigatorul va afişa textul `Inca tot al treilea paragraf` în continuarea lui `Al treilea paragraf, centrat`.

Text preformatat

Marcajul `PRE` se foloseşte pentru a include în corpul documentului porțiuni de text în care formatarea este critică. Spre deosebire de alte containere HTML, textul marcat cu o pereche `PRE` va fi monospațiat, trecerea pe linia următoare făcându-se la apăsarea lui **<Enter>** (`line break`) în textul sursă. Se pot folosi chiar tabulatori, deşi, în acest caz, folosirea spațiilor va avea acelaşi efect. Ca urmare a modului de afişare a textului cu caractere monospațiate nu se pot face setări

de tipuri de caractere (`fonts`). Imaginile nu sunt permise întrucât pot conduce la probleme de aliniere. Singurul argument, opțional, este `WIDTH=n` care indică lățimea textului în număr de caractere (pentru un fişier text setarea caracteristică de lățime este `WIDTH=80`).

Un exemplu de text preformatat este:

```
<HTML>
<HEAD>
<TITLE>Textul preformatat</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>

<PRE>
      Acest text va fi afisat exact asa cum este scris aici
                        fiind pastrata inclusiv alinierea
      sau scrierea      pe coloane
      dupa              cum se vede
      Se pot scrie      si linii vide (fara text)

      Se pot folosi marcajele <B>de text ingrosat</B> sau <I>inclinat</I> in
      textul preformatat.
</PRE>

</BODY>
```

El va fi afişat de navigator sub forma:


```
      Acest text va fi afisat exact asa cum este scris aici
                        fiind pastrata inclusiv alinierea
      sau scrierea      pe coloane
      dupa              cum se vede
      Se pot scrie      si linii vide (fara text)

      Se pot folosi marcajele de text ingrosat sau inclinat in
      textul preformatat.
```


Stiluri de liste

Listele HTML pot fi ordonate (`ordered`) sau neordonate (`unordered`). Ele pot conține elemente de tipurile bloc, text sau alte liste.

Lista neordonată

Marcajul `UL` indică o listă neordonată. Fiecare articol de listă trebuie marcat cu `LI`, caz în care elementul este precedat de un `disc` sau, dacă este nevoie de numerotare, se poate folosi marcajul `OL`. Tipul caracterului care precede conținutul articolului de listă poate fi stabilit prin atributul `TYPE` care poate lua valorile `disc`, `square` sau `circle`. Iată un exemplu în continuare:

```
<HTML>
<HEAD>
<TITLE>Liste neordonate</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
<UL>Urmeaza lista neordonata de articole:
      <LI>Articol 1</LI>
      <LI>Articol 2</LI>
      <LI>Articol 3</LI>
</UL>
</BODY>
```

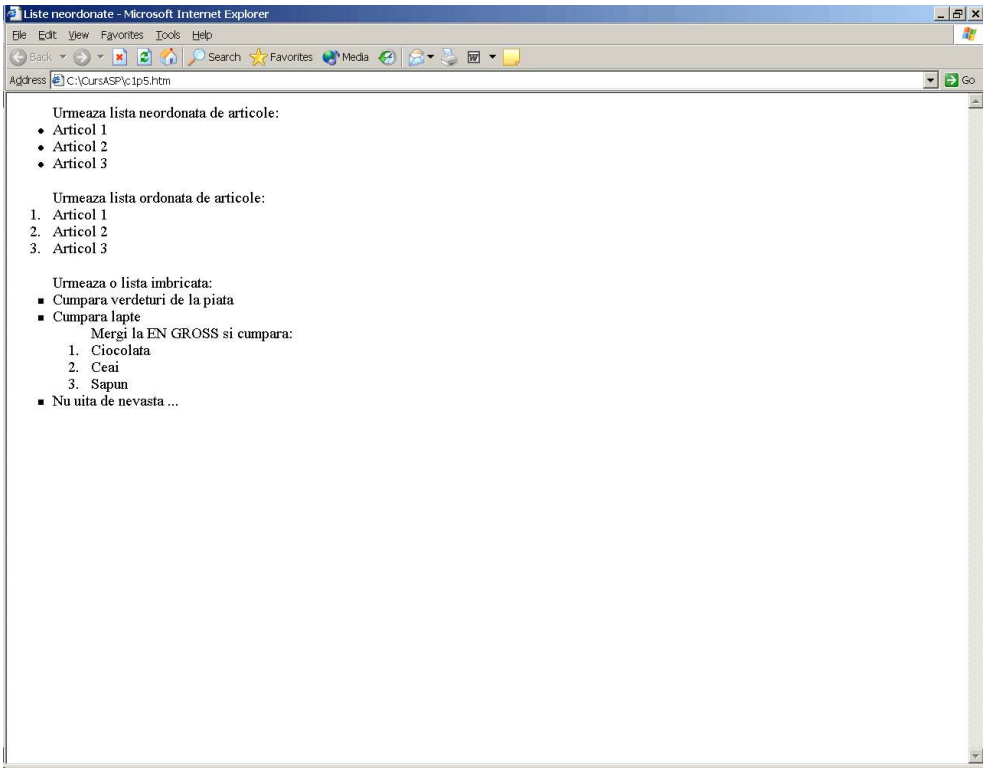
Lista ordonată

În cazul în care dorim ca elementele listei să fie numerotate se va folosi marcajul `OL`. De

asemenea, listele pot fi imbricate, adică în locul unui articol de lista se poate scrie o nouă listă. În **Figura 9**, se prezintă modul de afişare a listelor de către IE6, conform exemplului care urmează.

```
<HTML>
<HEAD>
<TITLE>Liste neordonate</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
<UL>Urmeaza lista neordonata de articole:
      <LI>Articol 1</LI>
      <LI>Articol 2</LI>
      <LI>Articol 3</LI>
</UL>

<OL>Urmeaza lista ordonata de articole:
      <LI>Articol 1</LI>
      <LI>Articol 2</LI>
      <LI>Articol 3</LI>
</OL>

<UL TYPE=square>Urmeaza o lista imbricata:
      <LI>Cumpara verdeturi de la piata</LI>
      <LI>Cumpara lapte</LI>
      <OL>Mergi la EN GROSS si cumpara:
                  <LI>Ciocolata</LI>
                  <LI>Ceai</LI>
                  <LI>Sapun</LI>
      </OL>
      <LI>Nu uita de nevasta ...</LI>
</UL>
</BODY>
```


Figure 9
Afişarea listelor în HTML

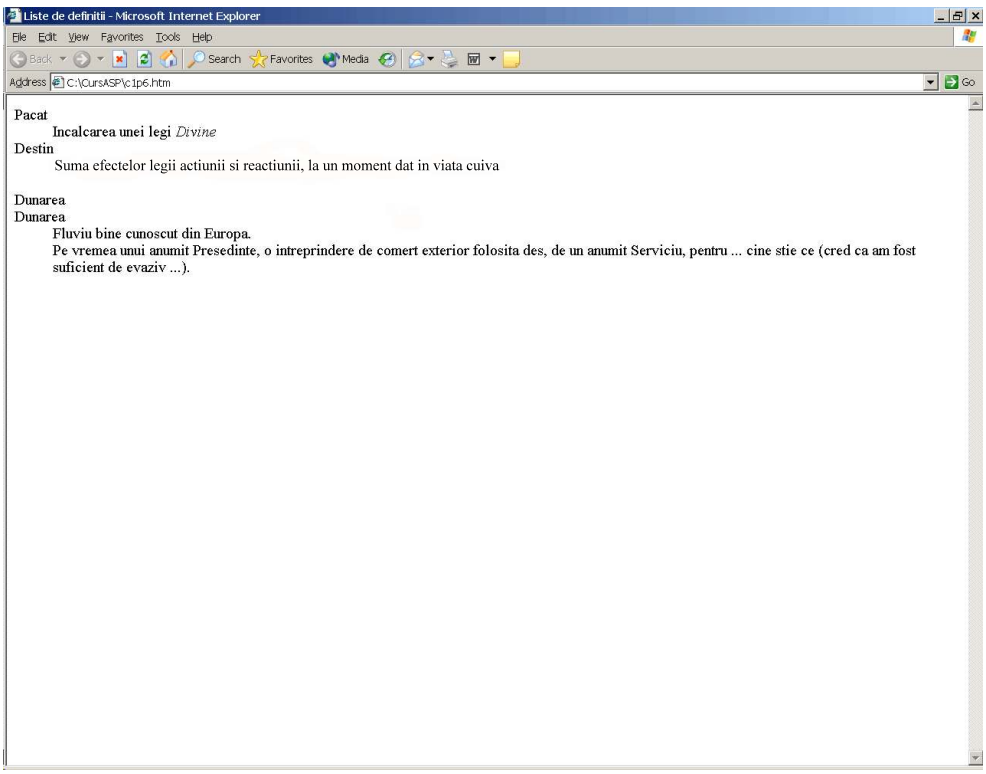


Definiţii

Marcajul DL permite crearea unei liste cu definiții. Acest stil de listă se deosebeşte de cele precedente prin faptul că fiecare articol al listei are două părți: termenul care se defineşte (începe cu DT) şi definiția acestuia (începe cu DD). Definiția poate conține elemente bloc. Cele două exemple care urmează sunt afişate de IE6 aşa cum se vede în **Figura 10**. Cea de a doua listă este un exemplu de creare a unei liste cu definiții şi termeni multipli.

```
<HTML>
<HEAD>
<TITLE>Liste de definitii</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>

<DL>
  <DT>Pacat
      <DD>Incalcarea unei legi <EM>Divine</EM>
  <DT>Destin
      <DD>Suma efectelor legii actiunii si reactiunii, la un moment dat, in
viata cuiva
</DL>

<DL>
   <DT>Dunarea
   <DT>Dunarea
      <DD>Fluviu bine cunoscut
             din Europa.
      <DD>Pe vremea unui anumit Presedinte, o
            intreprindere de comert exterior folosita des,
            de un anumit Serviciu,
            pentru ... cine stie ce (cred ca am fost suficient de evaziv ...).
</DL>
</BODY>
```


Figure 10
Liste cu definiții



Alte elemente

Câteva dintre elementele mai utile sunt prezentate în continuare:

- <HR> (Horizontal Rule) duce la afişarea unei linii orizontale; din exemplul care urmează se pot deduce semnificația şi modul de utilizare a atributelor:

```
<HR width="50%" align="center">
<HR size="5" width="50%" align="center">
<HR noshade size="5" width="50%" align="center">
```

-
 (BReak) realizează întreruperea unei linii, adică realizează trecerea la linia următoare, dar nu lasă spații între linii (aşa cum se face în cazul paragrafelor);

- <B> (Bold) realizează afişarea **îngroşată** a porțiunii de text marcate;

- <I> (Italic) realizează afişarea cu *litere înclinate* a porțiunii marcate;

- <TT> (TeleType) stil de scriere teletext, textul afişat este monospațiat;

- <BIG> stil de scriere în care caracterele sunt afişate mai mari decât cele normale;

- <SMALL> stil de scriere în care caracterele sunt afişate mai mici decât cele normale;

- <U> (Underline) stil de scriere în care textul marcat este afişat subliniat.

Iată un exemplu de utilizare în continuare:

```
<P><B>text ingrosat</B>,
<I>text inclinat</I>, <B><I>text ingrosat si inclinat</I></B>, <TT>text
teletype</TT>, si
text <BIG>mare</BIG> si <SMALL>text</SMALL> mic.
```

Includerea imaginilor în documentele HTML

Majoritatea paginilor Web conţin imagini. Includerea acestora într-un document este foarte simplă. Textul şi imaginile pot fi aranjate, pentru a ocupa spaţiul de afişare, în mai multe feluri cu ajutorul marcajului `<IMG>`.

Marcajul <IMG>

O imagine este inclusă într–un document HTML prin folosirea lui `IMG` împreună cu atributul `SRC=URL`. Cea mai simplă formă este `<IMG SRC="numefis.gif">` şi realizează afişarea fişierului `numefis.gif` în paragraful curent.

Dacă documentul conţine mai multe imagini este posibil ca acestea să nu fie afişate în ordinea scrierii lor în pagină. Navigatorul cere imaginile secvenţial, dar server-ul poate să trimită răspunsurile într-o altă ordine.

Marcajul are câteva atribute opţionale. Lăţimea şi înălţimea pot fi explicit specificate prin atributele `WIDTH="număr"` şi `HEIGHT="număr"`. În lipsa lor navigatorul afişează imaginea la dimensiunile originale. Folosirea lor va ajuta navigatorul să realizeze arajarea textului şi a imaginilor în pagină. Dacă dimensiunea unei imagini nu este cunoscută navigatorul afişează simbolul de imagine lipsă (`missing image icon`) pe locul acesteia până când aceasta este descărcată de pe server. Dacă dimensiunile sunt cunoscute atunci navigatorul va putea face calcule cu privire la aranjarea obiectelor în pagină. În lipsa dimensiunilor imaginilor este posibil ca bucăţi de text să fie mutate în pagină ca urmare a estimării greşite a mărimii imaginilor. Rezultatul final va fi o pagină care se încarcă mai încet decât aceea care conţine marcajele `WIDTH` şi `HEIGHT`.

Majoritatea imaginilor din paginile HTML sunt în formatele `GIF` (`Grafical Interchange Format`) sau `JPEG` (`Joint Photographical Experts Group`) deoarece mărimea acestora, ca urmare a comprimării, este semnificativ mai mică decât a altor formate grafice. Din punct de vedere tehnic nu există limitări cu privire la formatul fişierelor de imagine, dar este posibil ca navigatorul să nu le poată afişa decât cu ajutorul unor componente de vizualizare (`viewers`). Nativ, Netscape, ştie afişa fişiere `GIF` şi `JPEG`, iar IE ştie afişa, în plus şi formatul `BMP` (`bitmap`). Celelalte formate se afişează de către IE cu folosirea unor componente plug-in (un fişier care conţine date pentru modificarea, îmbunătăţirea sau extinderea operaţiilor unei aplicaţii program părinte; plug-in-ul este specific unui anumit sistem de operare şi afişează sau interpretează un format particular de fişier cum ar fi Shockwave, RealAudio, Adobe PDF, Macromedia Flash etc.) sau extensii ActiveX. Netscape, ştie să lucreze doar cu plug-in-uri, iar cu ajutorul unor componente software poate ajunge să lucreze şi cu controale ActiveX. Spre deosebire de programele executabile obişnuite, lipsa unei anumite resurse nu duce la blocarea afişării, navigatorul ignorând toate resursele pe care, din anumite motive, nu le poate accesa. În plus faţă de atributele deja discutate, în tabelul care urmează se prezintă câteva dintre valorile pe care le poate lua atributul de aliniere a imaginii `ALIGN`.

Valoarea atributului de aliniere	Rezultat
`ABSBOTTOM`	Aliniează imaginea relativ la punctul cel mai de jos posibil din text.
`ABSMIDDLE`	Aliniează imaginea în centru absolut al textului.

Valoarea atributului de aliniere	Rezultat
BASELINE	Aliniează imaginea la linia de bază a textului.
BOTTOM	Aliniează imaginea la partea de jos a marcajului în care apare.
LEFT	Aliniează imaginea la partea stîngă a marcajului în care apare.
MIDDLE	Aliniează imaginea la mijlocul marcajului în care apare.
RIGHT	Aliniează imaginea la dreapta marcajului în care apare.
TEXTTOP	Aliniează inaginea la marginea de sus a textului
TOP	Aliniează imaginea la marginea de sus a marcajului în care apare
BORDER	O valoarea întreagă care determină grosimea chenarului. Valoarea implicită este 1. Valoarea 0 marchează lipsa chenarului.
HSPACE	O valoare întreagă ce determină spaţiul între marginea stângă şi dreaptă a imaginii şi obiectele din jurul acesteia.
VSPACE	O valoare întreagă ce determină spaţiul între marginea de sus şi de jos a imaginii şi obiectele din jurul acesteia.

Plasarea imaginilor în pagină

Implicit, navigatorul plasează imaginea pe pagină în poziţia ce rezultă din analiza documentului HTML din care face parte. Există însă acţiuni care pot modifica poziţia implicită. Dacă, de exemplu, primul marcaj din pagină este `IMG` imaginea va fi plasată în colţul stânga sus al paginii clientului. Pagina client este zona din fereastră navigatorului în care se afişează conţinutul documetului. În cazul exemplului prezentat în continuare orice text care urmează imaginii va fi afişat în continuarea marginii din dreapta jos a imaginii, deoarece modul de plasarea a ei în pagină mută linia de bază în partea de jos a imaginii (vezi **Figura 11**).

```
<HTML>
<HEAD>
<TITLE>Plasarea imaginilor in pagina</TITLE>
</HEAD>
<BODY>
<IMG SRC="./imagini/NUFAR.JPG" WIDTH="227" HEIGHT="179">
Aceasta imagine se numeste NUFAR.JPG si este stocata in subdirectorul IMAGINI.
</BODY>
```

Dacă se doreşte modificarea setării implicite prin adăugarea unui atribut de aliniere, poziţia imaginii se va modifica. De exemplu, prin scrierea `<IMG SRC="./imagini/NUFAR.JPG" ALIGN="RIGHT" WIDTH="227" HEIGHT="179">` imaginea va fi aliniată în colţul din dreapta sus al ferestrei clientului. Dacă se adaugă text după imagine, ne aşteptăm ca acesta să fie afişat pe linia imediat următoare imaginii, începând din partea de jos a imaginii. În realitate însă modificarea alinierii imaginii o plasează pe aceasta corect, la dreapta, dar textul va fi afişat de navigator începând cu colţul stânga sus al ferestrei client. Dacă valoarea atributului de aliniere se modifică de la `RIGHT` la `LEFT`, textul va fi afişat în continuarea colţului dreapta sus al imaginii şi nu plecând din colţul dreapta jos, aşa cum ne-am aştepta. Pentru ca imaginea să fie afişată într-un paragraf separat de text trebuie pus un paragaraf nou înainte şi unul după imagine sau se pun `<BR>`-uri înainte şi după imagine.

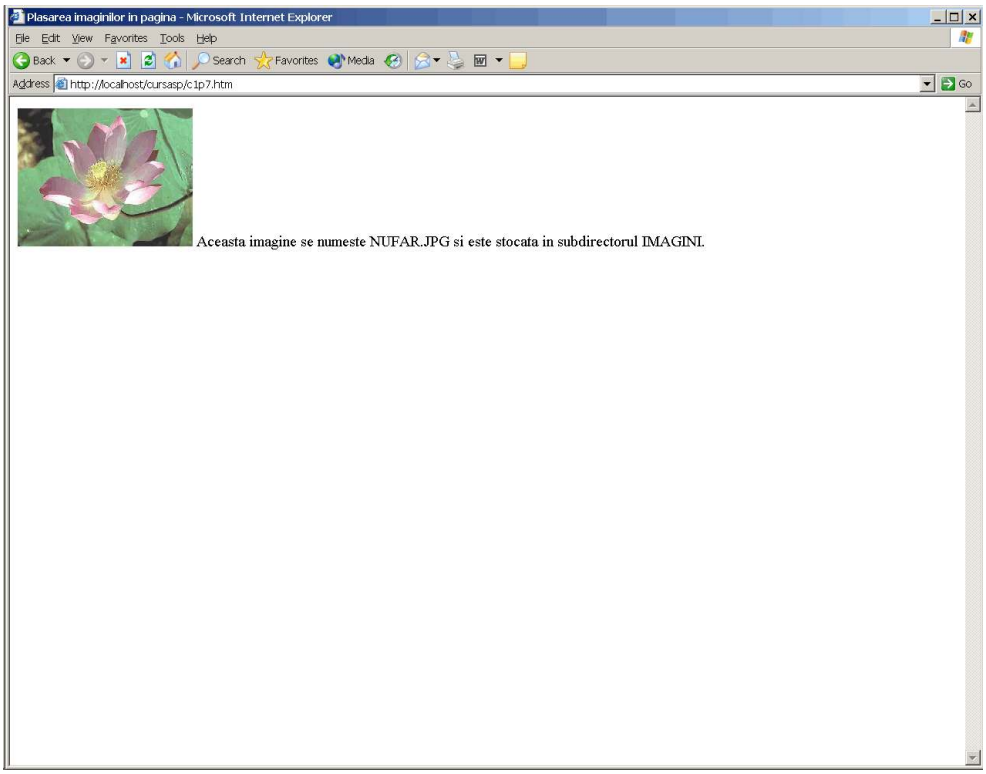


Figure 11 Plasarea implicită a imaginilor

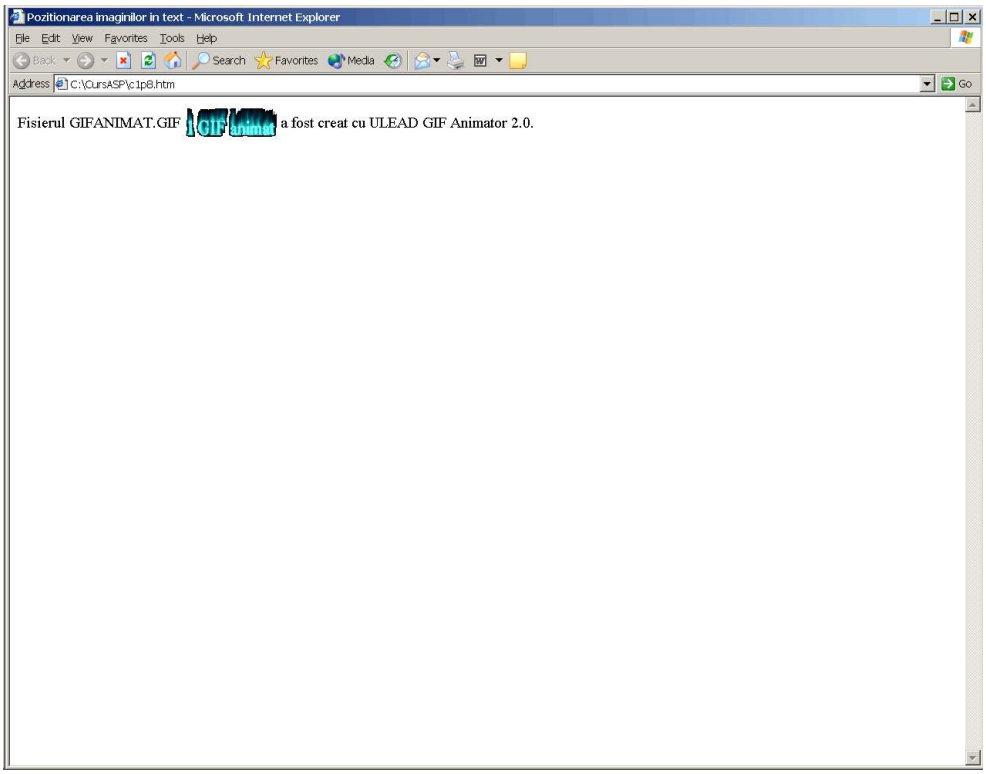
Ocuparea cu text a spaţiului din jurul imaginii

HTML nu permite poziționarea unei imagini în texte în așa fel încât textul să ocupe spațiul rămas, atât în dreapta şi cât şi în stânga, între marginile de afişare şi cele ale imaginii. Textul poate să ocupe, însă, numai partea din stânga sau numai pe cea din dreapta, prin setarea atributului `ALIGN` la una dintre valorile `LEFT` sau `RIGHT`. O soluție acceptabilă se prezintă în pagina din **Figura 12** al cărei cod este:

```
<HTML>
<HEAD>
<TITLE>Pozitionarea imaginilor in text</TITLE>
</HEAD>
<BODY>
Fisierul GIFANIMAT.GIF <IMG SRC="./imagini/GIFANIMAT.GIF" ALIGN="CENTER"
VALIGN="CENTER" WIDTH="120" HEIGHT="50"> a fost creat cu ULEAD GIF Animator
2.0.
</BODY>
</HTML>
```

Această strategie funcționează însă numai în cazul unor imagini mai mici şi realizează intercalarea ei, într-un text, prin centrarea imaginii în spațiul de afişare definit de atributele `WIDTH` şi `HEIGHT`.

Figure 12 Inserarea imaginilor în text



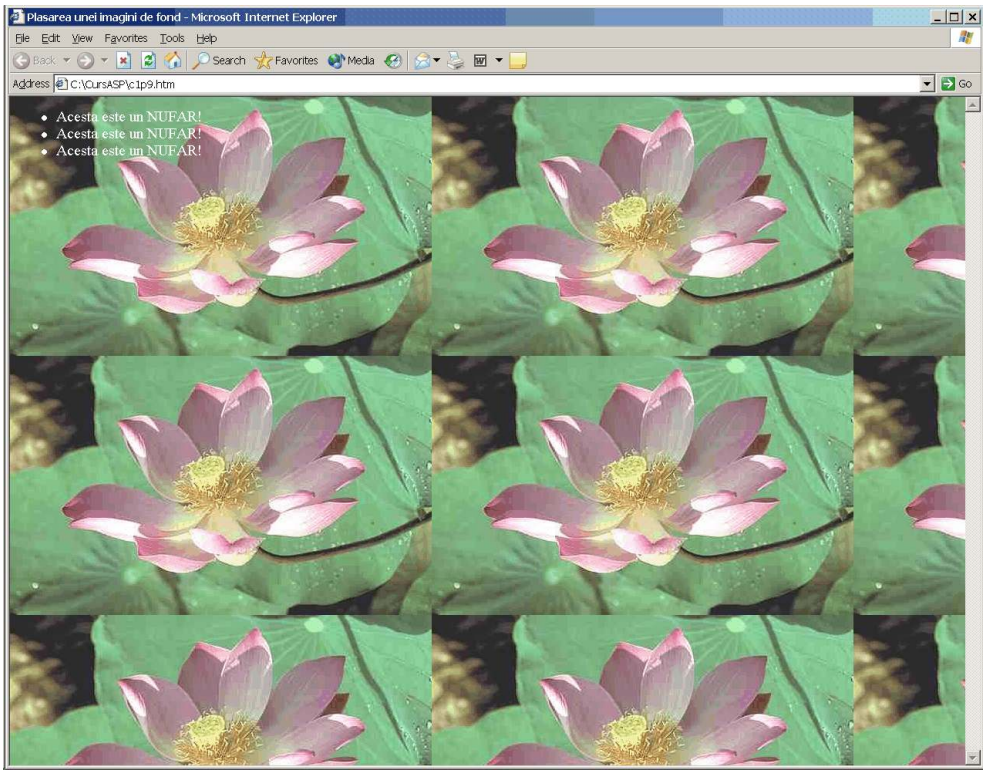
Imagini de fond

Atât IE cât şi Netcape permit folosirea unei imagini pentru definirea fondului de pagină. Imaginea este specificată ca atribut al marcajului `BODY` după cum urmează:

```
<HTML>
<HEAD>
<TITLE>Plasarea unei imagini de fond</TITLE>
</HEAD>
<BODY BACKGROUND="./imagini/NUFAR.JPG" TEXT=#FFFFFF>
      <P ALIGN="CENTER">
      <UL>
            <LI>Acesta este un NUFAR!</LI>
            <LI>Acesta este un NUFAR!</LI>
            <LI>Acesta este un NUFAR!</LI>
      </UL>
      </P>
</BODY>
</HTML>
```

În **Figura 13** se observă modul în care imaginea ocupă spaţiul de afişare. Aceasta este afişată de mai multe ori, fiecare imagine fiind imediat alăturată celei anterioare. Afişarea începe din colţul stânga sus, apoi imaginea se repetă pe toată lăţimea paginii. Când se ajunge la marginea din dreapta, afişarea reîncepe din marginea stîngă imediat sub prima imagine. Această tehnică de afişare este folosită deoarece, deseori, fondul este o imagine foarte simplă şi mică prin care se dă o textură şi o culoare fondului. IE permite şi o afişare specială a unei imagini de fond, fără repetarea acesteia după şablonul deja descris, de obicei, centrată pe vericală şi orizontală care are ordinea 0 pe axa z. Astfel, toate celelalte elemente care se afişează se pun peste această imagine (denumirea acestei tehnici de afişare, în engleză, este aceea de `watermark`).

Figure 13
Imagini de fond



Realizarea hiperlegăturilor

Hiperlegăturile asigură, la un nivel minimal, controlul utilizatorului asupra aplicației, mai precis asupra locului în care utilizatorul poate să ajungă pe ecran sau în document. Hiperlegăturile sunt o modalitate prin care este posibilă poziționarea pe o nouă locație sau topică. Pe măsură ce utilizatorul parcurge mulțimea de hiperlegături navigatorul formează o listă cu istoria locațiilor vizitate. Aceasta se poate folosi pentru travesarea inversă a locațiilor parcurse.

Marcajul de ancorare

Marcajul `A` permite legarea unei locații de o alta, prin specificarea unui URL. URL-ul este specificat ca o valoare a atributului `HREF`. Iată un exemplu de scriere:

```
<A HREF="http://server/site.pagina.htm">
Un text, de exemplu salt la o pagina
</A>
```

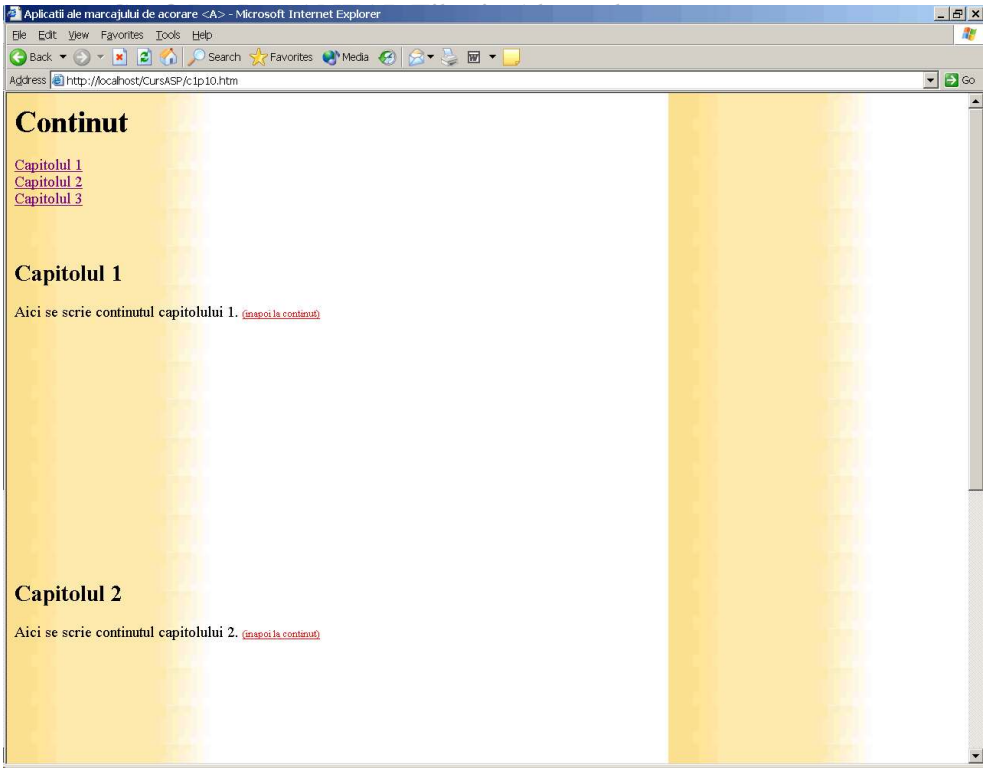
Navigatorul afişează pe ecran numai textul `Un text, de exemplu salt la pagina` dintre începutul marcajului `<A>` şi sfârsitul lui, formatat astfel Un text, de exemplu salt la o pagina. Acest text este cel pe care se fac clic pentru a realiza hiperlegătura. Există două tipuri de marcaje pentru ancorare - legătura (`link`) şi conținutul (`bookmark`). Marcajul de legătură este cel care activează o hiperlegătură, efectul acesteia este cel de realizare a procesului de legare. Marcajul de conținut este destinația unei legături şi nu este afişat pe ecran. Navigatoarele afişează în culori distincte o legătură după ce a fost vizitată față de culoarea de afişare folosită înainte de vizitare.

Pentru crearea unui marcaj de conținut trebuie folosit atributul de ancorare `NAME` după cum urmează: `<A NAME="Marcaj1">`. Se poate realiza saltul la un marcaj în acelaşi document sau unul aflat într-un alt document. Pentru a sări la marcajul de conținut definit mai sus scriem: `<A HREF="#Marcaj1">`. Semnul `#` din fața valorii atributului `HREF` informează navigatorul că pagina de legătură este un marcaj de conținut şi nu un alt document. Dacă se doreşte saltul la un marcaj de conținut al unui alt document va trebui specificat şi URL-ul documetului după cum urmează: `<A HREF="http://server/site.pagina.htm#Marcaj1">`. Exemplul care urmează este afişat de IE după cum se vede în **Figura 14.** El prezintă modul de realizare a unui document care are cuprins. Secvența `<P> </P>` se foloseşte pentru a crea, artificial, spații între capitole. Astfel, capitolele nu pot fi afişate pe o singură pagină, iar efectul salturilor între pagini devine vizibil.

```
<HTML>
<HEAD>
<TITLE>Aplicatii ale marcajului de acorare <A></TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080✂
 BACKGROUND="./imagini/Fond.jpg">
      <A NAME="Marcaj1">
      <H1>Continut</H1></A>
      <A HREF="#Marcaj2">Capitolul 1</A><BR>
      <A HREF="#Marcaj3">Capitolul 2</A><BR>
      <A HREF="#Marcaj4">Capitolul 3</A><BR>
      <P> </P>
      <A NAME="Marcaj2">
      <H2>Capitolul 1</H2></A>
      Aici se scrie continutul capitolului 1.
      <A HREF="#Marcaj1"><FONT SIZE="1" COLOR="RED">(inapoi la✂
 continut)</FONT></A><BR>
            <P> </P>
            <P> </P>
```

```
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
      <A NAME="Marcaj3">
      <H2>Capitolul 2</H2></A>
      Aici se scrie continutul capitolului 2.
      <A HREF="#Marcaj1"><FONT SIZE="1" COLOR="RED">(inapoi la✂
 continut)</FONT></A><BR>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>

      <A NAME="Marcaj4">
      <H2>Capitolul 3</H2></A>
      Aici se scrie continutul capitolului 3.
      <A HREF="#Marcaj1"><FONT SIZE="1" COLOR="RED">(inapoi la✂
 continut)</FONT></A><BR>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
            <P> </P>
</BODY>
```


Figure 14 Ancorarea la nivelul unui document



Conţinutul fişierului `c1p11a.htm`:

```
<HTML>
<HEAD>
```

```
<TITLE></TITLE>
</HEAD>
<BODY>

<P>Care dintre urmatoarele afirmatii <EM>nu</EM> este corecta?</P>
<OL>
  <LI><A HREF="c1p11b.htm">Rosu cu verde da albastru.</A></LI>
  <LI><A HREF="c1p11c.htm">Rosu cu albastru da verde.</A></LI>
  <LI><A HREF="c1p11d.htm">#00FFFF este albastru.</A></LI>
</OL>
<P>Clic pe raspuns.</P>

</BODY>
</HTML>
```

Conținutul fişierului `c1p11b.htm`:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>

<P>Incorect!</P>
<P>Ati selectat: &quot;1. Rosu cu verde da albastru.&quot; Rosu cu verde da✂
 galben.
Clic <A HREF="c1p11a.htm">pentru a continua</A> cu reluarea intrebarii.</P>
```



```
</BODY>
</HTML>
```

Conținutul fişierului `c1p11c.htm`:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>

<P>Incorect!</P>
<P>Ati selectat: &quot;2. Rosu cu albastru da verde; Rosu cu albastru da mov.
Clic <A HREF="c1p11a.htm">pentru a continua</A> cu reluarea intrebarii.</P>

</BODY>
</HTML>
```

Conținutul fişierului `c1p11d.htm`:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<P>Corect!</P>
<P>Ati selectat: &quot;3. #00FFFF este albastru.&quot;
</P>
</BODY>
</HTML>
```

Exemplul doreşte să arate modul în care se poate realiza un document cu hiperlegături către alte documente şi modul din care la selectarea unei hiperlegărturi se poate reveni la pagina de la care s-a plecat. În **Figura 15** se prezintă modul în care IE afişează pagina principală de pe care se poate sări la alte pagini, iar în **Figura 16** se prezintă modul de revenire la pagina principală dintr-

o pagină la care s-a sărit.

Figure 15
Exemplu cu hiperlegături

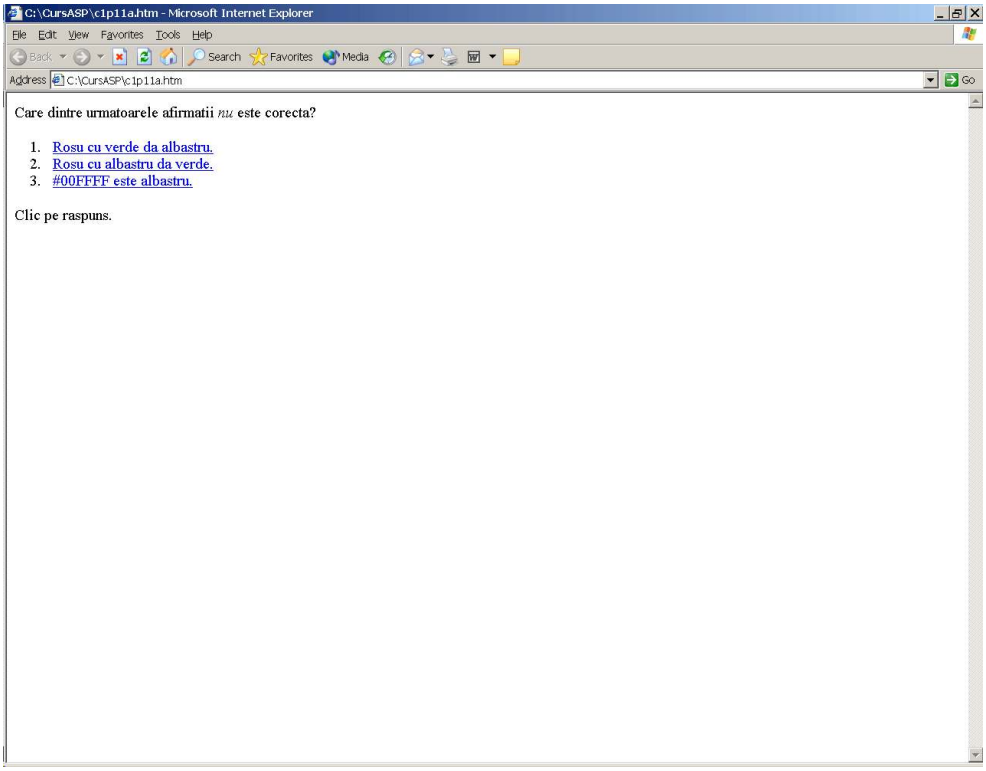
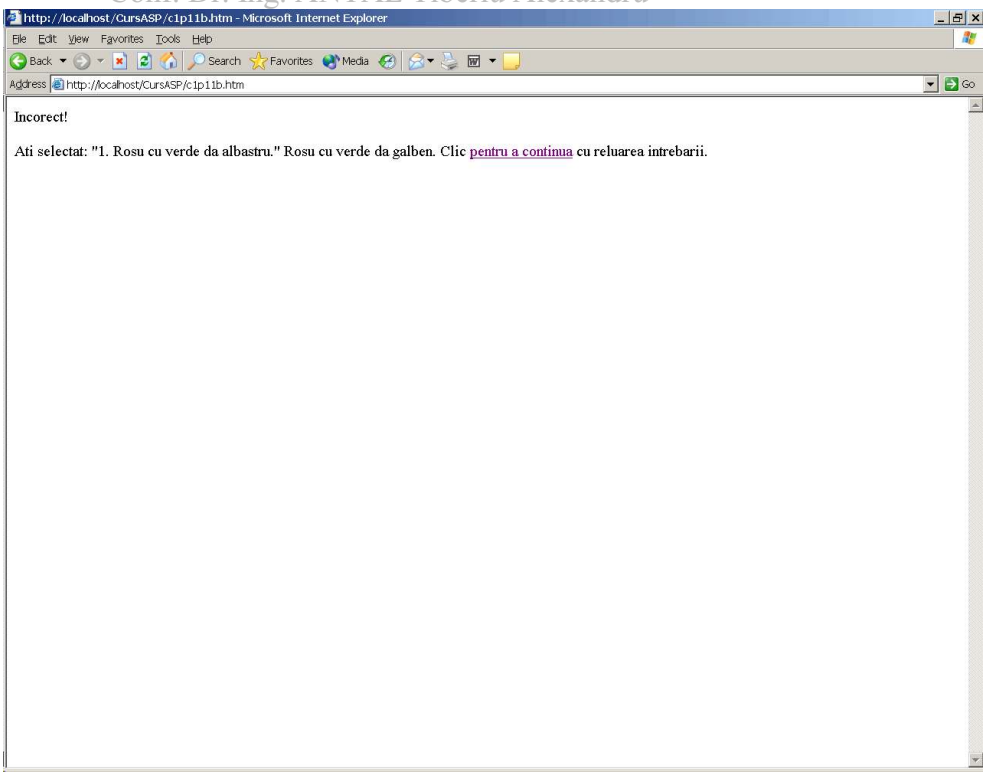


Figure 16
Revenirea la pagina inițială a întrebărilor



Ancorarea la imagini

Legăturile nu sunt limitate numai la porțiuni de texte. O imagine poate fi folosită pe post de sursă a legăturii. Paginile arată ceva mai bine dacă elemente grafice sunt folosite pentru a forma

legăturile. Pentru a crea o hiperlegătură folosind o imagine, marcajul de imagine trebuie pus în interiorul marcajului de ancorare. Iată un exemplu:

```
<A HREF="http://server/site/pagina.htm"
      <IMG SRC="http://server/site/imagine.gif" BORDER=0>>
</A>
```

Implicit, navigatorul afişează imaginea cu chenar. Culoarea chenarului se moşteneşte de la culoarea legăturilor către pagini. Dacă nu doriți afişarea chenarului proprietatea `BORDER` trebuie setată la valoarea `0`.

Tabele

Tabelele permit organizarea articolelor în linii şi coloane. Deoarece navigatorul ignoră spațiile albe, una dintre aplicațiile tabelelor este afişarea articolelor care trebuie separate prin spații albe. Navigatoarele moderne (peste versiunea 4) permit poziționarea absolută de articole la nivel de pixel. Totuşi, chiar şi în acest caz, tabelele sunt utilizate pentru afişarea datelor pe coloane.

Marcajul `TABLE` conține alte marcaje pentru definirea coloanelor şi rândurilor. Tabelele pot avea trei secțiuni: antetul (`header`), corpul (`body`) şi sfârşitul (`footer`). Rândurile de antet şi de sfârşit sunt fixe, nu se poate face defilare la nivelul lor şi sunt opționale. Dacă se include un antet sau un sfârşit atunci acesta trebuie să aibă şi o secțiune de corp. Tabelele pot avea şi marcajul `CAPTION`, pentru a da explicații cu privire la conținutul lor. Navigatorul va afişa acest text formatat deasupra antetului. Setările de chenar nu se aplică şi explicațiilor. În plus, IE poate conține marcajele `COLGROUP` şi `COL` care pot simplifica formatarea tabelelor. Marcajul `COLGROUP` defineşte o mulțime de coloane, iar `COL` defineşte coloane individuale la nivelul grupului de coloane.

Marcajele <TABLE>, <TR> şi <TD>

Tabelele încep cu marcajul `<TABLE>`. Rândurile sunt delimitate prin marcajul `<TR>` (`Table Row`), iar coloanele prin marcajul `<TD>` (`Table Data`). Închiderea marcajelor folosite pentru definirea unui tabel este obligatorie dacă motorul de afişare nu este IE. Numai în acest caz marcajele de închidere sunt opționale. Cel mai simplu tabel HTML (un rând, o coloană) este:


```
<HTML>
<HEAD>
<TITLE>Introducere in Tabele</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
      <TABLE>
            <TR>
                  <TD>Aha</TD>
            </TR>
      </TABLE>
</BODY>
```

IE va afişa acest tabel aşa cum se vede în **Figura 17.** Observați că, nu prea seamănă a tabel. Din fericire există un număr de atribute care pot controla poziția articolelor şi modul de afişare. Câteva dintre atributele care ajută la înfrumusețarea tabelelor sunt:

Atribut	Descriere
`ALIGN`	Atributul de aliniere poate lua una dintre valorile `LEFT`, `RIGHT` sau `CENTER`. Controlează poziția pe orizontală a tabelului în pagină.
`BACKGROUND`	Atributul, asemenea lui `BACKGROUND` din `BODY`, permite inserarea unui URL cu imagine. Navigatorul afişează imaginea, pe axa z, la nivelul 0, iar toate datele vin afişate peste imagine.
`BGCOLOR`	Este o valoare de culoare HTML pentru fondul tabelului. Valoarea poate fi suprascrisă la nivelul fiecărui rând sau al celulelor individuale.
`BORDER`	O valoare întreagă care controlează grosimea chenarului desenat în jurul tabelului şi al celulelor individuale. Valoarea implicită este 0, adică linia de separare nu se afişează.

Atribut	Descriere
CELLPADDING	O valoare întreagă care controlează spațiul dintre conținutul şi chenarul celulei.
CELLSPACING	O valoare întreagă care controlează spațiul dintre celule.
COLS	O valoare întreagă care specifică numărul de coloane ale tabelului. Atunci când se foloseşte, duce la creşterea vitezei de afişare a tabelului. În lipsa lui navigatorul trebuie să parcurgă întregul tabel înainte de a-l afişa; în prezența lui afişarea rândurilor poate începe imediat.
HEIGTH	Valoare întreagă care informează navigatorul cu privire la înălțimea necesară pentru afişarea tabelului. Asemenea lui COLS, prin prezența lui creşte viteza de afişare a tabelului. Poate fi specificat în pixeli sau în procente din înălțimea zonei client vizibilă.
WIDTH	Valoare întreagă care informează navigatorul cu privire la lățimea necesară pentru afişarea tabelului. Asemenea lui COLS, prin prezența lui creşte viteza de afişare a tabelului. Poate fi specificat în pixeli sau în procente din lățimea zonei client vizibilă.
VALIGN	Realizează alinierea pe verticală a datelor într-o celulă. Valori posibile sunt: TOP, MIDDLE, BOTTOM, BASELINE.

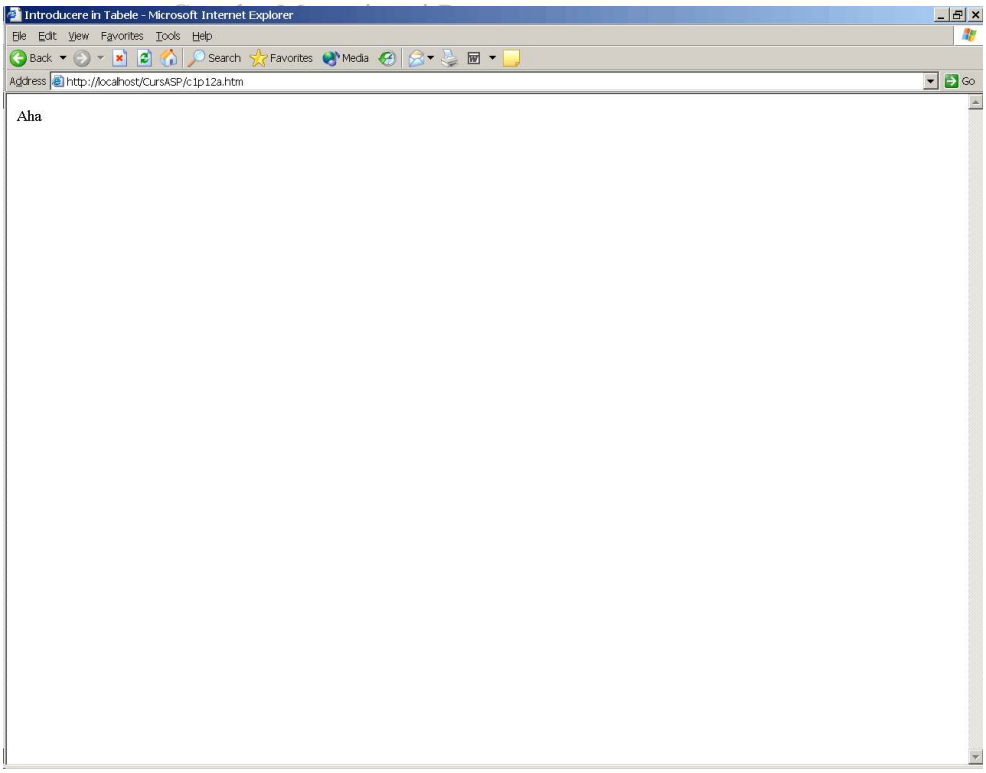
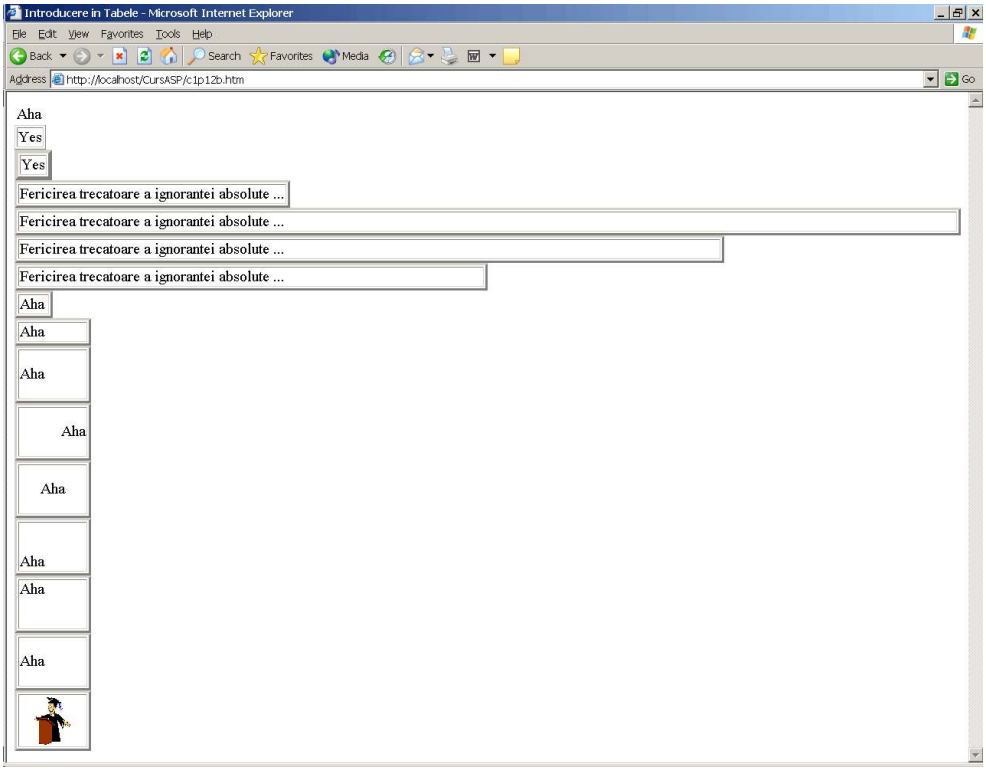


Figure 17
Cel mai simplu tabel HTML

În **Figura 18** se prezintă modul în care pot fi utilizate atributele pentru realizarea unei afişări mai spectaculoase. Codul HTML este prezentat în continuare. Se observă efectele modificării grosimii chenarului, a specificării dimensiunilor tabelului, a poziționării pe orizontală şi pe verticală asupra textelor de afişat în celulele tabelului. În final, se arată modul de afişare a unei imagini

într-o celulă de tabel.

Figure 18
Efecte de afișare prin folosirea atributelor



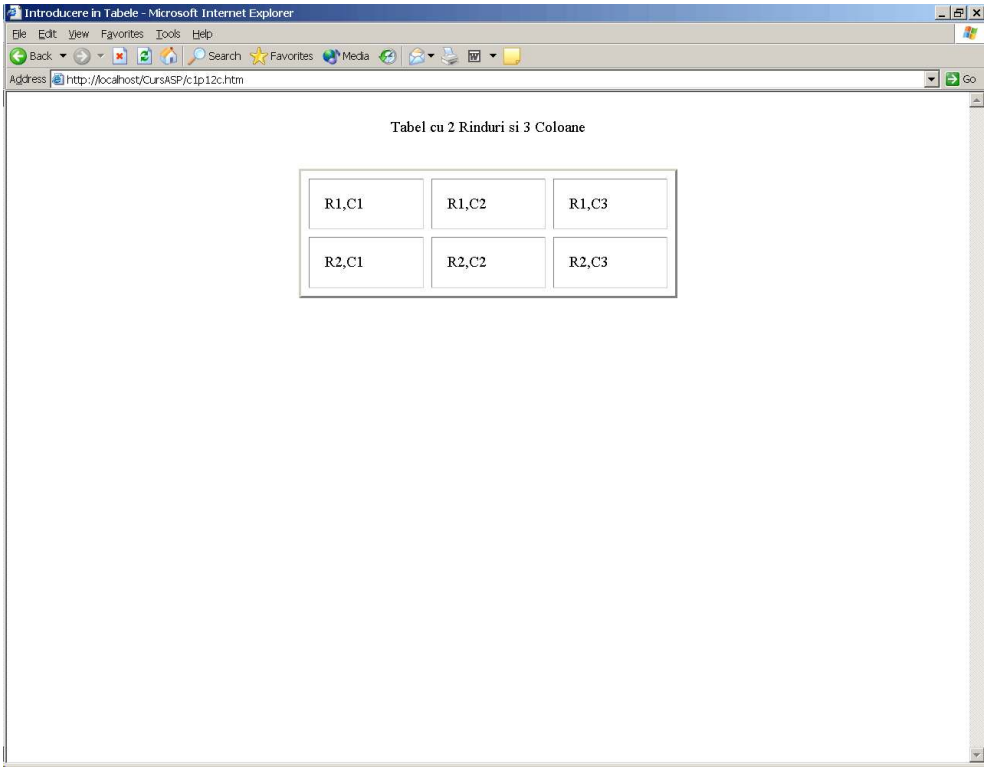

```
<HTML>
<HEAD>
<TITLE>Introducere in Tabele</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
      <TABLE>
            <TR>
                  <TD>Aha</TD>
            </TR>
      </TABLE>

      <TABLE BORDER=1>
            <TR>
                  <TD>Yes</TD>
            </TR>
      </TABLE>

      <TABLE BORDER=5>
            <TR>
                  <TD>Yes</TD>
            </TR>
      </TABLE>

      <TABLE BORDER=3>
            <TR>
                  <TD>Fericirea trecatoare a ignorantei absolute ...</TD>
            </TR>
      </TABLE>

      <TABLE BORDER=3 WIDTH=100%>
            <TR>
                  <TD>Fericirea trecatoare a ignorantei absolute ...</TD>
            </TR>
```

```
</TABLE>

<TABLE BORDER=3 WIDTH=75%>
      <TR>
            <TD>Fericirea trecatoare a ignorantei absolute ...</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=50%>
      <TR>
            <TD>Fericirea trecatoare a ignorantei absolute ...</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=50>
      <TR>
            <TD>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100>
      <TR>
            <TD>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=RIGHT>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=CENTER>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=LEFT VALIGN=BOTTOM>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=LEFT VALIGN=TOP>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=LEFT VALIGN=CENTER>Aha</TD>
      </TR>
</TABLE>

<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
      <TR>
            <TD ALIGN=CENTER VALIGN=MIDDLE><IMG SRC="./imagini/grad.gif"
WIDTH=50 HEIGHT=65></TD>
```

```
            </TR>
      </TABLE>

</BODY>
```

Exemplele prezentate până acum au fost didactice. În continuare se prezintă câteva variante de tabele care au o legătură mai apropiată cu practica. Codul HTML pentru **Figura 19** se prezintă în continuare.

Figure 19 Tabel cu 2 rânduri şi 3 coloane

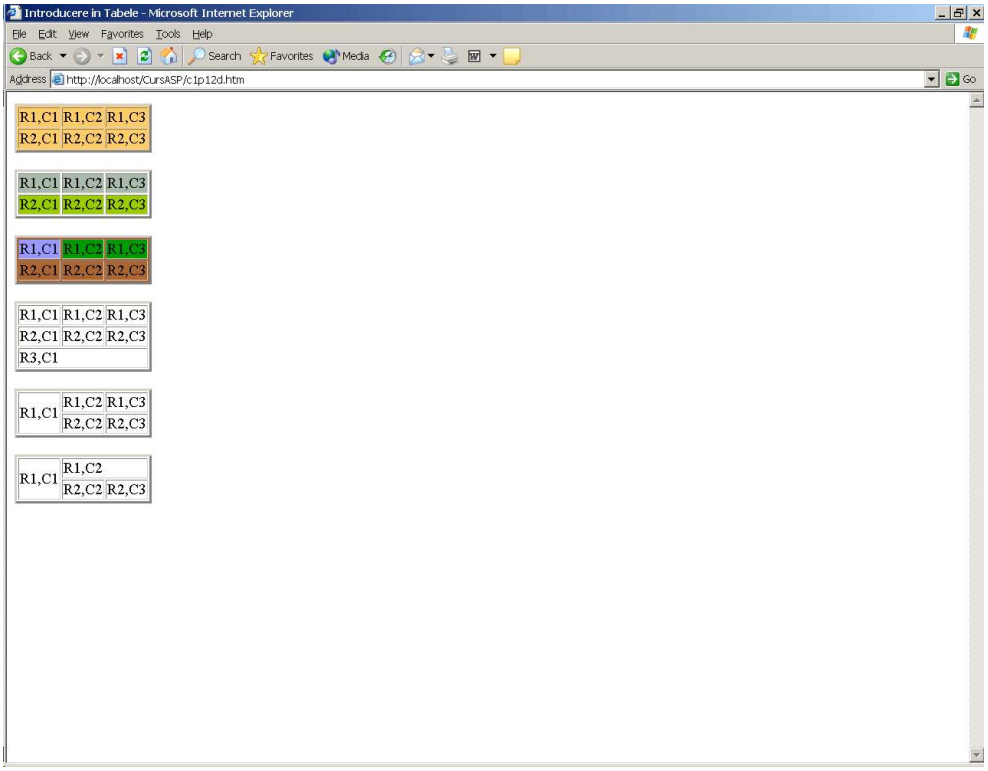



```
<HTML>
<HEAD>
<TITLE>Introducere in Tabele</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
      <TABLE BORDER="3" ALIGN="CENTER" CELLPADDING="20" CELLSPACING="10"
WIDTH="40%" HEIGHT="20%">
      <CAPTION>Tabel cu 2 Rinduri si 3 Coloane</CAPTION>
            <TR>
                  <TD>R1,C1</TD>
                  <TD>R1,C2</TD>
                  <TD>R1,C3</TD>
            </TR>
            <TR>
                  <TD>R2,C1</TD>
                  <TD>R2,C2</TD>
                  <TD>R2,C3</TD>
            </TR>
      </TABLE>
</BODY>
```

Codul care urmează, are efectul prezentat în **Figura 20**. Marcajul `TD` poate conține două atribute adiționale, `COLSPAN` şi `ROWSPAN`. Valorile luate de aceste atribute sunt numere întregi. Ele specifică numărul de coloane respectiv rânduri peste care se extinde o celulă. Se folosesc atunci

când este nevoie ca tabelul să aibă rânduri cu număr diferit de coloane.

Figure 20
Câteva variante de tabele




```
<HTML>
<HEAD>
<TITLE>Introducere in Tabele</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF TEXT=#000000 LINK=#0000FF VLINK=#800080>
      <TABLE BORDER="3" BGCOLOR="#FFCC66">
            <TR>
                  <TD>R1,C1</TD>
                  <TD>R1,C2</TD>
                  <TD>R1,C3</TD>
            </TR>
            <TR>
                  <TD>R2,C1</TD>
                  <TD>R2,C2</TD>
                  <TD>R2,C3</TD>
            </TR>
      </TABLE>
      <BR>
      <TABLE BORDER="3">
            <TR BGCOLOR="#AABBAA">
                  <TD>R1,C1</TD>
                  <TD>R1,C2</TD>
                  <TD>R1,C3</TD>
            </TR>
            <TR BGCOLOR="#99CC00">
                  <TD>R2,C1</TD>
                  <TD>R2,C2</TD>
                  <TD>R2,C3</TD>
            </TR>
      </TABLE>
      <BR>
      <TABLE BORDER=3 BGCOLOR="#AA6633">
            <TR BGCOLOR="#009900">
                  <TD BGCOLOR="#9999FF">R1,C1</TD>
                  <TD>R1,C2</TD>
```

```
                <TD>R1,C3</TD>
            </TR>
            <TR>
                <TD>R2,C1</TD>
                <TD>R2,C2</TD>
                <TD>R2,C3</TD>
            </TR>
        </TABLE>
        <BR>
        <TABLE BORDER="3">
            <TR>
                <TD>R1,C1</TD>
                <TD>R1,C2</TD>
                <TD>R1,C3</TD>
            </TR>
            <TR>
                <TD>R2,C1</TD>
                <TD>R2,C2</TD>
                <TD>R2,C3</TD>
            </TR>
            <TR>
                <TD COLSPAN=3>R3,C1</TD>
            </TR>
        </TABLE>
        <BR>

        <TABLE BORDER=3>
            <TR>
                <TD ROWSPAN=2>R1,C1</TD>
                <TD>R1,C2</TD>
                <TD>R1,C3</TD>
            </TR>
            <TR>
                <TD>R2,C2</TD>
                <TD>R2,C3</TD>
            </TR>
        </TABLE>
        <BR>

        <TABLE BORDER=3>
            <TR>
                <TD ROWSPAN=2>R1,C1</TD>
                <TD COLSPAN=2>R1,C2</TD>
            </TR>
            <TR>
                <TD>R2,C2</TD>
                <TD>R2,C3</TD>
            </TR>
        </TABLE>
</BODY>
```

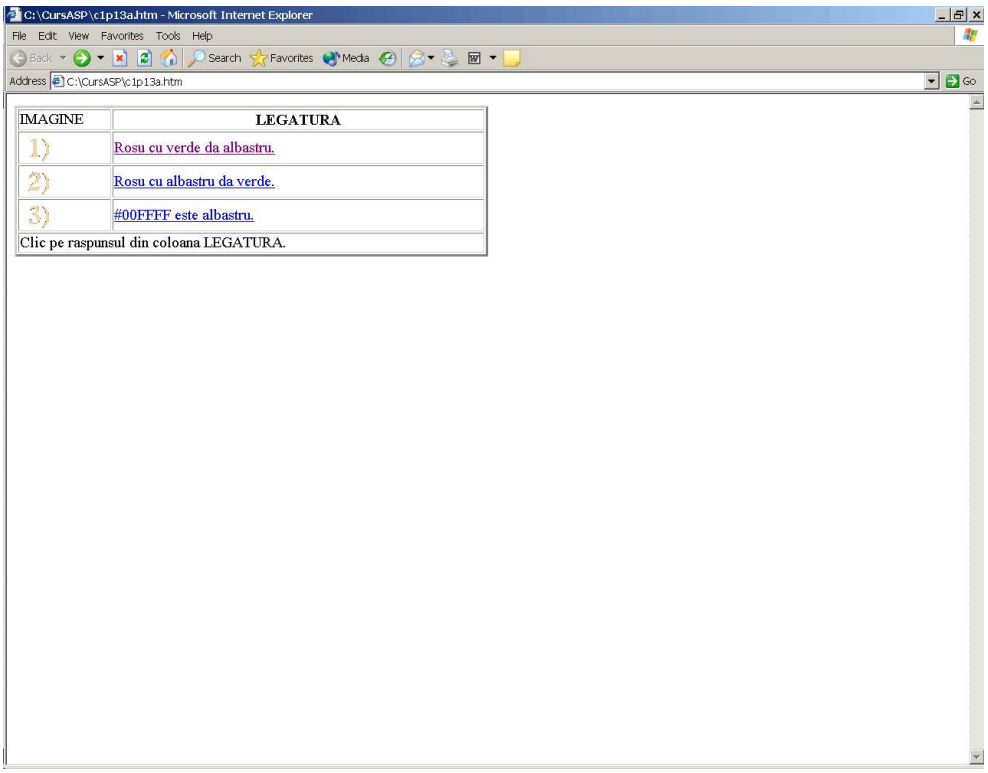

Marcajele <THEAD> şi <TFOOT>

Am spus deja ca tabelul este format din trei secţiuni distincte. În **Figura 21** se prezintă un exemplu de lucru cu `THEAD` şi `TFOOT`. Scopul primar al folosirii unui antet (`THEAD`) constă în duplicarea antetului la tipărirea tabelului în cazul în care tabelul se întinde pe mai multe pagini. Fiecare tabel poate avea un singur antet şi un singur sfârşit. Acestea pot fi formate însă din mai multe rânduri. Celulele antetului pot fi formatate automat dacă în locul lui `TD` se foloseşte `TH`.

Acelaşi exemplu prezintă şi o altă aplicaţie practică a tabelelor. Ele pot fi folosite pentru a realiza aranjarea imaginilor şi textelor asociate lor independent de rezoluţia la care se vizualizează pagina. Fiecare realizator de pagini are obiceiul de a lucra cu ecranul într-o anumită rezoluţie. Din acest motiv pagina arată bine dacă este vizualizată la rezoluţia în care a fost creată. Dacă pagina este vizualizată într-o altă rezoluţie este posibil ca între anumite elemente să rămână spaţii prea

mari sau să nu mai fie spațiu suficient. În cazul tabelelor, în funcție de rezoluția la care se vizualizează pagina, VGA(640x480), SuperVGA (800x600), XGA (1280x768), un atribut de tipul `WIDTH=50%` îi spune navigatorului să folosească 50% din lățimea zonei de afişare a clientului pentru tabel. Se poate asigura astfel o independență de rezoluția de afişare, dar efectul este acela că textul tabelului va fi aranjat diferit, în funcție de rezoluția de vizualizare. O metodă prin care se poate păstra proporția între celulele tabelului, pentru a putea depăşi acest dezavantaj, se prezintă în exemplul care urmează. Dacă tabelul ocupă 50% din spațiul de afişate, prima coloană va ocupa 10% din acest spațiu, iar coloana următoare 40%.

Figure 21
Tabel cu antet şI sârşit




```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>

<TABLE BORDER=3 WIDTH=50%>
      <THEADH>
            <TR>
                  <TD>IMAGINE</TD>
                  <TH>LEGATURA</TH>
            </TR>
      </THEAD>
      <TR>
            <TD     WIDTH=10%><IMG     SRC="./imagini/unu.gif"     WIDTH=50
HEIGHT=38></TD>
            <TD WIDTH=40%><a href="c1p11b.htm">Rosu cu verde da albastru.</a>
      </TR>
      <TR>
            <TD     WIDTH=10%><IMG     SRC="./imagini/doi.gif"     WIDTH=50
HEIGHT=38></TD>
            <TD WIDTH=40%><a href="c1p11c.htm">Rosu cu albastru da verde.</a>
      </TR>
      <TR>
            <TD     WIDTH=10%><IMG     SRC="./imagini/trei.gif"     WIDTH=50
```

```
HEIGHT=38></TD>
            <TD WIDTH=40%><a href="c1p11d.htm">#00FFFF este albastru.</a>
      </TR>

      <TFOOT>
            <TR>
                  <TD COLSPAN="2">Clic pe raspunsul din coloana LEGATURA.</TD>
            </TR>
      </TFOOT>

</TABLE>
</BODY>
</HTML>
```

Cadre (FRAMEs)

Pentru înțelegerea cadrelor trebuie să revenim la începuturile Windows-ului. O fereastră este o regiune de memorie în care se stochează o zonă rectangulară de pixeli. Fiecare program are mai multe ferestre, fiecare fereastră fiind una principală sau una copil. Toate ferestrele sunt copiii ferestrei principale din Windows, adică a desktop-ului. Ordinea pe axa z (`z-order`) controlează poziția ferestrelor în stiva de ferestre. Cea care este deasupra are numărul de ordine 0. Toate celelalte ferestrele se afişează în fereastra activă şi au numărul de ordine pe axa z mai mare. Fereastra principală a unei aplicații are numeroase ferestre copil. De exemplu, fiecare fereastra de tipul toolbar este un copil al ferestrei principale. Ferestrele copil, deşi sunt ferestre asemenea celor principale, apar şi dispar înpreună cu cea principală. De exemplu, dacă se minimizează fereastra principală, toate ferestrele copil dispar împreună cu aceasta.

La nivelul navigatorului definirea unei ferestre copil se face prin conceptul de cadru (`frame`). Cadrele sunt părți ale ferestrei principale, dar fiecare cadru este tratat de navigator independent de celelelte. Astfel, navigarea în acestea se desfăşoară independent. Din punctul de vedere al navigatorului fiecare cadru este tratat asemenea feresterei principale, cu excepția faptului că acestea sunt dependente de fereastra principală. Deci, fiecare cadru este un document HTML obişnuit. Cadrele în sine ne permit să divizăm fereastra navigatorului în ferestre mai mici, de dimensiunea dorită. Cadrele pot exista numai în cadrul conceptului de mulțime de cadre (`frameset`). Mulțimea de cadre nu este vizibilă în fereastra navigatorului, ea conține cadre şi se defineşte cu marcajul `FRAMESET`, iar un cadru cu marcajul `FRAME`.

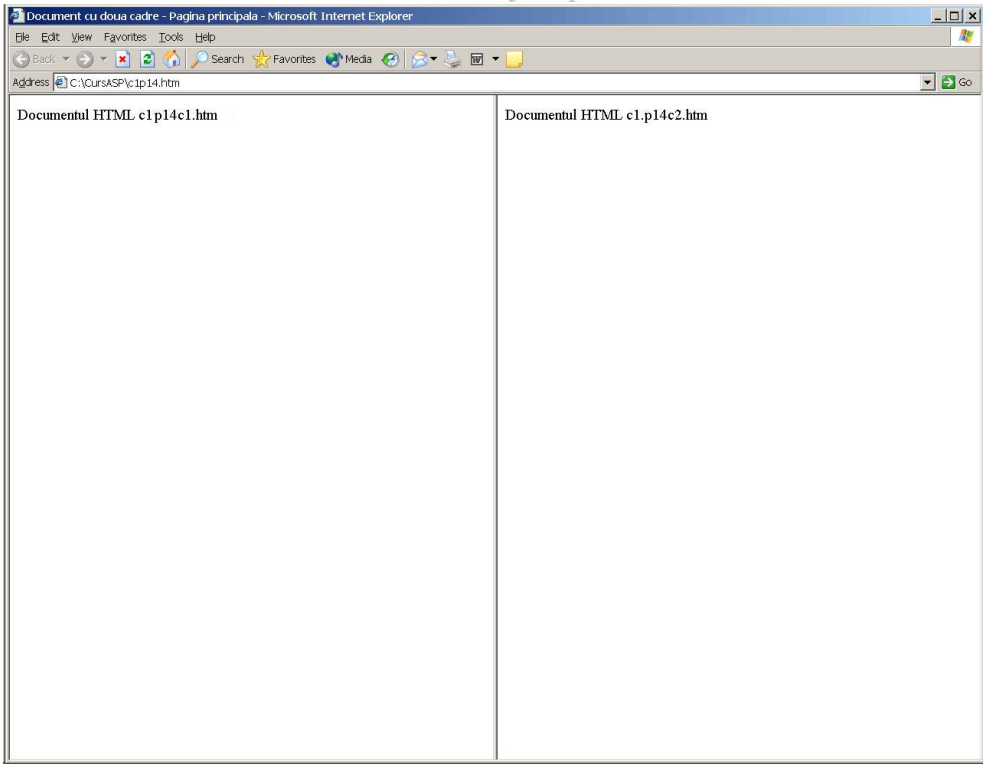


Figure 22 Fereastră principală cu două cadre

De exemplu, pentru a crea o mulțime de cadre ce divizează fereastra client în două porțiuni egale se scrie codul HTML următor. Aici, la început, se creează două documente HTML distincte, `c1p14c1.htm` şi `c1p14c2.htm`, stocate în directorul `c1p14c`, apoi fişierul `c1p14.htm` în directorul părinte a lui *c1p14c*.

Fişierul `c1p14c1.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p14c1.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p14c1.htm
</BODY>
```

Fişierul `c1p14c2.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p14c2.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p14c2.htm
</BODY>
```

Fişierul `c1p14.htm`:

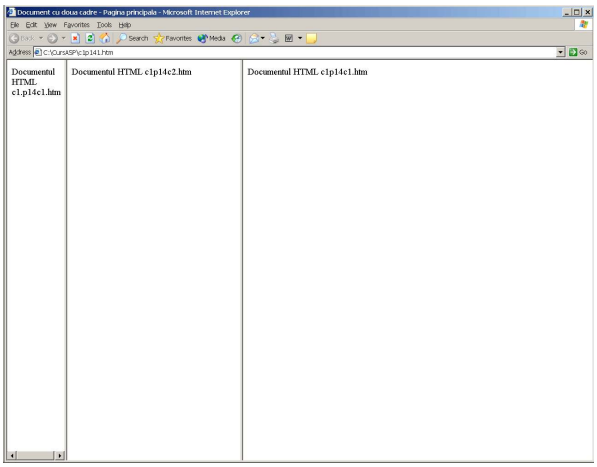
```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=50%,50%>
      <FRAME SRC="./c1p14c/c1p14c1.htm">
      <FRAME SCR="./c1p14c/c1p14c2.htm">
</FRAMESET>
</HTML>
```

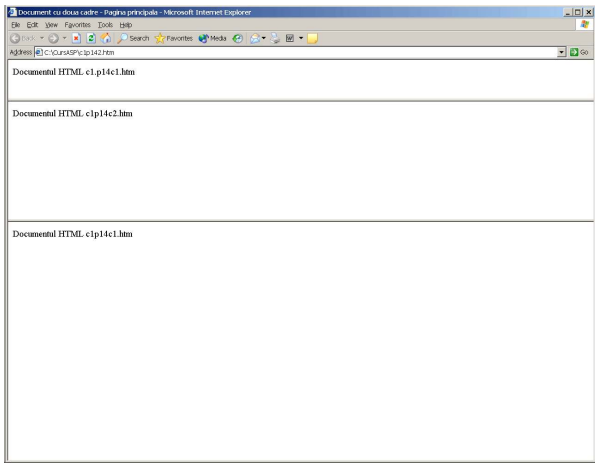

În **Figura 22** se prezintă modul de afişare a celor două cadre în fereastra principală.

Cadrele pot avea dimensiuni diferite. În exemplul care urmează aceste dimensiuni sunt stabilite în procente din fereastra principală. Observați că mai este nevoie de încă un document HTML care va fi stocat în fişierul `c1p14c3.htm`.

```
<HTML>
<HEAD>
<TITLE>Document cu trei cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=10%,30%,60%>
      <FRAME src="./c1p14c/c1p14c1.htm">
      <FRAME src="./c1p14c/c1p14c2.htm">
      <FRAME src="./c1p14c/c1p14c3.htm">
</FRAMESET>
</HTML>
```

Specificarea divizării pe linii sau pe coloane a ferestrei se face prin atributele `COLS` şi `ROWS`. Codul HTML ce urmează este asemenea celui anterior, doar că face divizarea pe rânduri a feresteri principale. În figurile care urmează se observă diferențele între cele două metode de divizare.






```
<HTML>
<HEAD>
<TITLE>Document cu trei cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET ROWS=10%,30%,60%>
      <FRAME SRC="./c1p14c/c1p14c1.htm">
      <FRAME SRC="./c1p14c/c1p14c2.htm">
      <FRAME SRC="./c1p14c/c1p14c3.htm">
</FRAMESET>
</HTML>
```

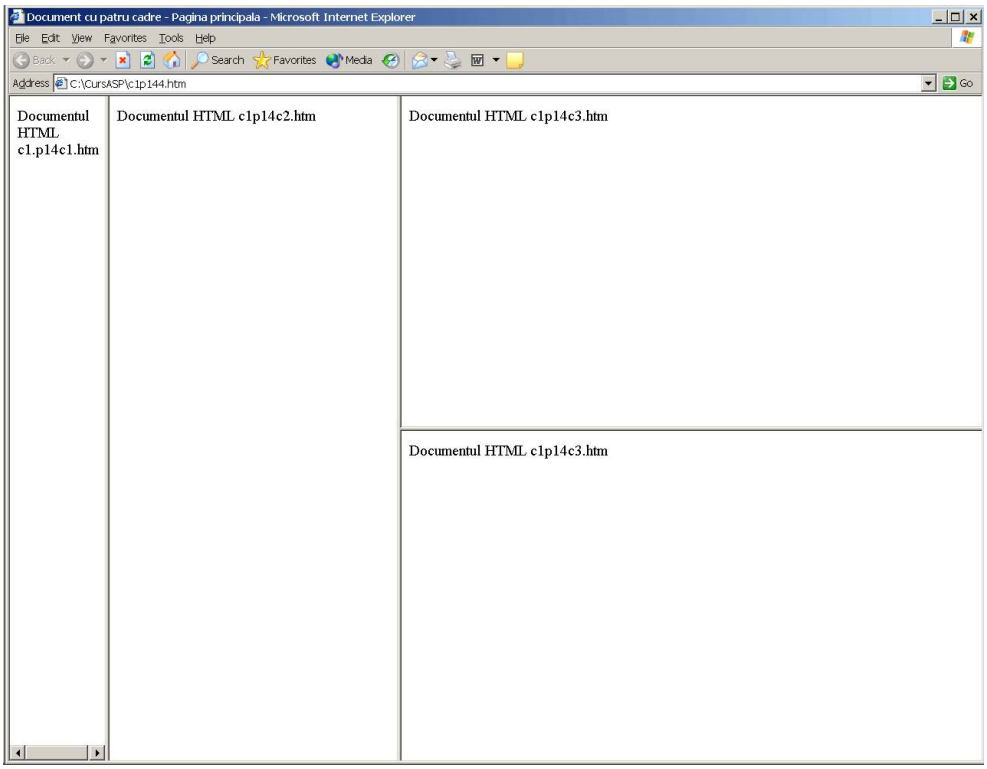

Lungimile se pot specifica atât în procente cât şi în pixeli. În exemplul care urmează `50` are semnificaţia de `50` de pixeli, iar `*` de ceea ce rămîne.

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=50,*>
      <FRAME SRC="./c1p14c/c1p14c1.htm">
      <FRAME SRC="./c1p14c/c1p14c2.htm">
</FRAMESET>
</HTML>
```

La utilizarea lui `FRAMESET` prin `*` se poate obţine un cadru de dimensiuni elastice. Importanţa lor este mare atunci când se încearcă realizarea unor pagini care funcţionează corect la rezoluţii diferite. Atunci când se folosesc procente pentru specificarea de dimensiuni 10% se calculează din dimensiunea pe orizontală a ecranului, dacă cineva în loc de rezoluţia 800x600 de pixeli o foloseşte pe cea 640x400, ecranul va avea o altă lăţime, deci şi lungimile calculate prin procente vor fi diferite. În acest caz cadrele vor arăta altfel. Pentru a rezolva aceasta problemă se poate face cadrul din stânga de dimensiune fixă, prin specificarea lungimii lui în pixeli, iar cel din dreapta elastic (prin `<FRAMESET COLS=50,*>`). În cazul în care avem mai multe coloane se pot folosi relaţii aritmetice pentru specificarea mărimii cadrelor. Iată un exemplu: `<FRAMESET COLS=50,*,2*>`, aici prima coloană va avea lungimea de 50 de pixeli. Ceea ce rămâne se împarte între cadrul al doilea şi al treilea, dar cadrul al treilea va fi de două ori mai mare decât al doilea.

Dacă dorim ca un cadru să fie divizat, în două, pe orizontală folosiţi codul următor. Efectele lui se prezintă în **Figura 23**.

Figure 23
Document cu patru cadre

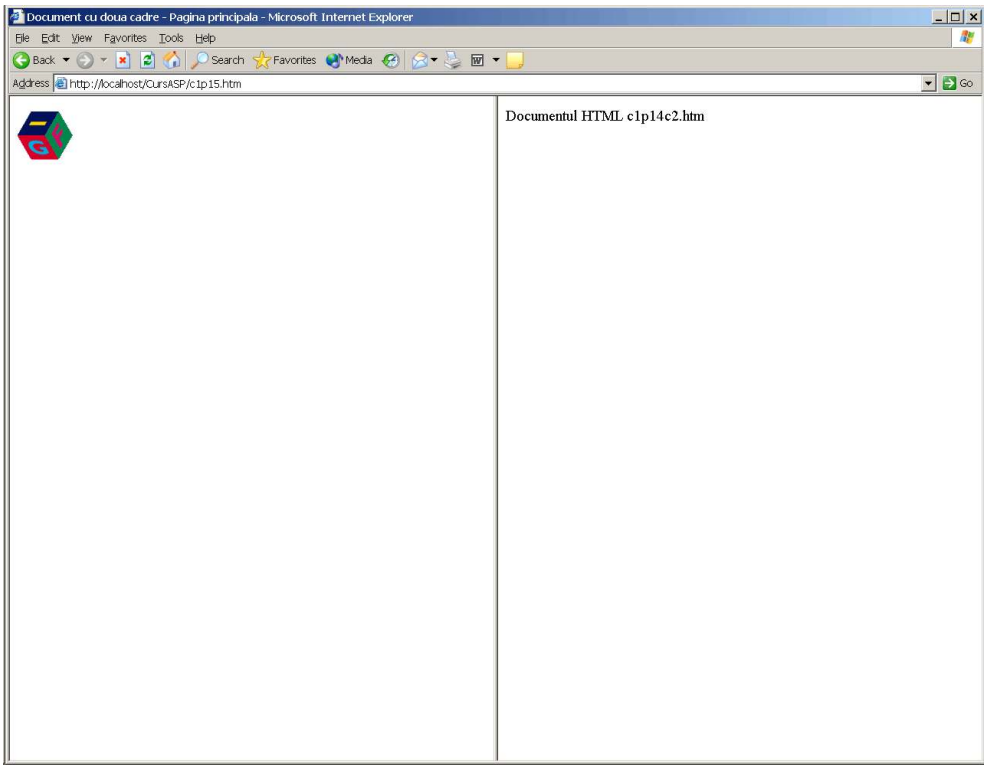



```
<HTML>
<HEAD>
<TITLE>Document cu patru cadre - Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=10%,30%,60%>
      <FRAME SRC="./c1p14c/c1p14c1.htm">
      <FRAME SRC="./c1p14c/c1p14c2.htm">
      <FRAMESET ROWS= 50%,50%>
            <FRAME SRC="./c1p14c/c1p14c3.htm">
            <FRAME SRC="./c1p14c/c1p14c3.htm">
      </FRAMESET>
</FRAMESET>
</HTML>
```

Cadrele se pot folosi şi pentru afişarea de imagini după cum se observă în **Figura 24**. Codul HTML este următorul:

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=50%,50%>
      <FRAME src="./imagini/cub_GIF.gif" WIDHT=90 HEIGHT=100>
      <FRAME src="./c1p14c/c1p14c2.htm">
</FRAMESET>
</HTML>
```

Figure 24
Afişarea unei imagini



Pentru a reduce primul cadru la dimensiunea imaginii scriem:

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=90,*>
      <FRAME src="./imagini/cub_GIF.gif" WIDTH=90 HEIGHT=100>
      <FRAME src="./c1p14c/c1p14c2.htm">
</FRAMESET>
</HTML>
```

Vom trece mai departe şi vom diviza primul cadru în alte două, primul având înălțimea imaginii.

```
<HTML>
<HEAD>
<TITLE>Document cu trei cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=90,*>
      <FRAMESET ROWS=100,*>
            <FRAME src="./imagini/cub_GIF.gif" WIDTH="90" HEIGHT="100">
            <FRAME src="./c1p14c/c1p14c2.htm">
      </FRAMESET>
      <FRAME src="./c1p14c/c1p14c3.htm">
</FRAMESET>
</HTML>
```

Observați ce se petrece dacă în loc de `<FRAME SRC="./imagini/cub_GIF.gif" WIDTH="90" HEIGHT="100">` se va scrie `<FRAME SRC="./imagini/cub_GIF.gif" WIDTH="90" HEIGHT="100" SCROLLING=NO>`. Dacă în acest moment imaginea nu se vede toată va trebuie să manipulăm marginile cadrului după cum urmează: `<FRAME SRC="./imagini/cub_GIF.gif" WIDTH="90" HEIGHT="100" SCROLLING=NO MARGINWIDTH=1 MARGINHEIGHT=1>`.

Marginile cadrelor pot fi îngroşate sau colorate prin folosirea atributelor `BORDER` şi `BORDERCOLOR`.

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET COLS=90,* BORDER=20 BORDERCOLOR="#FFOOOO">
      <FRAMESET ROWS=100,*>
            <FRAME src="./imagini/cub_GIF.gif" WIDTH="90" HEIGHT="100"
            SCROLLING=NO MARGINWIDTH=1 MARGINHEIGHT=1>
            <FRAME src="./c1p14c/c1p14c2.htm">
      </FRAMESET>
      <FRAME src="./c1p14c/c1p14c3.htm">
</FRAMESET>
</HTML>
```

De asemenea, dacă linia corespunzătoare de mai sus se înlocuieşte cu `<FRAMESET ROWS=100,* FRAMEBORDER=NO>` atunci marginea nu va fi afişată.

Uneori dorim ca navigatorul să nu realizeze modificarea dimensiunilor cadrelor. Pentru aceasta în locul liniei corespunzătoare de mai sus se scrie `<FRAME src="./c1p14c/c1p14c3.htm" NORESIZE>`.

Una dintre aplicațiile cadrelor se prezintă în exemplul următor. În directorul `c1p16c` sunt stocate trei documente HTML. Numele fişierele şi codul corespunzător lor este:

Fişierul `c1p16c1.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p16c1.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p16c1.htm
      <P>
      O legatura catre documentul <A HREF="c1p16c3.htm">c1p16c3.htm</A>
      </P
</BODY>
```

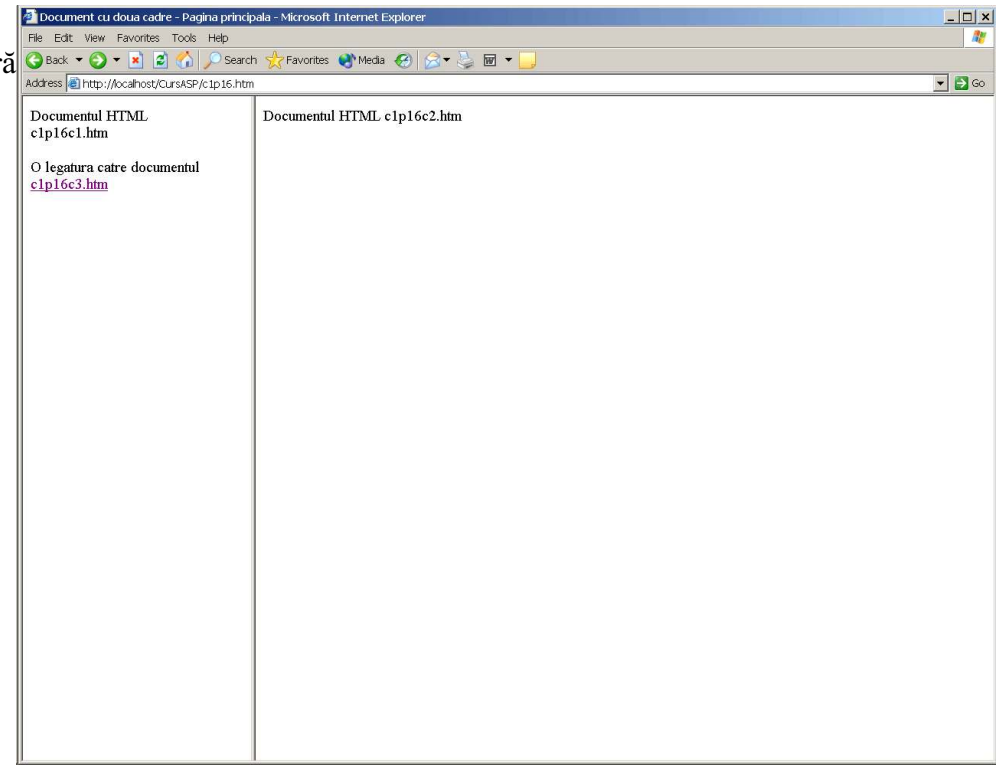
Fişierul `c1p16c2.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p16c2.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p16c2.htm
</BODY>
```

Fişierul `c1p16c3.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p16c3.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p16c3.htm
</BODY>
```

Figure 25
Cadru cu legătură



În directorul părinte se află fişierul `c1p16.htm` ce conţine codul HTML:

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET cols=300,*>
      <FRAME src="./c1p16c/c1p16c1.htm">
      <FRAME src="./c1p16c/c1p16c2.htm">
</FRAMESET>
</HTML>
```

Navigatorul va afişa documentul `c1p16c1.htm` după cum se vede în **Figura 25**. Când se face clic pe legătura (vezi **Figura 25**) din prima coloană în locul ei se va afişa conţinutul documentului către care se face legătura. Sigur, aceasta nu este o soluţie elegantă, ar fi fost de aşteptat ca acest document să fie afişat în locul celui din coloana din drepta. Pentru aceasta trebuie făcute următoarele modificări:

În fişierul `c1p16.htm`:

```
<HTML>
<HEAD>
<TITLE>Document cu doua cadre -  Pagina principala</TITLE>
</HEAD>
<FRAMESET cols=300,*>
      <FRAME src="./c1p16c/c1p16c1.htm">
      <FRAME src="./c1p16c/c1p16c2.htm" NAME="FER-1">
</FRAMESET>
</HTML>
```

În fişierul `c1p16c1.htm`:

```
<HTML>
<HEAD>
<TITLE>Cadre, c1p16c1.htm</TITLE>
</HEAD>
<BODY>
      Documentul HTML c1p16c1.htm
      <P>
      O legatura catre documentul <A HREF="c1p16c3.htm" TARGET="FER-1">
c1p16c3.htm</A>
      </P>
</BODY>
```

Ca urmare a modificărilor, pagina la care se face legătura se va încărca în cadrul cu numele `FER-1`.

Avantajele cadrelor

Marele avantaj al cadrelor este acela că permit separarea a vizuală documentelor şi afişarea lor independentă. Se poate defini o legătură într-un cadru care va afişa sau va genera o altă acțiune, într-un alt cadru, fără a redesena întregul ecran.

Un alt avantaj al cadrelor este acela că ele se pot redimensiona. Utilizatorul poate trage de marginile cadrelor pentru a mări sau a micşora zona de ecran acoperită de cadru.

Dezavantajele cadrelor

Marele dezavantaj al cadrelor este crearea şi controlul mai complicat. Pentru afişarea unei pagini cu două cadre trebuie create trei fişiere. Unul va conține definițiile mulțimii de cadre, apoi câte un fişier pentru fiecare cadru.

Un alt dezavantaj ar fi timpul de afişare care creşte odată cu numărul cadrelor. Navigatorul adresează o cerere server-ului pentru pagina principală, apoi câte o cerere pentru fiecare cadru. Din acest punct de vedere afişarea unei pagini cu două cadre este de trei ori mai lungă decât a unei pagini fără cadre.

Deseori, cadrele, necesită cod scris în **VBScript**, dacă lucrați cu IE, sau în **JavaScript**, dacă lucrați cu Netscape. De exemplu, dacă la un clic de legătură doriți să schimbați conținutul a două cadre trebuie să scrieți codul aferent. De asemenea, generarea unor mesaje de eroare corespunzătoare cadrelor necesită scrierea de cod.

O altă problemă este depanarea documentelor care au cadre, aceasta fiind mult mai complicată decât cea a documentelor fără cadre.

Modalităţi de evitare a cadrelor

Pentru evitarea cadrelor informația care rămâne neschimbată trebuie repetată. Majoritatea server-elor au un mecanism pentru aceasta, de exemplu în IIS, instrucțiunea `include` permite includerea unor fişiere într-un document. Server-ul va înlocui această instrucțiune cu conținutul fişierului de inclus. De exemplu, dacă se doreşte afişarea unei adrese pe fiecare pagină distinctă din document, aceasta poate fi pusă în sfârşitul de pagină (`footer`) prin `include`. Împreună cu `include`, tabelele pot fi folosite pentru a specifica o anumită aranjare a obiectelor pe ecran.

Foi de stil (Style Sheets)

Modul de afişare a elementelor dintr-o pagină se poate defini şi prin folosirea foilor de stil. Acestea conţin grupuri de reguli de afişare. Fiecare regulă se compune dintr-un selector şi o declaraţie. Selectorul, identifică elementele din pagină ce vor fi afectate, iar declaraţia, specifică modul de afişare. Fiecare declaraţie constă din una sau mai multe perechi `proprietate:valoare`. Iată un exemplu:

```
P {color:red}
```

Selectorul este `P`, declaraţie este `{color:red}`. Proprietatea este `color` iar valoarea `red`. În situaţia în care dorim să folosim mai multe perechi `proprietate:valoare`, ele trebuie separate prin caracterul `;`.

Fiecare navigator are un mod implicit de afişare a elementelor HTML, acesta poate fi însă suprascris prin folosirea foilor de stil. Când mai multe stiluri se aplică aceluiaşi element dintr-o pagină, regula de selecţie a modului de afişare este cea a cascadării. Din acest motiv, în literatura de specialitate, foile de stil sunt prescurtate CSS (Cascading Style Sheet), adică foi de stil cascadate. Cascadarea utilizează moştenirea, specificitatea şi poziţia pentru determinarea regulii ce va fi aplicată afişării unui anumit element. În **Figura 26** se prezintă modul de afişare a codului HTML din fişierul `c1p3css1.htm`. Acesta are drept scop explicarea conceptelor de moştenire şi de specificitate. Înainte de aceasta voi prezenta, pe scurt, metodele prin care se pot da nume elementelor HTML. Scopul numirii elementelor este acela de identificare. Numele de element poate fi unic sau al unei clase. Acest nume va fi utilizat la aplicarea unui anumit stil tuturor elementelor cu acelaşi nume. Pentru identificarea unică a unui element, în marcajul de început vom scrie `id = "nume"`. Pentru asocierea unui element la o clasă, în marcajul de început se va specifica numele clasei prin `class= "nume"`. Pentru a selecta un anumit element în vederea aplicării unui anumit stil se va folosi un selector, astfel pentru `P.ingrup {color:blue}` numele de element este `P`, iar `ingrup` este numele de clasă. Regula de stil va fi aplicată asupra elementelor ce fac parte din clasa `ingrup`. Pentru cazul unui element identificat cu `id="centru"` regula de stil se scrie `P#centru {color:green; text-align:left}`.

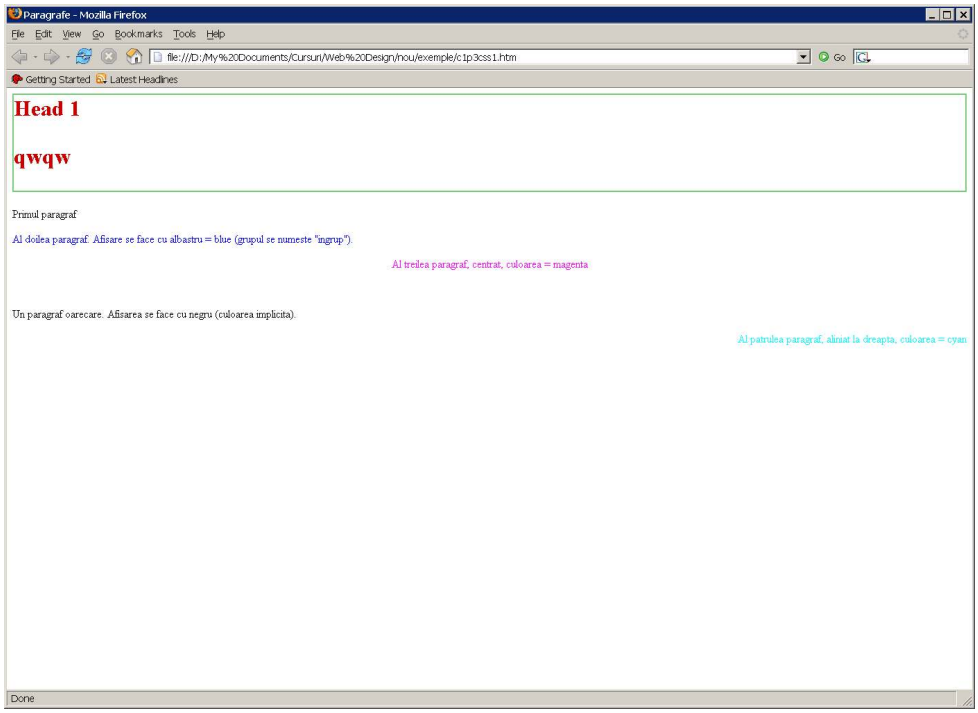
Fişierul `c1p3css1.htm`:

```
<HTML>
<HEAD>
<STYLE TYPE = "text/css">
      P.ingrup {color:blue}
      P#stinga {color:green; text-align:left}
      P#centru {color:magenta; text-align: center}
      P#dreapta {color:cyan; text-align: right}

      H1 {color:#c00;border:2px solid;border-color:#7c7}
</STYLE>
<TITLE>Paragrafe</TITLE>
</HEAD>
<BODY>
<H1>Head 1 <P>qwqw</H1>

<P>Primul paragraf</P>

<P class ="ingrup">Al doilea paragraf. Afisare se face cu albastru = blue
(grupul se numeste "ingrup").</P>

<P id="centru">Al treilea paragraf, centrat, culoarea = magenta</P>
<BR>
```

```
<P> Un paragraf oarecare.
Afisarea se face cu negru (culoarea implicita). </P>
<P  class="ingrup"  id="dreapta">Al  patrulea  paragraf,  aliniat  la  dreapta,
culoarea = cyan</P>
</BODY>
</HTML>>
```

Orice element are un grup de proprietăți, o parte dintre acestea sunt moştenite, altele sunt proprii elementului în cauză. Moştenirea defineşte ce anume se petrece cu un element când nu există reguli de stil aplicate explicit asupra lui. În exemplul prezentat, marcajul `<P>` , ce este cuprins în marcajul `<H1>`, are proprietatea culoare (`color`) moştenită de la marcajul `<H1>`. Ca urmare a moştenirii acestei proprietăți, conținutul paragrafului cuprins în antet fiind afişat şi el cu roşu.

Figure 26
Pagină cu CSS.



Antetul mai are şi proprietatea chenar (`border`) setată, aceasta însă nu este moştenită, motiv pentru care textul de paragraf cuprins în antet nu are chenarul afişat. Regula de specificitate se utilizează atunci când mai multe stiluri sunt aplicate unui element. Conform acestei reguli, cu cât selectorul este mai specific, cu atât mai puternic este stilul. În exemplul prezentat, ultimul paragraf are aplicat simultan doi selectori prin **`class="ingrup"  id="dreapta"`**. Deoarece atributul `id` este mai specific decât cel `class`, el trebuind să fie unic la nivel de pagină, stilul acestuia va fi cel aplicat paragrafului. Poziția unei regului contează şi ea, regulile care apar mai târziu sunt cele care domină.

Stocarea regulilor de stil

Regulile de stil pot fi stocate intern, adică în pagina asupra căreia dorim să acționeze sau extern, într-un fişier text distinct de pagina de Web. Atunci când regulile de stil sunt stocate intern acestea pot să acționeze asupra întregii pagini sau local, numai asupra elementelor cărora s-au asociat.

Stocarea externă a regulilor de stil

Fişierul ce va stoca aceste reguli de stil trebuie să fie unul text ASCII. Marcajul `LINK REL` ce trebuie să fie scris în secțiune `HEAD`, asigură legătura între pagina curentă şi foaia de stil externă,

după cum se vede în `c1p3css2.htm`.

Fişierul `c1p3css2.htm`:

```
<HTML>
<HEAD>
      <TITLE>Paragrafe</TITLE>
      <LINK REL="stylesheet" TYPE="text/css" HREF="c1p3css2.css">
</HEAD>
<H1>Head 1 <P>qwqw

</H1>

<P>Primul paragraf</P>

<P class ="ingrup">Al doilea paragraf. Afisare se face cu albastru = blue
(grupul se numeste

"ingrup").</P>

<P id="centru">Al treilea paragraf, centrat, culoarea = magenta</P>
<BR>
<P> Un paragraf oarecare.
Afisarea se face cu negru (culoarea implicita). </P>
<P class ="ingrup" id="dreapta">Al patrulea paragraf, aliniat la dreapta,
culoarea = cyan</P>
</BODY>
```


Atributul **`TYPE="text/css"`** specifică faptul că foaia de stil este scrisă în conformitate cu specificațiile CSS, iar în **`HREF="c1p3css2.css"`** este dat URL-ul cu numele fişierului ce stochează foaia cu stiluri. Acest URL este relativ la locația fişierului de stil pe serverul de Web şi nu relativ la locația paginii de Web.

Fişierul `c1p3css2.css`:

```
P.ingrup {color:blue}
P#stinga {color:green; text-align:left}
P#centru {color:magenta; text-align: center}
P#dreapta {color:cyan; text-align: right}

H1 {color:#c00;border:2px solid;border-color:#7c7}
```

Stocarea externă a foilor de stil se utilizează atunci când dorim să aplicăm aceleaşi stiluri mai multor documente. Marele avantaj al acestora constă în actualizarea imediată a modificărilor făcute într-o singură foaie de stil în toate paginile HTML ce folosesc respectiva foaie de stil.

Stocarea internă a regulilor de stil

Fişierul `c1p3css1.htm` prezintă modul de scriere a unei foi de stil interne. Există şi posibilitatea aplicării de stiluri local, la nivelul unui marcaj. La nivelul unui marcaj, regula de stil nu se scrie între acolade şi nu are nevoie de selector luând forma **`STYLE = " ..."`**.

Exemplu: Meniuri şi fonduri cu CSS

Pagina care urmează îşi propune crearea unui meniu principal, fără submeniuri, cu ajutorul CSS. Codul HTML corespunzător imaginii din **Figura 27** este stocat în fişierul `p2css2.htm`.

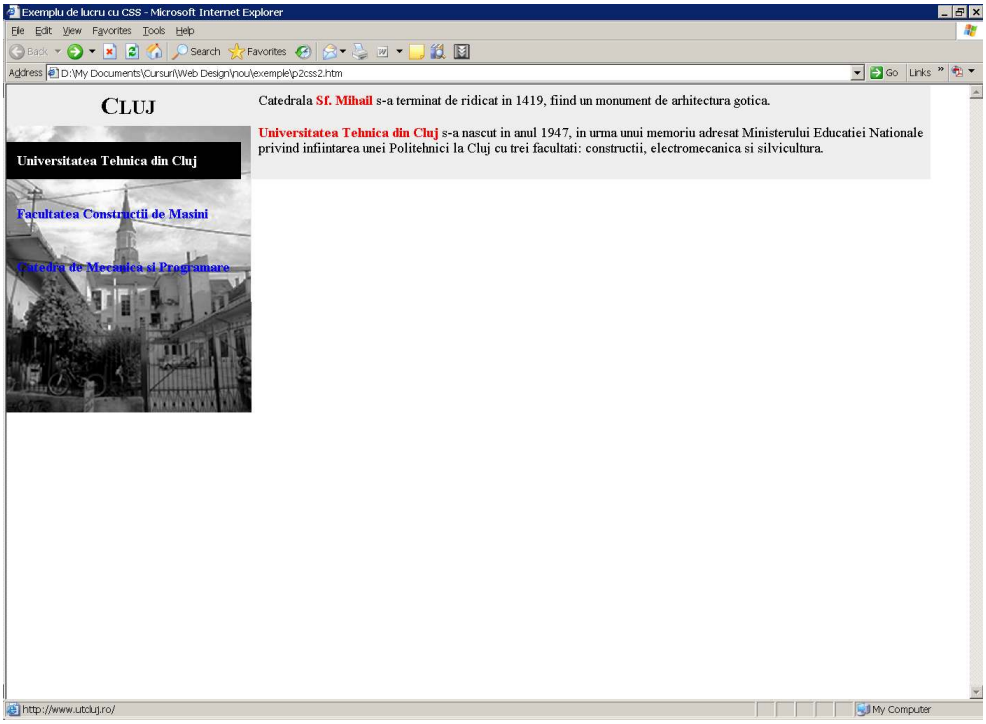


Figure 27 - Crearea unui meniu cu CSS.

Fişierul `p2css2.htm`:

```
<HTML>
<HEAD>
<TITLE>Exemplu de lucru cu CSS</TITLE>

<STYLE TYPE = "text/css">
      DIV#fond {
            position:absolute;
            top:1px;
            left:0px;
            z-index:1;
            background-color:white;
      }

      DIV#fond IMG {
            width:350px;
            color:white;
      }

      DIV#continut {
            position:absolute;
            top:1px;
            left:350px;
            background-color:#eee;
            padding:10px;
            z-index:2;
      }

      DIV#navigare {
            position:absolute;
            width:350px;
            top:1px;
            left:0px;
            border-right:1px solid #eee;
            z-index:3;
      }
```

```
      H1 {
            background-color:#eee;
            color:#000;
            font-size:34px;
            font-variant:small-caps;
            font-weight:bold;
            padding:10px;
            text-align=center;
      }

      SPAN.ingrosat {
            font-weight:bold;
            color:red;
      }

      A:LINK {
            text-decoration: none;
            font-weight:bold;
            color: #0000000;
            padding:15px;
            width:335px;
      }

      A:VISITED {
            text-decoration: none;
            font-weight:normal;
            padding:17px;
            width:335px;
      }

      A:HOVER {
            font-weight:bold;
            color: #ffffff;
            background-color:#000000;
            padding:15px;
            width:335px;
      }
</STYLE>
</HEAD>
<BODY>
<!-- Regiunea de navigare, apare in stanga paginii -->
<div id="navigare">
      <h1>Cluj</h1>
      <p><a href="http://www.utcluj.ro">Universitatea Tehnica din Cluj</a></p>
      <p><a  href="http://www.east.utcluj.ro/mb">Facultatea  Constructii  de
Masini</a></p>
      <p><a  href="http://www.east.utcluj.ro/mb/mep">Catedra  de  Mecanica  si
Programare</a></p>
      <p>
</div>

<!-- Continutul documentului, apare in drepata regiunii de navigare -->
<div id="continut">
      Catedrala <span  class="ingrosat">Sf.  Mihail</span>  s-a  terminat  de
ridicat in 1419, fiind un monument de arhitectura gotica.
      <p><span class="ingrosat">Universitatea Tehnica din Cluj</span>  s-a
nascut in anul 1947, in urma unui memoriu adresat Ministerului Educatiei
Nationale privind infiintarea unei Politehnici la Cluj cu trei facultati:
constructii, electromecanica si silvicultura.
      <P>
</div>

<!-- Fondul paginii, apare afisat, datorita sterilor de stil, in stanga, sub
regiune de navigare -->
<div id ="fond">
```

```
      <img src="catedrala.gif" alt = "Catedrala Sf. Mihail"></p>
</div>
</BODY>
</HTML>
```

Dacă o pagină de Web foloseşte CSS, fiecare element este conținut într-un dreptunghi invizibil. Dreptunghiul are mai multe regiuni, câteva dintre cele mai importante fiind: conținutul, spațiul ce înconjoară conținutul (`padding`), marginea exterioară a padding-ului (`border`), spațiul invizibil din jurul border-ului (`margin`). Utilizarea CSS permite definirea poziției şi a modului de afişare a elementelor din pagină. Implicit, elementele sunt afişate în ordinea în care sunt scrise în codul HTML, de la stânga la dreapta şi de sus în jos. Pentru ca un element să fie afişat în pagină, într-o poziție fixă, independent de poziția codului de element, sistemul de coordonate utilizat trebuie să fie absolut (**`position:absolute`**). Dacă mai multe astfel de elemente se suprapun, ordinea de suprapunere poate fi şi ea controlată prin **`z-index`**. Mai sus există trei elemente, **`fond`**, **`continut`** şi **`navigare`** ce sunt poziționate absolut. Observați că, deşi ordinea lor în codul HTML este **`navigare`**, **`continut`** şi **`fond`**, afişarea lor se face fără a se ține cont de poziția relativă a elemenelor din document. În contextul acestei aplicații, **`fond`** nu se referă la fondul întregii pagini, ci doar la fondul unui element particular. Avantajul este acela că apare posibilitatea schimbării fondului pentru orice element, ținând cont de poziția lui absolută. Meniul are la bază modul de afişare a hiperlegăturilor, existând definițiile **`A:LINK`** , **`A:VISITED`** şi **`A:HOVER`**, unde este specificată afişarea: la deschiderea paginii, după vizitare şi la poziționarea pe hiperlegătură. Pentru a face dispărută sublinierea hiperlegăturii se foloseşte **`text-decoration: none`**, iar pentru a da efectul "poziționare pe un articol de meniu" se va modifica fondul, prin **`background-color:#000000.`**

Exemplu: Tabele cu CSS

În **Figura 28** este prezentată o pagină de Web ce foloseşte stiluri pentru definirea aspectelor celor două tabele. Codul HTML din `tabelcss.html` are câteva paticularități. Observați că regulile de stil sunt scrise general respectiv acestea nu mai conțin numele elementului particular care dorim să fie afectat. De asemenea, regulile sunt corpul unui comentariu HTML. În situația în care navigatorul nu recunoaşte aceste regului, le va considera ca parte a comentariului şi nu le va afişa, altfel, regulile vor fi luate în considerare şi marcajul de comentariu va fi ignorat. Am încercat să prezint cât mai multe proprietăți. Deşi majoritatea nu le-am explicat, contextul în care apar şi numele acestora sugerează semnificația lor.

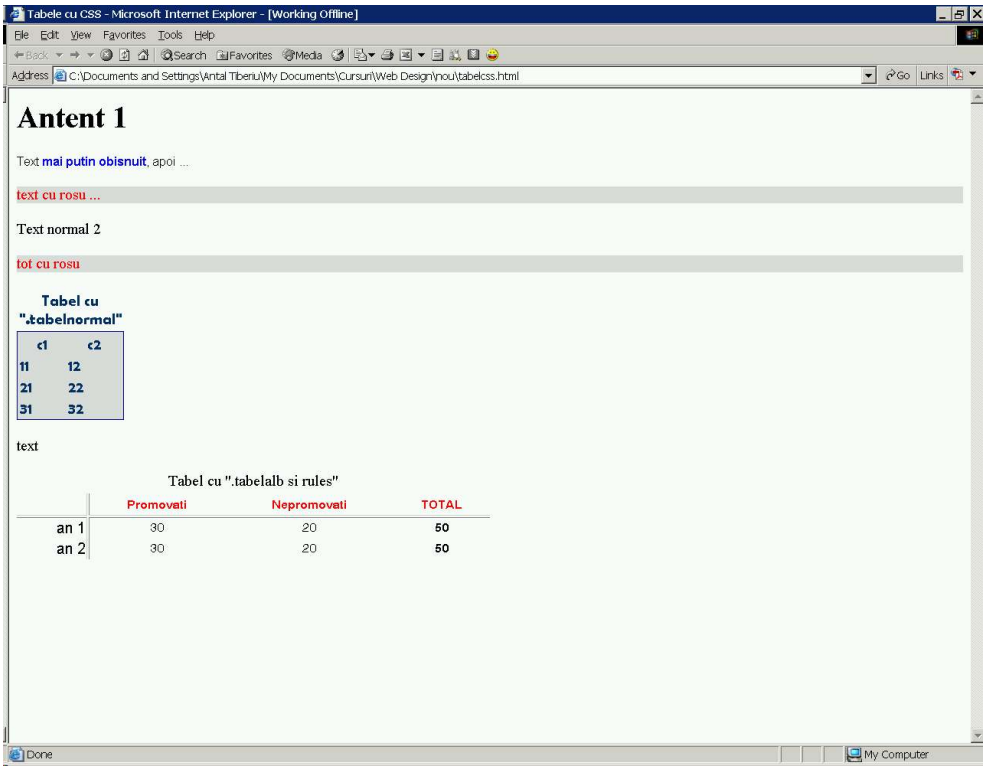
Fişierul `tabelcss.html` :

```
<HTML>
<HEAD>
  <TITLE>Tabele cu CSS</TITLE>

<STYLE TYPE="text/css">
<!--
      .rosu {color: #FF0000; background: #dddddd;text-decoration: none;}
      #gros {color: #0000FF; font-weight: bold;}

      .textnormal
      {
       font-family: Arial, Helvetica, sans-serif;
       font-size: 16px;
       color: #000000;
       text-decoration: none;
       border: none;
      }

      .tabelnormal {
            font-family: Berlin Sans FB Demi, Sans-serif;
```

```
            font-size: 20px;
            color: #003366;
            background: #dddddd;
            border-left: 1px solid #000080;
            border-right: 1px solid #000080;
            border-top: 1px solid #000080;
            border-bottom: 1px solid #000080;
      }

      .tabelalb {color: #000000; background: #FFFFFF;}

      .capdetabel {color: #FF0000; background: #FFFFFF;text-decoration: none;}

      .caption{
            font-family: "Arial", "Verdana", sans-serif;
            font-weight:bold;
            color:#FF9900}

      .anidestudiu {
            font: 20px "Arial", "Verdana", sans-serif;
            text-align:right}
      .date {font:16px "Helvetica", "Verdana", sans-serif;text-align:center}

      .totale {font-weight: bold;}
-->
</STYLE>

</HEAD>
<BODY>
<H1>Antent 1</H1>
<P class="textnormal">Text <SPAN id="gros">mai putin obisnuit</SPAN>, apoi ...
<P class = rosu> text cu rosu ...
<P> Text normal 2
<P class = rosu> tot cu rosu
<P>

<TABLE class="tabelnormal">
      <CAPTION ALIGN=top>Tabel cu ".tabelnormal"</CAPTION>
      <THEAD>
            <Th>c1</Th> <Th>c2</Th>
      </THEAD>
      <TR>
            <TD>11 </TD> <TD>12 </TD>
      </TR>
      <TR>
            <TD>21 </TD> <TD>22 </TD>
      </TR>
      <TR>
            <TD>31 </TD> <TD>32 </TD>
      </TR>
</TABLE>
<P>text
<P>
<TABLE class="tabelalb" frame="void" rules="groups" width=50%>
      <CAPTION ALIGN=top>Tabel cu ".tabelalb si rules"</CAPTION>
      <COLGROUP class="anidestudiu">
            <COLGROUP SPAN=3 class="date">
            <COL SPAN=2>
            <COL class="totale">
      </COLGROUP>
      <THEAD class="capdetabel">
            <TH> </TH>
            <TH>Promovati</TH>
            <TH>Nepromovati</TH>
            <TH>TOTAL</TH>

      </THEAD>
      <TR>
```

```
            <TD>an 1</TD> <TD>30</TD> <TD>20</TD> <TD>50</TD>
      </TR>
      <TR>
            <TD>an 2</TD> <TD>30</TD> <TD>20</TD> <TD>50</TD>
      </TR>
<TABLE>
<BODY>
<HTML>
```

Figure 28
Tabele cu CSS



CSS permite divizarea unui tabel în grupuri de coloane. Avantajul folosirii acestor grupuri constă în posibilitatea aplicării unei anumite reguli de stil unei coloane sau unui grup de coloane. Gruparea pe coloane se poate face cu două marcaje: `COLGROUP` şi `COL`. Primul, permite controlul afişării sau inhibării liniilor din chenarul de grup. După marcajul `TABLE` se va scrie marcajul `COLGROUP`. Dacă din grup vor face parte mai multe coloane, atunci valoarea "`n`" din `SPAN=n` va defini numărul coloanelor din grup. Controlul afişării chenarelor la nivelul tabelului se face cu ajutorul marcajului `FRAME="loc"`. Câteva dintre valorile pe care le poate lua `loc` sunt: **`box`** - este valoarea implicită, produce afişarea tuturor liniilor din chenar; **`none`** - nu se afişează, **`above`** - pentru o singură linie în partea de sus, **`rhs`** - o singură linie la stânga etc. Controlul vizibilității liniilor chenarelor interioare se face cu `RULES="zona"`. Valoarea zona poate fi: **`none`**- nu există reguli; **`rows`** - pentru reguli pe orizontală, între liniile de tabel; **`cols`** - reguli pe verticală, între coloanele de tabel; **`groups`** - pentru reguli la nivel de grupuri de coloane create cu `COLGROUPS`; **`all`** - valoarea implicită, regulile se aplică pentru toate coloanele şi rândurile de tabel.

Formulare (Forms)

Marcajul FORM este un element bloc care permite definirea unui formular HTML. Aspectul vizual al formularelor este foarte variat plecând de la simple controale de tipul cutie de text până la butoane sau meniuri. Pentru a înțelege ce sunt formularele închipuiți-vă o fereastră de dialog. Majoritatea aplicațiilor au aceste ferestre pentru a permite utilizatorului să intervină în anumite situații speciale sau pentru a modifica anumite setări implicite, eventual pentru a furniza informații în plus față de cele cunoscute deja. Formularele HTML sunt echivalentul Web al ferestrelor de dialog. Legat de formulare se disting două etape în proiectare: definirea structurii formularului şi acțiunile pe care le va realiza acesta. Definirea structurii formularului se face asemenea oricărei pagini de Web, pe baza unor elemente HTML, efectul fiind, unul vizual. În acest caz însă este important ca fiecare element să primească un nume, deoarece acest nume vor fi utilizate la prelucrarea datelor din formular. Faptul că un formular a fost publicat pe Web nu înseamnă nimic mai mult decât o interfață. Prelucrarea datelor introduse în formular se face pe baza unor limbaje de programare simple, numite generic, script-uri care se execută pe server-ul de Web. Script-ul citeşte datele din formular, le prelucrează, de exemplu prin adăugarea într-o bază de date, apoi întoarce un rezultat, de exemplu textul "Ați fost adăugat în baza de date." Formularele nu pot fi imbricate, dar pot să existe mai multe într-o singură pagină. Dacă furnizorul de Internet nu vă asigură facilitatea de rularea script-urilor pe server-ul de Web, datele unui formular pot fi trimise şi pe o adresă de e-mail.

Definirea unui formular

Cea mai simplă definiție de formular este:


```
<HTML>
<HEAD>
<TITLE>Formulare 1</TITLE>
</HEAD>
<BODY>
      <FORM>
      </FORM>
</BODY>
```

Salvați formularul pe disc sub numele de frm1.htm şi păstrați deschise pe ecran simultan, editorul de texte (de exemplu, notepad-ul) şi pagina Web. Atunci când doriți să reîncărcați conținutul paginii apăsați simultan **Ctrl** şi cea de reîncărcare (**F5** - **Refresh** în IE). În continuare trebuie să spunem navigatorului unde să fie trimise datele adunate şi cum să fie acestea trimise. La acest nivel datele fie se trimit unui script pe server pentru prelucrare, fie se trimit ca e-mail, unei persoane care le va prelucra ulterior. În primul caz, cel care a scris script-ul ştie modul în care trebuie trimise datele. În cazul al doilea, formularul trebuie să conțină următoarele atribute:

```
<HTML>
<HEAD>
<TITLE>Formulare 1</TITLE>
</HEAD>
<BODY>
      <FORM METHOD=POST ACTION="mailto:xxx@xxx.xxx"
ENCTYPE="application/x-www-form-urlencoded">
      </FORM>
</BODY>
```

Mai sus, trebuie completată adresa de e-mail a destinatarului, iar restul de text trebuie păstrat întocmai. Dacă formularul are câmpurile NUME, PRENUME, ADRESA şi ORAS acestea vor fi transmise sub forma:

```
NUME=Vasile&PRENUME=Ionel+Adrian&ADRESA=Oasului+nr+14&ORAS=Fara
```

în loc de:

```
NUME=Vasile
PRENUME=Ionel Adrian
ADRESA=Oasului nr 14
ORAS=Fara
```

Unele programe de mail ştiu să convertească datele fără să necesite ajutorul unui program extern. Dacă doriți să testați această variantă în locul lui `ENCTYPE="application/x-www-form-urlencoded"` scrieți `ENCTYPE="text/plain"`.

Atunci când suntem interesați în scrierea de script-uri forma generală pentru definiția unui formular este:

```
<FORM ACTION="URL" METHOD=METODA>
</FORM>
```

Aici, atributul `ACTION` spune navigatorului unde să trimită datele formularului. În acest caz *URL* va fi adresa programului script care prelucrează datele (de exemplu, `ACTION="http://postinfo.east.utcluj.ro/AdaugaClient1.asp"`).

Atributul `METHOD` spune navigatorului cum se trimit datele la URL-ul specificat prin atributul `ACTION`. Sunt disponibile doar două variante:

- `GET`: Navigatorul adaugă un `semn de întrebare` (`?`) urmat de datele din formular la sfârşitul atributului `ACTION` şi apoi cere de la server această combinație de `URL` şi date;
- `POST`: Navigatorul trimite datele culese din formular către server într-un mesaj separat.

Atunci când se alege metoda folosită trebuie să țineți cont de faptul că pentru fiecare control de pe suprafața formularului se trimit către server numele controlului şi datele introduse în acesta. Astfel, formularul poate să ajungă să trimită foarte multe date server-ului, mai ales dacă are pe suprafața lui un număr mare de controale, sub forma unui şir de caractere foarte lung. În unele cazuri lungimea URL-ului trimis către server-e este restricționată, astfel combinația URL?date generată de metoda GET poate sfârşi trunchiată.

Elemente de intrare

Este caracteristic a acestor element că nu au marcajul de terminare, sau în termeni mai tehnici, nu sunt containere. Forma cea mai simplă prin care se defineşte un element de intrare este:

```
<INPUT TYPE="TIP" NAME="NUME">
```

Unul dintre atributele importante este `VALUE = "sir"` şi permite atribuirea unei valori elementului de intrare. `TYPE` poate lua mai multe valori predefinite specificând tipul controlului de intrare (text, parolă, imagine, selector checkbox, selector radio etc.). `NAME` este numele pe care elementul de intare îl primeşte, el trebuie sa fie unic în pagină iar specificarea lui este obligatorie. Mai trebuie reținut că toate aceste elemente se scriu între `<FORM> </FORM>`, fiind specifice unui formular.

Butonul Submit

Majoritatea ferestrelor de dialog au un buton **OK.** Apăsarea lui are semnificația acceptării

setărilor făcute împreună cu generarea efectelor lor. Formularele pot avea şi ele butoane de două tipuri: transfer (`submit`) şi resetare (`reset`). Butoanele de tipul transfer sunt echivalentul lui **OK.** Când se face clic pe un astfel de buton conţinutul formularului este trimis programului specificat în atributul `ACTION` al marcajului `<FORM>`. Cea mai simplă variantă este:

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

În acest caz valoarea dată în `VALUE` este un text care va fi afişat pe buton.

O variaţie pe tema butonului de transfer este folosirea unei imagini în acelaşi scop. Forma generală de scriere în acest caz este:

```
<INPUT TYPE=SUBMIT SRC="cale/fişier">
```

Butonul Reset

În cazul unor formulare mari este utilă includerea în formular a unui buton de `reset`. La apăsarea lui datele introduse deja în formular sunt şterse şi în locul lor reapar cele implicite. Cea mai simplă variantă de buton de reset este:

```
<INPUT TYPE=RESET VALUE="text">
```

Un exemplu de formular cu două butoane este:


```
<HTML>
<HEAD>
<TITLE>Exemplu de transfer al datelor</TITLE>
</HEAD>
<BODY>
<H3>Date despre mine</H3>
      <FORM ACTION="mailto:xxx@xxx.xxx" METHOD=POST>
            <INPUT TYPE=SUBMIT VALUE="Trimite">
            <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```

Introducerea textelor în formulare

Pentru introducerea unei singure linii de text, de exemplu, un nume, prenume sau o adresă de persoană, se foloseşte elementul de intrare de tipul `TEXT` , numit uneori cutie de text (`text box`), cu următoarea scriere:

```
<INPUT TYPE=TEXT NAME="numecâmp">
```

Aici, *`numecâmp`* este un nume unic, la nivelul formularului, pe care îl dăm câmpului de intrare. Iată un exemplu de utilizare:

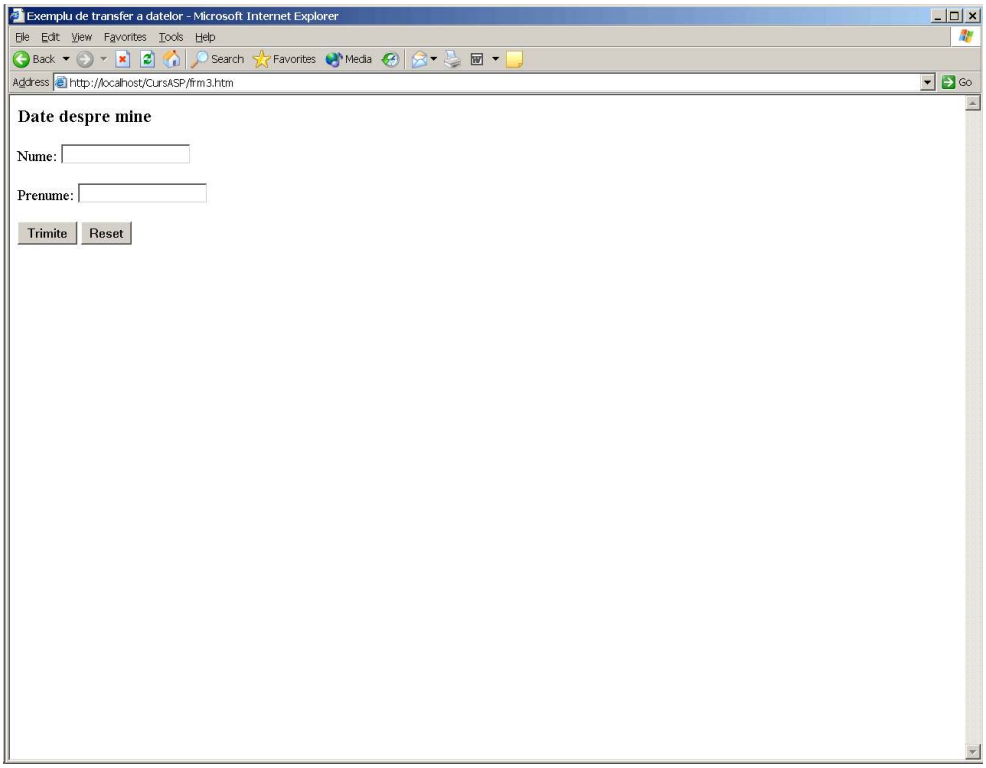
```
<HTML>
<HEAD>
<TITLE>Exemplu de transfer a datelor</TITLE>
</HEAD>
<BODY>
<H3>Date despre mine</H3>
      <FORM ACTION="mailto:xxx@xxx.xxx" METHOD=POST>
            Nume: <INPUT TYPE=TEXT NAME="Nume">
            <P>
            Prenume: <INPUT TYPE=TEXT NAME="Prenume">
            <P>
            <INPUT TYPE=SUBMIT VALUE="Trimite">
```

```
            <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```

IE va afişa acest formular după cum se vede în **Figura 29**. Cutiile de text pot fi făcute să aibă o **valoare implicită**, de exemplu dacă în locul liniei corespunzătoare din codul de mai sus scriem linia:

```
<INPUT TYPE=TEXT NAME="Nume" VALUE="Introduce numele">
```

Figure 29
Formular cu butoane şi cutii de text



atunci, la deschiderea formularului, textul între ghilimele va fi afişat automat în prima cutie de text. O altă posibilitate este **specificarea lungimii**, în număr de caractere, **de afişare** a cutiei de text prin atributul `SIZE`, iată un exemplu:

```
<INPUT TYPE=TEXT NAME="Nume" SIZE=50>
```

Atributul anterior determină numai lungimea de afişare a cutei de texte, nu şi numărul maxim de caractere care pot fi introduse în aceasta. Atributul `MAXLENGHT` defineşte **numărul maxim de caractere** care pot fi introduse în cutia de text. Iată un exemplu:

```
<INPUT TYPE=TEXT NAME="Nume" MAXLENGTH=20>
```

O variantă specială a cutiei de text este aceea în care se introduce o **parolă**, în acest caz în locul textului introdus se vor afişa doar asteriscuri, codul fiind de forma:

```
<INPUT TYPE=PASSWORD NAME="numecâmp">
```

Dacă este important ca utilizatorul să aibă mai mult spaţiu pentru introducerea textului, sau dorim să introducem un **text organizat pe mai multe linii**, în locul cutiei de text, se va folosi elementul

zonă de text (`text area`). Acesta poate fi privit ca un dreptunghi în care se pot introduce date organizate pe mai mult de o linie. Forma generală este:

```
<TEXTAREA NAME="numecâmp" VALUE="text" ROWS=nrrând COLS=nrcol WRAP>
</TEXTAREA>
```

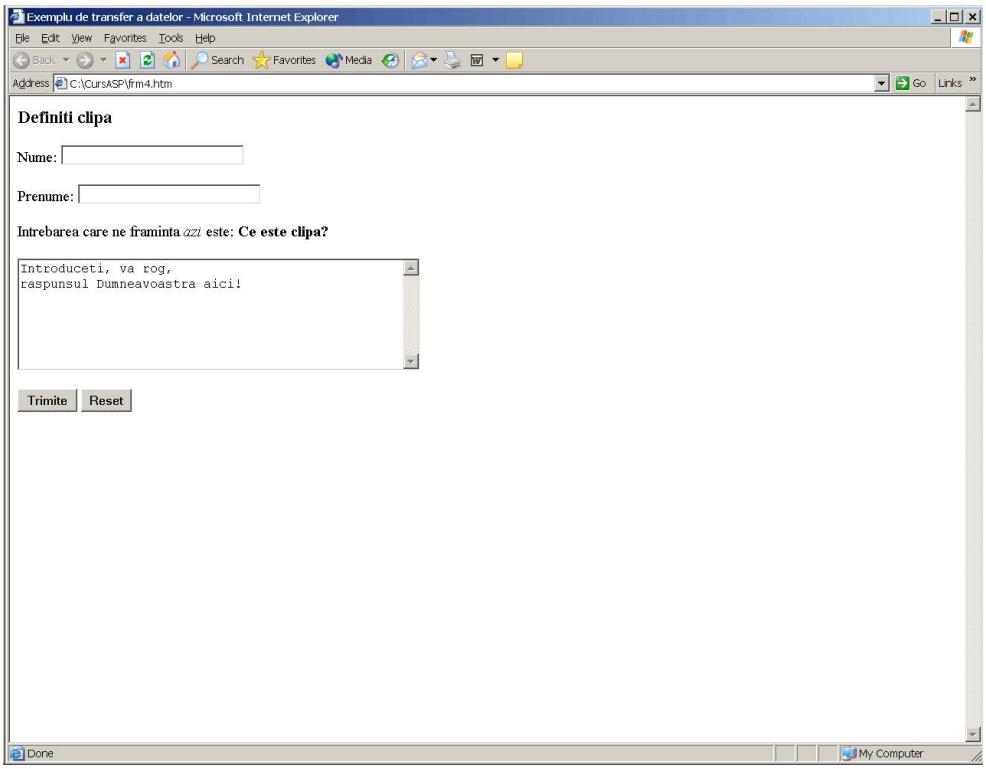
Aici, `numecâmp` este un nume care identifică unic marcajul, `text` este textul de afişat, `nrrând` numărul total de linii care se afişează şi `nrcol` numărul total de coloane afişate. Atributul `WRAP` spune navigatorului dacă textul trebuie să fie divizat pe linii sau nu, atunci când se ajunge la marginea din dreapta a unei linii. Observați că marcajul <`TEXTAREA`> necesită şi terminatorul `</TEXTAREA>`. Valorile implicite pot fi scrise pe una sau mai multe linii între aceste marcaje după cum se va vedea în exemplul următor:

```
<HTML>
<HEAD>
<TITLE>Exemplu de transfer al datelor</TITLE>
</HEAD>
<BODY>
<H3>Definiti clipa</H3>
      <FORM ACTION="mailto:xxx@xxx.xxx" METHOD=POST>
            Nume: <INPUT TYPE=TEXT NAME="Nume" SIZE=30 MAXLENGTH=30>
            <P>
            Prenume: <INPUT TYPE=TEXT NAME="Prenume" SIZE=30 MAXLENGTH=30>
            <P>
            Intrebarea care ne framinta <I>azi</I> este: <B>Ce este clipa?</B>
            <P>
            <TEXTAREA NAME="raspuns" ROWS=7 COLS=50 WRAP>
Introduceti, va rog,
raspunsul Dumneavoastra aici!
            </TEXTAREA>
            <P>
            <INPUT TYPE=SUBMIT VALUE="Trimite">
            <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```


IE afişează formularul sub forma prezentată în **Figura 30**.

`WRAP` poate lua valorile `VIRTUAL`, `PHYSICAL` şi `OFF`. Dacă navigatorul nu le înțelege, le va ignora. Valoarea defineşte modul în care textul introdus va fi trimis către server. Pentru valoarea `VIRTUAL` se realizează trecerea pe linia următoare când se ajunge la marginea din dreapta, dar textul este trimis sub forma unui şir continuu şi lung. În cazul lui `PHYSICAL` textul este divizat pe linii şi trimis sub formă de mai multe şiruri. Valoarea implicită este aceea de `OFF`, aici nu se ține cont de marginea din dreapta la afişare, astfel scrierea se face pe o singură linie, iar şirul este trimis exact aşa cum s-a introdus de la tastatură.

Figure 30
Exemplu cu zonă de text



Selectarea mai multor opţiuni, dintre mai multe variante, prin butoane de validare

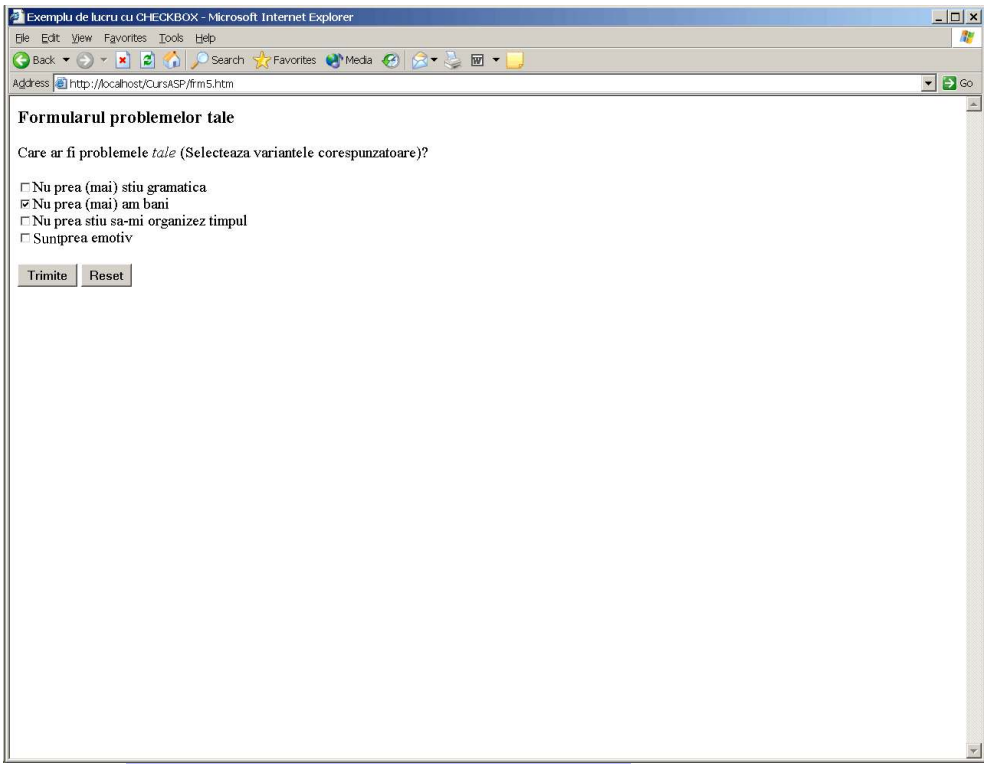
Atunci când se doreşte colectarea unor informaţii prin răspunsuri de tipul da/nu sau adevărat/fals se vor utiliza butoanele de validare (`check box`). Forma generală de scriere este:

```
<INPUT TYPE=CHECKBOX NAME="numecâmp">
```

Şi aici *numecâmp* este un nume de câmp unic. În plus, se poate adăuga atributul `CHECKED` care spune navigatorului să afişeze butonul de validare activat. Codul care urmează este afişat de IE aşa cum se vede în **Figura 31**.

```
<HTML>
<HEAD>
<TITLE>Exemplu de lucru cu CHECKBOX</TITLE>
</HEAD>
<BODY>
<H3>Formularul problemelor tale</H3>
      <FORM ACTION="mailto:antaltibi@pcnet.ro" METHOD=POST>
      <P>
      Care ar fi problemele <I>tale</I> (Selecteaza variantele
corespunzatoare)?
      <P>
      <INPUT TYPE=CHECKBOX NAME="Agramat">Nu prea (mai) stiu gramatica<BR>
      <INPUT TYPE=CHECKBOX NAME="Sarac" CHECKED>Nu prea (mai) am bani<BR>
      <INPUT TYPE=CHECKBOX NAME="Dezorganizat">Nu prea stiu sa-mi organizez
timpul<BR>
      <INPUT TYPE=CHECKBOX NAME="Emotiv">Sunt prea emotiv<BR>
      <P>
      <INPUT TYPE=SUBMIT VALUE="Trimite">
      <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```

Figure 31 Exemplu de formular cu checkbox



Selectarea unei singure opţiuni, dintre mai multe variante, prin butoane de opţiune

Dacă în locul unor răspunsuri de tipul da/nu se doreşte colectarea unor informaţii care au mai mult de două variante de răspunsuri se vor folosi butoane de opţiune (`option butons`). Cu acest buton utilizatorul are la dispoziţie mai multe variante, dar nu poate selecta decât o singură opţiune. Iată forma generală:

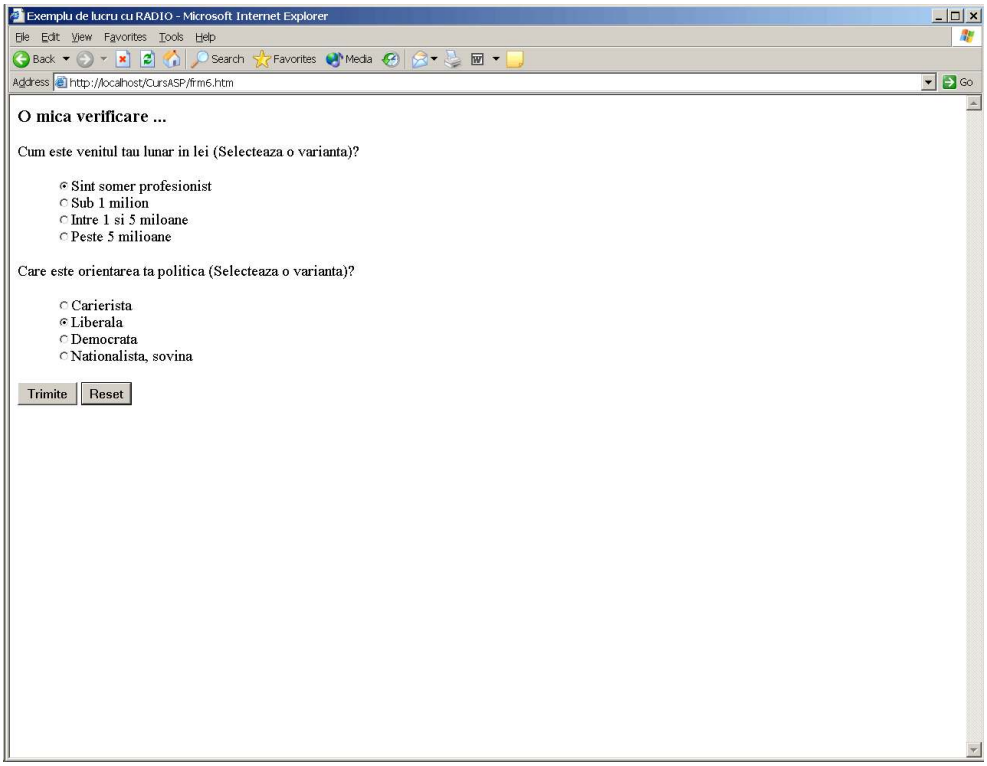
```
<INPUT TYPE=RADIO NAME="numecâmp" VALUE="valoare">
```

numecâmp este numele cîmpului, dar în acest caz el este acelaşi pentru toate variantele dintre care va selecta o singură opţiune. În acest fel navigatorul va şti cum anume să grupeze butoanele. `valoare` este un şir unic care specifică prin valoarea lui opţiunea selectată. În plus, se poate folosi şi aici atributul `CHECKED` pentru a defini varianta care este selectată implicit. Codul care urmează se afişează de IE aşa cum se vede în **Figura 32**.

```
<HTML>
<HEAD>
<TITLE>Exemplu de lucru cu RADIO</TITLE>
</HEAD>
<BODY>
<H3>O mica verificare ...</H3>
      <FORM ACTION="mailto:antaltibi@pcnet.ro" METHOD=POST>
      <P>
      Cum este venitul tau lunar in lei (Selecteaza o varianta)?
      <UL>
      <INPUT TYPE=RADIO NAME="Salar" VALUE="Fara" CHECKED>Sint somer
profesionist<BR>
      <INPUT TYPE=RADIO NAME="Salar" VALUE="Sarac">Sub 1 milion<BR>
      <INPUT TYPE=RADIO NAME="Salar" VALUE="Rezist">Intre 1 si 5 miloane<BR>
      <INPUT TYPE=RADIO NAME="Salar" VALUE="Hehe">Peste 5 milioane<BR>
      </UL>
      Care este orientarea ta politica (Selecteaza o varianta)?
```

```
      <UL>
      <INPUT TYPE=RADIO NAME="Politica" VALUE="Trestie">Carierista<BR>
      <INPUT TYPE=RADIO NAME="Politica" VALUE="Liberal" CHECKED>Liberala<BR>
      <INPUT TYPE=RADIO NAME="Politica" VALUE="Democrat">Democrata<BR>
      <INPUT TYPE=RADIO NAME="Politica" VALUE="Cvt">Nationalista, sovina<BR>
      </UL>
      <P>

      <INPUT TYPE=SUBMIT VALUE="Trimite">
      <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```

Figure 32 Formular cu butoane de opțiune



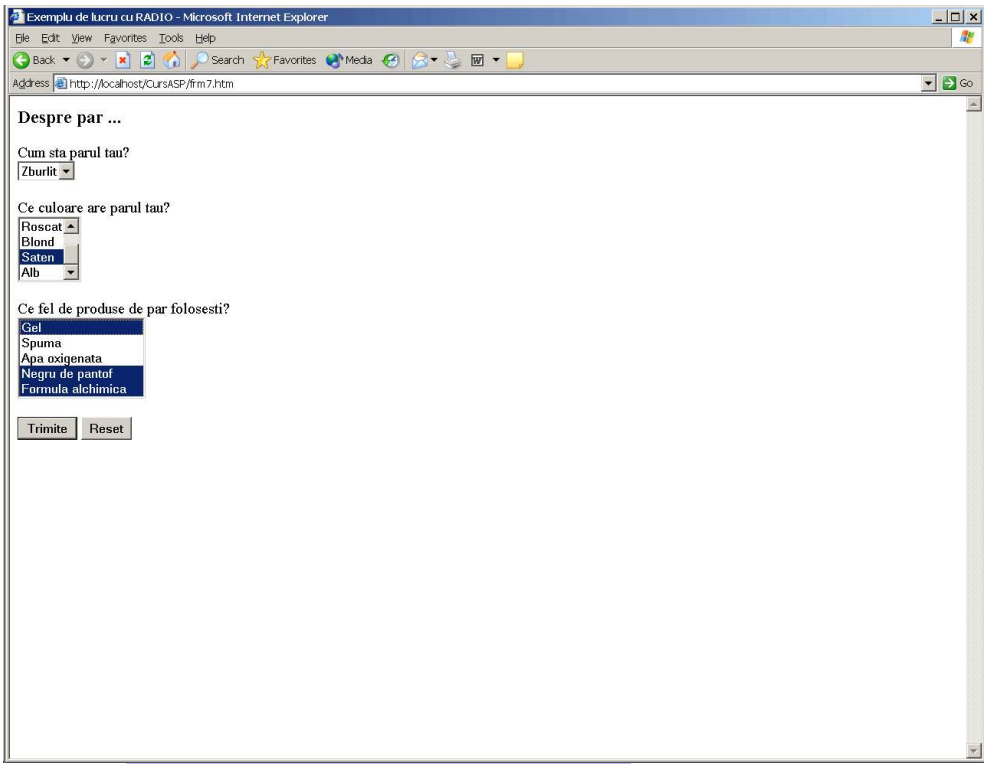
Selectarea din liste

Butoanele de opțiune sunt o variantă elegantă de selectare a unei opțiuni dacă numărul de variante nu este mai mare de şase, şapte. Pentru mulțimi de variante cu mai multe elemente, listele (`lists`) sunt mai bune. Crearea unei liste necesită scrierea unui cod cu mai multe marcaje decât în celelalte cazuri. Iată forma generală:

```
<SELECT NAME="numecâmp" SIZE=numardearticole>
<OPTION>Primul articol</OPTION>
<OPTION>Al doilea articol</OPTION>
<OPTION>Si asa mai departe ...</OPTION>
</SELECT>
```

Atributul `SIZE`, prin numărul întreg `numardearticole`, permite definirea numărului de articole pe care le va afişa navigatorul. Dacă acesta este omis, lista devine una desfăşurabilă (`drop-down list`); dacă `SIZE` ia o valoare mai mare de 1, ea devine un dreptunghi cu bare pentru defilare, pentru selectarea opțiunii. Folosirea atributului `MULTIPLE` permite selectarea simultană a mai multor variante din listă. Dacă unei opțiuni se i adaugă atributul `SELECTED` articolul corespunzător devine cel implicit selectat la afişarea listei de către navigator. Iată un exemplu de lucru cu listele, formularul corespunzător, afişat de IE, este prezentat în **Figura 33**.

Figure 33
Formular cu liste de variante




```
<HTML>
<HEAD>
<TITLE>Exemplu de lucru cu RADIO</TITLE>
</HEAD>
<BODY>
<H3>Despre par ...</H3>
      <FORM ACTION="mailto:antaltibi@pcnet.ro" METHOD=POST>
      <P>
      Cum sta parul tau?<BR>
      <SELECT NAME="Stil">
      <OPTION>Lins</OPTION>
      <OPTION SELECTED>Zburlit</OPTION>
      <OPTION>Cret</OPTION>
      </SELECT>
      <P>
      Ce culoare are parul tau?<BR>
      <SELECT NAME="Culoare" SIZE=4>
      <OPTION>Negru</OPTION>
      <OPTION>Brunet</OPTION>
      <OPTION>Roscat</OPTION>
      <OPTION>Blond</OPTION>
      <OPTION SELECTED>Saten</OPTION>
      <OPTION>Alb</OPTION>
      </SELECT>
      <P>
      Ce fel de produse de par folosesti?<BR>
      <SELECT NAME="Produs" SIZE=5 MULTIPLE>
      <OPTION>Gel</OPTION>
      <OPTION>Spuma</OPTION>
      <OPTION>Apa oxigenata</OPTION>
      <OPTION>Negru de pantof</OPTION>
      <OPTION>Formula alchimica</OPTION>
      </SELECT>
      <P>

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

```
      <INPUT TYPE=RESET VALUE="Reset">
      </FORM>
</BODY>
```

Controale ascunse

Atunci când într-o aplicație se scriu formulare multe, se poate ca unele dintre ele să fie asemănătoare prin aşezarea şi tipul de controale folosite. În cazul formularelor care sunt puțin diferite nu are sens scrierea de script-uri diferite pentru prelucrarea datelor. Pentru a folosi un singur script este nevoie de o modalitate de ramificare, la nivelul lui, în funcție de formularul utilizat. O metodă simplă de a realiza ramificarea este folosirea unor controale ascunse în fiecare formular după cum urmează:

```
<INPUT TYPE=HIDDEN NAME="numecâmp" VALUE="valoare">
```

Controalele de tipul `HIDDEN` nu sunt vizibile pentru utilizator, dar atributele sunt trimise unui script în vederea prelucrării. Fie exemplul:

```
<INPUT TYPE=HIDDEN NAME="NumeFormular" VALUE="Formularul 1">
```

Script-ul va putea testa variabila *`NumeFormular`*. Dacă valoarea ei este `Formularul 1` atunci se vor realiza anumite prelucrări. Altfel, dacă de exemplu, numele ar fi `Formularul 2`, se pot realiza prelucrări ale datelor printr-un grup de instrucțiuni diferite de primul caz.

Transmiterea datelor din formular prin e-mail

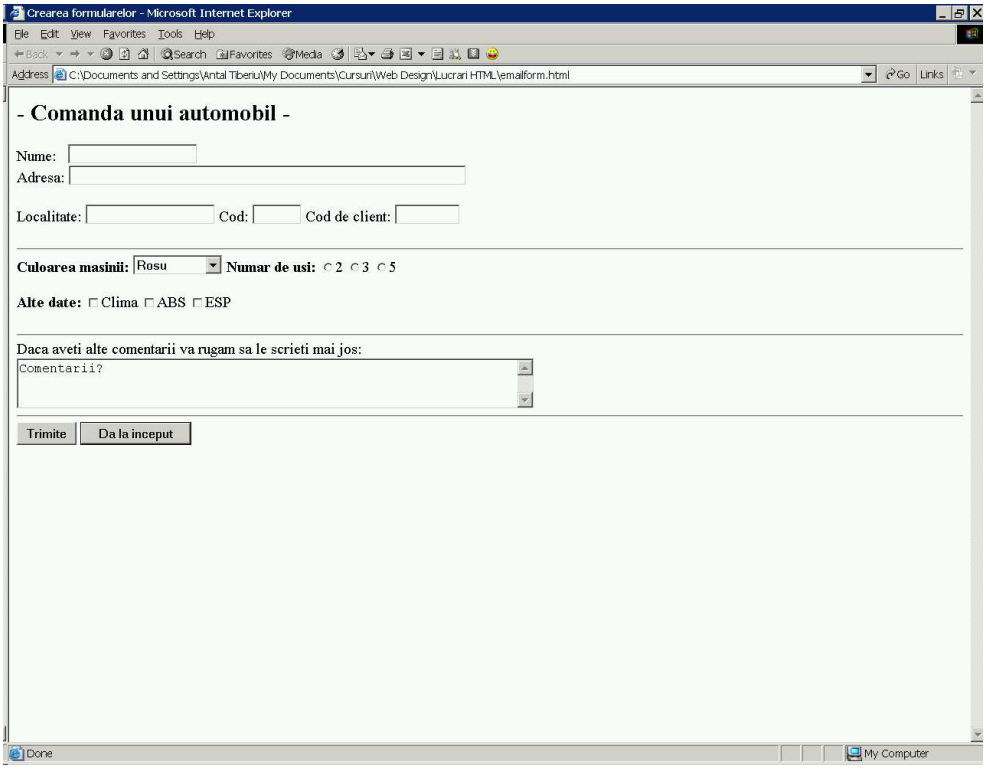
În **Figura 34** se prezintă un formular al cărui conținut va fi transmis la adresa de e-mail **`numeutiliz@server.ro`**. Conținutul documentului este prezentat în fişierul `emailform.html`. La apăsarea butonului `Trimite`, apare o fereastră de dialog în care vizitatorul paginii este avertizat ca adresa lui de e-mail va fi vizibilă la destinație şi că datele din formular nu sunt criptate. Pentru ca datele să fie trimise trebuie apăsat butonul OK. Acestea vor fi conținute în corpul mesajului de e-mail.

Fişierul `emailform.html`:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="CONTENT-TYPE "CONTENT="TEXT/HTML; CHARSET=ISO-8859-1" />
<TITLE>Crearea formularelor</TITLE>
</HEAD>
<BODY>
<H2> - Comanda unui automobil - </H2>

<FORM METHOD="POST" ENCTYPE="TEXT/PLAIN" ACTION="mailto:numeutiliz@server.ro">
Nume:  
 <INPUT TYPE="TEXT" NAME="NUME">
<BR>
Adresa:
 <INPUT TYPE="TEXT" NAME="ADRESA" SIZE="70">
<P>Localitate:
 <INPUT TYPE="TEXT" NAME="LOCALITATE">
Cod: <INPUT TYPE="TEXT" NAME="COD" SIZE="5" MAXLENGTH="5">
Cod de client:<INPUT TYPE="PASSWORD" NAME="CODECLIENT" SIZE="8"></P>
<HR>
<B>Culoarea masinii:</B>
<SELECT NAME="TIPCULOARE">
 <OPTION VALUE="ROSU">Rosu</OPTION>
 <OPTION VALUE="VERDE">Verde</OPTION>
 <OPTION VALUE="ALBASTRU">Aalbastru</OPTION>
 <OPTION VALUE="GRISOARECE">Gri soarece</OPTION>
```

```
</SELECT>
<B>Numar de usi:</B>
<INPUT TYPE="RADIO" NAME="USI" VALUE="2">2
<INPUT TYPE="RADIO" NAME="USI" VALUE="3">3
<INPUT TYPE="RADIO" NAME="USI" VALUE="5">5
<P><B>Alte date:</B> <INPUT TYPE="CHECKBOX" NAME="ALTE" VALUE="CLIMA">Clima
<INPUT TYPE="CHECKBOX" NAME="ALTE" VALUE="ABS">ABS
<INPUT TYPE="CHECKBOX" NAME="ALTE" VALUE="ESP">ESP</P>
<HR>
Daca aveti alte comentarii va rugam sa le scrieti mai jos:<BR>
    <TEXTAREA    NAME="COMMENTS"    ROWS="3"    COLS="65"
WRAP="WRAP">Comentarii?</TEXTAREA>
<HR>
<INPUT TYPE="SUBMIT" VALUE="Trimite" NAME="trimite">
<INPUT TYPE="RESET" VALUE="De la inceput" NAME="reia">
</FORM>
</BODY>
</HTML>
```

Figure 34 Transmiterea datelor din formulare prin e-mail



Limbajul de programare VBScript

Limbajele de script ASP au un singur tip de dată, tipul `Variant`. Variabilelor de acest tip li se alocă 16 octeți şi pot stoca orice valoare de tipul `Integer`, `Long`, `String`, `Array` sau `Object`. Variabilele `Variant` sunt mai "mari" decât celelalte variabile întrucât trebuie să stocheze atât tipul cât şi valoarea corespunzătoare tipului.

Limbajul **VBScript** este un subset al limbajului **VBA** (**Visual Basic for Applications**) folosit de produsul Microsoft Office şi de către multe alte aplicații comerciale. **VBA** este, la rândul lui, un subset al limbajului **VB** (**Visual Basic**). Toate aceste limbaje au aproape aceleaşi cuvinte cheie, proprietăți şi funcții. Cea mai mare diferență între ele este aceea că **VB** este compilat în timp ce **VBScript** nu este.

VBScript este limbajul ASP implicit. La nivelul clientului se preferă înlocuirea lui cu **JavaScript** sau **JScript** întrucât codul acestor limbaje pot fi executate de majoritatea navigatoarelor.

Limbajul **VBScript** are un număr mare de funcții interne şi metode pentru majoritatea activităților de programare comune. Ultima lui versiune adaugă posibilități de căutare avansate şi suport pentru crearea de clase şi obiecte.

Cuvinte cheie

Codul script este o combinație de cuvinte cheie, de obiecte, de apeluri de funcții interne şi de apeluri de rutine scrise de utilizator. Cuvintele cheie sunt cuvinte recunoscute de analizor ca parte a limbajului **VBScript** şi au o semnificație predefinită. Este interzisă crearea unor variabile cu acelaşi nume. Acesta este motivul pentru care ele vor fi prezentate, în continuare, succint, într-o formă tabelară, în ordinea alfabetică.

Cuvânt cheie	Tip	Descriere
`Abs`	Funcție	Valoarea absolută a unui număr.
`Adunare (+)`	Operator	Adună două valori.
`And`	Operator	Operatorul boolean `ŞI`.
`Array`	Funcție	Creează un tablou `Variant`.
`Asc`	Funcție	Codul ASCII al primului unui caracter.
`Atn`	Funcție	Arctangenta unui număr.
`Atribuire (=)`	Operator	Atribuie o valoare unei variabile; în **VBScript** se mai poate folosi pentru testarea echivalenței.
`Call`	Instrucțiune	Apelează o subrutină sau o funcție.
`CBool`	Funcție	Întoarce un `Variant` de subtipul `Boolean`.
`CByte`	Funcție	Întoarce un `Variant` de subtipul `Byte`.
`CCur`	Funcție	Întoarce un `Variant` de subtipul `Currency`.
`CDate`	Funcție	Întoarce un `Variant` de subtipul `Date`.
`CDbl`	Funcție	Întoarce un `Variant` de subtipul `Double`.

Cuvânt cheie	Tip	Descriere
Chr	Funcție	Caracterul corespunzător codului ASCII specificat printr-un număr întreg.
CInt	Funcție	Întoarce un Variant de subtipul Integer.
Class	Obiect	Obiectul întors atunci când se creează o definiție de clasă cu instrucțiunea Class.
Class	Instrucțiune	Creează o clasă; trebuie specificat numele clasei, proprietățile şi metodele.
Clear	Metodă	Reinițializează obiectul Err.
CLng	Funcție	Întoarce un Variant de subtipul Long.
Concatenare (&)	Operator	Concatenează două şiruri.
Const	Instrucțiune	Creează o constantă.
Cos	Funcție	Cosinusul unui număr.
CreateObject	Funcție	Creează o variabilă de tipul obiect.
CSng	Funcție	Întoarce un Variant de subtipul Single.
CStr	Funcție	Întoarce un Variant de subtipul String.
Date	Funcție	Întoarce un Variant de subtipul Date.
DateAdd	Funcție	Întoarce o dată sau o oră prin adunarea unei date sau ore cu o constantă care reprezintă zile, luni, ani, secunde etc.
DateDiff	Funcție	Întoarce diferența calculată în luni, săptămîni, zile, ani, minute, secunde sau ore între două date sau ore.
DatePart	Funcție	Întoarce un interval de tipul zi, lună, an etc. al unei date.
DateSerial	Funcție	Întoarce o dată specificată prin an, lună şi zi.
DateValue	Funcție	Întoarce un Variant de subtipul Date corespunzător unui parametru dată şir; transformă un şir într-o dată.
Day	Funcție	Întoarce ziua datei specificate.
Description	Proprietate	Întoarce descrierea erorii stocate în obiectul Err.
Dictionary	Obiect	Un obiect ce stochează o colecție de perechi (cheie, valoare).
Dim	Instrucțiune	Declară o variabilă.
Do ... Loop	Instrucțiune	Permire reluarea unui grup de instrucțiuni; realizează o ciclare.

Cuvânt cheie	Tip	Descriere
`Empty`	Valoare	Valoarea unui `Variant` neinițializat.
`Eqv`	Operator	Identic cu `And` boolean.
`Erase`	Instrucțiune	Reinițializează un tablou.
`Err`	Obiect	Stochează informația de eroare.
`Eval`	Funcție	Evaluează script-ul primit drept parametru ca pe o expresie. La un moment dat se poate evalua o singură expresie, pentru evaluarea mai multor expresii, vezi instrucțiunea `Execute`.
`Execute`	Metodă	Execută o căutare într-o "expresie de căutare" a şirului primit ca argument.
`Execute`	Instrucțiune	Execută una sau mai multe linii de cod transferate prin parametri. Liniile pot fi separate prin caracterul **:** sau de <**Enter**>; script-ul astfel executat poate accesa variabile globale, dar poate fi executat numai în contextul procedurii în curs de execuție.
`ExecuteGlobal`	Instrucțiune	Execută una sau mai multe linii de cod transferate prin parametri. Liniile pot fi separate prin caracterul **:** sau de <**Enter**>; script-ul astfel executat rulează în context global, poate accesa variabile globale şi poate fi apelat din orice alt loc în script.
`Exit`	Instrucțiune	Face ieşirea dintr-o subrutina, funcție sau un bloc de cod care este ciclat.
`Exp`	Funcție	**e** ridicat la puterea x (e^x).
`False`	Valoare	Valoarea booleană fals a lui 0.
`FileSystemObject`	Obiect	Un obiect care permite lucrul cu discul.
`Filter`	Funcție	Întoarce o submulțime a unui tablou şir pe baza condițiilor specificate în parametru.
`FirstIndex`	Proprietate	Întoarce distanța, în caractere, a primului caracter al unui obiect `Match` întors dintr-o căutare într-o "expresie de căutare". Aceasta înseamnă că şirul căutat a fost găsit în cel destinație din poziția indexului poantat de proprietatea `FirstIndex`.
`Fix`	Funcție	Întoarce numărul trunchiat la o valoare întreagă.
`For Each ... Next`	Instrucțiune	Permite parcurgerea articolelor unui obiect colecție sau a unui tablou.
`For ... Next`	Instructiune	Permite reluarea unei secvențe de cod de un număr fixat de ori.

Cuvânt cheie	Tip	Descriere
FormatCurrency	Funcție	Formatează valorile valutare pe baza criteriilor specificate.
FormatDateTime	Funcție	Formatează datele sau orele pe baza unui criteriu de formatare.
FormatNumber	Funcție	Formatează numerele pe baza unui criteriu de formatare.
FormatPercent	Funcție	Formatează numerele sau expresiile la formatul procentual.
Function	Instrucțiune	Defineşte începutul unei funcții.
GetLocale	Funcție	Întoarce identificatorul local al calculatorului pe care se rulează script-ul.
GetObject	Funcție	Întoarce o referință la un obiect încărcat dintr-un fişier. Se vor furniza numele fişierului şi opțional aplicația prin care s-a creat sub forma Excel.WorkSheet sau Word.Application.
GetRef	Funcție	Întoarce un pointer la o funcție ce poate fi legat la un eveniment DHTML.
Global	Proprietate	Determină dacă într-o căutare de "expresie de căutare" se vor prinde toate aparițiile sau numai prima.
HelpContext	Proprietate	Setează sau întoarce o valoare HelpContextID reprezentând ID-ul unui topic într-un fişier de Help.
HelpFile	Proprietate	Setează sau întoarce numele fişierului de Help asociat unui anumit obiect.
Hex	Funcție	Întoarce o valoare numerică în formatul hexazecimal.
Hour	Funcție	Întoarce ora dintr-o expresie de tipul timp.
If ... Then ... Else	Instrucțiune	Permite ramificarea codului pe baza valorii unei expresii. Dacă expresia ia valoarea True se executa secțiunea Then, dacă expresia ia valoarea False se execută codul din secțiunea Else.
IgnoreCase	Proprietate	Determină căutarea în expresii obişnuite ținând cont sau nu de scrierea cu litere mari sau mici.
Înmulțire (*)	Operator	Înmulțeşte două numere.
Imp	Operator	Operatorul logic implică.
Împărțire (/)	Operator	Împarte două valori numerice.

Cuvânt cheie	Tip	Descriere
`Împărţire întreagă (\)`	Operator	Împarte două valori numerice şi converteşte rezultatul într-un `Integer`.
`Initialize`	Eveniment	Apare atunci când o clasă **VBScript** este instanțiată.
`InputBox`	Funcție	Cere utilizatorului introducerea unei informații; funcționează numai la client.
`InStr`	Funcție	Poziția, în caractere, corepunzătoare primei apariții ale unui subşir într-un şir.
`InStrRev`	Functie	Poziția, în caractere, corepunzătoare ultimei apariții ale unui subşir într-un şir.
`Int`	Funcție	Partea întreagă a unui număr. Dacă acesta este negativ întoarce primul număr negativ întreg mai mic sau egal decât el. Dacă argumentul este şir va întoarce numărul întreg corespunzător.
`Is`	Operator	Test de echivalență. Întoarce `True` dacă doi pointer-i poantează la acelaşi obiect.
`IsArray`	Funcție	Întoarce `True` dacă argumentul este de subtipul `Array`.
`IsDate`	Funcție	Întoarce `True` dacă argumentul este de subtipul `Date`.
`IsEmpty`	Funcție	Întoarce `True` dacă argumentul este un `Variant` cu valoarea `Empty`.
`IsNull`	Funcție	Întoarce `True` dacă argumentul este un `Variant` cu valoarea `Null`.
`IsNumeric`	Funcție	Întoarce `True` dacă argumentul este un număr sau se poate converti la un număr.
`IsObject`	Funcție	Întoarce `True` dacă argumentul este un `Variant` de subtipul `Object`.
`Join`	Funcție	Acceptă un tablou de şiruri şi întoarce un şir separat printr-un delimitator specificat. `Join` este opusul lui `Split`.
`LBound`	Funcție	Întoarce indexul de jos al argumentului tablou.
`LCase`	Funcție	Întoarce şirul argument convertit în litere mici.
`Left`	Funcție	Întoarce un subşir al argumentului şir plecând de la primul caracter până la unul specificat într-un parametru index.
`Len`	Funcție	Întoarce lungimea, în caractere, a şirului argument.

Cuvânt cheie	Tip	Descriere
`Length`	Proprietate	Întoarce lungimea unui obiect `Match` găsit într-o "expresie de căutare".
`LoadPicture`	Funcție	Întoarce un obiect imagine încărcat de pe disc.
`Log`	Funcție	Logaritmul natural al unui număr (ln(x)).
`LTrim`	Funcție	Întoarce un şir fără spațiile din partea stângă a şirului argument.
`Matches`	Colecție	Colecție de obiecte `Match` rezultate din căutarea încheiată cu succes a unor expresii obişnuite.
`Mid`	Funcție	Întoarce un subşir al argumentului şir dintr-un punct de plecare specificat şi cu lungime dată.
`Minute`	Funcție	Întoarce minutul, ca şi întreg în domeniul 0-59, al orei specificate.
`Mod`	Operator	Restul împărțirii a două valori întregi.
`Month`	Funcție	Întoarce luna, ca şi întreg în domeniul 1-12, pentru data specificată.
`MonthName`	Funcție	Întoarce numele lunii corespunzătoare argumentului întreg.
`MsgBox`	Funcție	Afişează o fereastră de dialog Windows care conține un mesaj, titlu, simbol grafic sau butoane. Întoarce o constantă care defineşte butonul selectat de utilizator. Se foloseşte în script-uri client.
`Not`	Operator	Operatorul de negare a valorii unei expresii.
`Nothing`	Valoare	Valoarea unui obiect neinițializat.
`Now`	Funcție	Întoarce data şi ora curentă.
`Null`	Valoare	O "valoare" care are semnificația de fără valoare; este diferiă de zero, de şirul nul, de `Empty`, şi de `Nothing`.
`Number`	Proprietate	Proprietate ce stochează numărul erorii unui obiect `Err`.
`Oct`	Funcție	Întoarce valoarea octală a unui număr.
`On Error`	Instrucțiune	Permite controlul erorilor în timpul execuției programului.

Cuvânt cheie	Tip	Descriere
`Option Explicit`	Instrucțiune	Forțează declararea variabilelor. Atunci când este activă, la compilarea codului **VBScript**, la întâlnirea unui simbol necunoscut, dă un mesaj de eroare. Implicit, opțiunea nu este activă, în acest caz la întâlnirea unui simbol necunoscut **VBScript** va crea o nouă variabilă.
`Or`	Operator	Operatorul `SAU` logic.
`Pattern`	Proprietate	Se foloseşte pentru setarea şirului pentru o căutare obişnuită.
`Private`	Instrucțiune	Permite crearea unei variabile, funcții sau subrutine private (la nivel de script).
`PropertyGet`	Instrucțiune	Întoarce valoarea unei proprietăți dintr-un obiect clasă.
`PropertyLet`	Instrucțiune	Dă valoarea unei proprietăți dintr-un obiect clasă.
`PropertySet`	Instrucțiune	Dă valoarea unei proprietăți obiect dintr-un obiect clasă.
`Public`	Instrucțiune	Permite crearea unei variabile, funcții sau subrutine publice.
`Raise`	Metodă	Permite ridicarea unei erori. Rezultatul ridicării unei erori depinde de activarea lui `On Error Resume Next` şi de către starea curentă de prelucrare a erorilor.
`Randomize`	Instrucțiune	Inițializarea generatorului de numere aleatoare.
`ReDim`	Instrucțiune	Schimbă dimensiunea unui tablou. Are restricții în cazul tablourilor multidimensionale.
`RegExp`	Obiect	Reprezintă un obiect "expresie de căutare". Permite realizarea unor căutări complexe în şirul sursă.
`Rem`	Instrucțiune	Permite crearea unui comentariu. Apostroful are acelaşi efect.
`Replace`	Metodă	Înlocuieşte una sau mai multe apariții ale unui subşir într-un şir cu un alt subşir dintr-o expresie de căutare obişnuită.
`Replace`	Funcție	Înlocuieşte una sau mai multe apariții ale unui subşir într-un şir cu un alt subşir.
`RGB`	Funcție	Transformă trei valori de culori distincte în valoarea Long corespunzătoare de culoare.
`Ridicare la putere (^)`	Operator	Ridică un număr a la puterea x (a^x).

Cuvânt cheie	Tip	Descriere
`Right`	Funcție	Întoarce numărul de caractere specificat din dreapta şirului.
`Rnd`	Funcție	Un număr aleator între 0 şi 1.
`RTrim`	Funcție	Întoarce un şir fără spațiile din partea dreaptă a şirului argument.
`Scădere (-)`	Operator	Întoarce un număr înmulțit cu -1 sau diferența a două numere.
`ScriptEngine`	Funcție	Întoarce un şir cu numele motorului curent de execuție a script-urilor.
`ScriptEngineBuildVersion`	Funcție	Întoarce un şir cu numărul de versiune a motorului curent de execuție a script-urilor.
`ScriptEngineMajorVersion`	Funcție	Întoarce un şir cu versiunea majoră a motorului curent de execuție a script-urilor.
`ScriptEngineMinorVersion`	Funcție	Întoarce un şir cu versiunea minoră a motorului curent de execuție a script-urilor.
`Second`	Funcție	Întoarce secunda din ora specificată.
`Select ... Case`	Instrucțiune	Permite execuția mai multor regiuni de cod pe baza evaluării unei expresii.
`SetLocale`	Funcție	Setează ID-ul local pentru script. Se foloseşte pentru a transforma ieşirea pentru date, ore şi valută în forma specificată de ID.
`Sgn`	Funcție	Semnul unui număr.
`Sin`	Funcție	Sinusul unui unghi dat în radiani.
`Source`	Proprietate	Întoarce sursa erorii. În cazul unui script de server conține numele paginii cu eroarea.
`Space`	Funcție	Întoarce un şir umplut cu un număr specificat de spații.
`Split`	Funcție	Descompune un şir într-un tablou de subşiruri pe baza unui delimitator. `Split` este opusul lui `Join`.
`Sqr`	Funcție	Rădăcina pătrată a unui număr ($\sqrt{x}$).
`StrComp`	Funcție	Compară două şiruri şi întoarce -1 dacă este mai mic decât al doilea, 0 în caz de egalitate şi 1 dacă primul este mai mare decât al doilea şir.
`String`	Funcție	Întoarce un şir format prin repetarea unui caracter de un anumit număr de ori.

Cuvânt cheie	Tip	Descriere
`StrReverse`	Funcție	Întoarce un şir în care s-au inversat caracterele față de margini.
`Sub`	Instrucțiune	Defineşte începutul unei subrutine.
`Tan`	Funcție	Tangenta unui unghi dat în radiani.
`Terminate`	Eveniment	Apare înainte de distrugerea unei clase **VBScript**. Se foloseşte pentru distrugerea obiectelor alocate dinamic.
`Test`	Metodă	Execută o căutare într-o "expresie de căutare".
`Time`	Funcție	Întoarce ora curentă a sistemului cu precizia de 1 secundă.
`Timer`	Funcție	Întoarce numărul de secunde trecute de la miezul nopții.
`TimeSerial`	Funcție	Întoarce o valoare de tipul timp specificată prin oră, minut şi secundă.
`TimeValue`	Funcție	Întoarce timpul din argument. Dacă valoarea conține şi o dată atunci numai timpul va fi întors.
`Trim`	Funcție	Întoarce şirul argument fără spațiile din stânga şi dreapta lui.
`True`	Valoare	Valoarea booleană `True` (adevărat). În **VBScript** este numărul -1.
`TypeName`	Funcție	Întoarce numele intern **VBScript** al unei variabile de tipul scalar sau obiect.
`UBound`	Funcție	Indexul superior al unui tablou.
`UCase`	Funcție	Întoarce un şir convertit la litere mari.
`Value`	Proprietate	Întoarce textul unui obiect `Match` rezultat în urma unei căutări cu succes a unei expresii obişnuite.
`VarType`	Funcție	Întoarce o constantă sau, pentru tablouri, o combinație de constante care reprezintă tipul intern **VBScript** al unei variabile.
`Weekday`	Funcție	Acceptă un argument dată şi întoarce un număr între 1 şi 7 corespunzător zilei din săptămînă a argumentului.
`WeekdayName`	Funcție	Acceptă un argument dată şi întoarce un şir pentru data corespunzătoare zilei din săptămînă a argumentului.

Cuvânt cheie	Tip	Descriere
`While ... Wend`	Instrucțiune	Realizează ciclarea condiționată a unui grup de instrucțiuni atât timp cât condiția de evaluare ia valoarea `True`.
`With ... End With`	Instrucțiune	Ține o referință locală la un obiect atât timp cât se fac operații multiple cu acesta.
`Xor`	Operator	Operatorul boolean sau exclusiv.
`Year`	Funcție	Întoarce un întreg ce reprezintă anul datei specificată ca argument.

Variabile

Limbajul **VBScript** are un singur tip de variabilă, tipul **`Variant`**, care poate stoca însă valori scalare, tablouri şi pointer-i la obiecte. Implicit, **VBScript** presupune că orice simbol care nu este cuvânt cheie este variabilă. Acesta este motivul pentru care apar foarte des probleme dacă în cod s-a greşit scrierea unui nume de variabilă. **VBScript** va accepta acest nume, fără a genera un mesaj de eroare şi va crea o variabilă nouă. De exemplu, dacă în loc de **`Maxim`**, accidental, scriem `Maxin`, **VBScript** va crea o variabilă nouă cu numele **`Maxin`**. Pentru a preveni apariția acestor erori se va folosi comanda **`Option Explicit`** la începutul codului script, apoi instrucțiunea `Dim` pentru declararea tuturor variabilelor din program. Dacă `Option Explicit` este prezentă, **VBScript** va genera o eroare de compilare la întâlnirea unui simbol necunoscut. Astfel, numele de variabile scrise greşit pot fi găsite deja în faza de proiectare a codului.

Denumirea de variabile scalare se foloseşte în cazul şirurilor şi al numerelor. Subtipurile `Variant` pentru variabilele scalare sunt: `Boolean`, `Integer`, `Long`, `Single`, `Double`, `Date`, `Currency` şi `String`.

Pentru a crea o **variabilă scalară** numele acesteia se defineşte folosind instrucțiunea `Dim`, apoi numelui de variabilă i se pot atribui valori. Iată un exemplu:

```
Dim x
x=1
document.Write x
```

Pentru ca să puteți vedea efectele codului puteți crea o pagină HTML cu următorul cod:

```
<HTML>
<HEAD>
<TITLE>Cod client 1</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Dim x
            x=1
            document.Write x
      </SCRIPT>

</BODY>
```

La deschiderea paginii cu navigatorul IE pe ecran se va afişa numărul `1`, adică valoarea atribuită variabilei `x`.

Tablourile permit stocarea unor liste de valori scalare sau de valori de pointer. Practic, tabloul nu stochează o listă de valori, ci un pointer la prima poziție a colecției în memorie. Atunci când se creează un tablou, acestuia i se va aloca suficient spațiu în memorie pentru a stoca toate valorile. **VBScript** suportă lucrul cu **tablouri dinamice**, adică tablouri a căror dimensiune poate fi modificată după creare. Atunci când se face redimensionarea, noul spațiu pentru tablou se alocă într-o nouă regiune de memorie, apoi vechiul tablou este copiat în cel nou. Din acest motiv redimensionarea unui tablou durează mult în comparație cu crearea unui tablou de dimensiune fixă. Tablourile pot fi create în două feluri. Un caz este cel în care dimensiunea lui este fixă, iar celălalt este cel în care dimensiunea tabloului este variabilă:

```
Dim a(10) 'se creeaza un tablou cu 11 elemente
```

sau

```
Dim a
a=Array()
Redim a(10)
```

Accesarea elementelor de tablou se face prin index, astfel:

```
a(0)=1
a(1)=4
a(2)=7
```


Implicit, **VBScript** creează tablourile cu primul index 0, deci instrucțiunea `Dim a(10)` creează un tablou cu 11 elemente. Domeniul valid al indexurilor este determinat de limitele tabloului. Limita de jos, care în **VBScript** este întotdeauna 0, respectiv cea de sus se pot determina prin funcțiile `LBound(a)`, respectiv `UBound(a)`. Următoarea secvență de cod poate fi rulată în IE:


```
Dim a(10)
document.Write "Limita de jos =" & LBound(a) & "<BR>"
document.Write "Limita de sus =" & UBound(a) & "<BR>"
```

Numărul total de articole dintr-un tablou poate fi găsit cu formula: `(UBound(a)-LBound(a)+1)`. În continuare, se prezintă o aplicație în care numărul de elemente de tablou se alocă dinamic, apoi se afișează limitele noului tablou şi numărul lui de elemente.

```
<HTML>
<HEAD>
<TITLE>Cod client 3</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Dim a, nrelem
            a=Array()
            ReDim a(12)
            Document.Write "Limita de jos =" & LBound(a) & "<BR>"
            Document.Write "Limita de sus =" & UBound(a) & "<BR>"
            nrelem=(UBound(a)-LBound(a)+1)
            Document.Write "Numarul de elemente = " & nrelem & "<BR>"
      </SCRIPT>
</BODY>
```

`Variant` poate stoca şi un pointer la obiect. Dacă un obiect este creat pe server cu funcția `Server.CreateObject`, **VBScript** alocă spațiul pentru stocarea datelor lui şi întoarce un pointer la zona alocată (`CreateObject` simplu se va folosi dacă obiectul va fi creat pe client). Variabila obiect stochează acel pointer, la fel ca în cazul unui tablou când se stochează un pointer la primul

element de tablou. Pentru ca **VBScript** să poată face diferență între un pointer la obiect şi o valoare numerică trebuie folosit cuvântul cheie `Set` atunci când se creează variabila obiect. De exemplu:

```
Dim obDictionar
Set obDictionar=CreateObject("Scripting.Dictionary")
```

Prin folosirea funcției `IsObject` se poate determina dacă o variabilă stochează un obiect. După crearea variabilei obiect proprietățile şi metodele ei se accesează prin folosirea punctului. De exemplu pentru adăugarea unor articole în dicționar se foloseşte metoda `Add` astfel:

```
obDictionar.Add cheie, valoare
```

ASP va distruge toate variabilele locale la terminarea paginii. Totuşi, este o strategie de programare bună ca variabilele obiect să fie setate la valoarea `Nothing` după cum urmează:

```
Set obDictionar=Nothing
```

Astfel, memoria utilizată pentru stocarea obiectului va fi eliberată şi va putea fi utilizată în alte scopuri.

În continuare se prezintă un exemplu de lucru funcțional în IE (nu este nevoie de server pentru rularea lui; dacă se doreşte rularea pe server, în locul lui `CreateObject` se va folosi `Server.CreateObject`).


```
<HTML>
<HEAD>
<TITLE>Cod client 4</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Dim obDictionar, O
            Set obDictionar=CreateObject("Scripting.Dictionary")

            obDictionar.Add 1, "unu"
            obDictionar.Add 2, "doi"

            for each O in obDictionar
                  Document.Write obDictionar(O) & "<BR>"
            next

      if IsObject(obDictionar) then
                  Set obDictionar=Nothing
      end if

      </SCRIPT>
</BODY>
```

Subrutine şi funcţii

VBScript permite plasarea codului în blocuri cu nume denumite generic **rutine**. Unele limbaje au un singur tip de rutină numită funcție. **VBScript** are două: **funcția** şi **subrutina**. Funcțiile sunt rutine care întorc o valoare în timp ce subrutinele sunt rutine care nu întorc valoare. Deseori, o anumită secvență de cod poate fi scrisă atât prin folosirea subrutinelor cât şi prin folosirea funcțiilor. În continuare se va scrie o subrutină şi o funcție care realizează aceeaşi acțiune de concatenare a două şiruri.

În cazul subrutinei pentru concatenarea a două şiruri avem doi parametri de intrare, pe `sir1` şi

pe `sir2`, şi un singur parametru de ieşire, pe `sirrasp`, care va conține rezultatul concatenării:

```
Sub ConSirS(sir1, sir2, sirrasp)
      sirrasp=sir1 & sir2
End Sub
```

O variantă mai utilă este funcția următoare:

```
Function ConSirF(sir1, sir2)
      ConSirF=sir1 + sir2
End Function
```

Acțiunea este aceeaşi, de concatenare a celor două şiruri, însă aici funcția va întoarce ca valoare rezultatul concatenării. Observați că în cazul funcției codul este scris diferit, în locul lui `&` folosindu-se +. În această variantă dacă argumentele sunt şiruri ele se vor concatena, iar dacă sunt numere ele se vor aduna.

Apelul funcțiilor şi al subrutinelor se scrie diferit. În cazul subrutinelor sunt posibile următoarele variante de scriere:

```
Call ConSirS(TSir(0), TSir(1), SirRasp)
ConSirS TSir(0), TSir(1), SirRasp
```

iar în cazul funcției:


```
SirRasp=ConSirF(1,2)
```

Tipurile argumentelor sunt o problemă serioasă în **VBScript** mai ales atunci când se scriu funcții. Un programator bun trebuie să verifice dacă tipurile sunt compatibile, iar în cazul în care nu sunt să folosească funcții de conversie pentru a rezolva incompatibilitățile. O soluție pentru problema propusă se prezintă în funcția `ConSirFOK()`.

```
<HTML>
<HEAD>
<TITLE>Cod client 5</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
      Dim  TSir, SirRasp

      Sub ConSirS(sir1, sir2, sirrasp)
            sirrasp=sir1 & sir2
      End Sub

      Function ConSirF(sir1, sir2)
            ConSirF=sir1 + sir2
      End Function

      Function ConSirFOK(sir1, sir2)
            If varType(sir1) <> vbString Then
                  sir1=CStr(sir1)
            End If
            If varType(sir2) <> vbString Then
                  sir2=CStr(sir2)
            End If
            ConSirFOK=sir1 + sir2
      End Function

      TSir=Array("unu","doi")
```

```
      Call ConSirS(TSir(0), TSir(1), SirRasp)
      Document.Write "Concatenarea lui " & TSir(0) & " cu " & TSir(1) & " da✂
 -> " & SirRasp & "<BR>"
      SirRasp=ConSirF(1,2)
      Document.Write "Concatenarea lui 1 cu 2 da -> " & SirRasp & "<BR>"
      SirRasp=ConSirFOK(1,2)
      Document.Write "Concatenarea lui 1 cu 2 da -> " & SirRasp & "<BR>"

      </SCRIPT>
</BODY>
```

VBScript permite crearea de **rutine recursive**. O rutină se numeşte recursivă dacă se apelează pe ea însăşi până când o anumită condiție de părăsire a apelului recursiv devine adevărată. Un exemplu de calcul al factorialului se prezintă în continuare.

```
<HTML>
<HEAD>
<TITLE>Cod client 6</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
      Function Fact(n)
            If (n > 0) Then
                  Fact=n*Fact(n-1)
            Else
                  Fact=1
            End If
      End Function

      Document.Write "Fact(7) = " & Fact(7) & "<BR>"
      </SCRIPT>
</BODY>
```


Operatori VBScript

Valorile stocate în variabile rezultă în urma evaluării unor expresii. Expresiile sunt formate din secvențe de operatori şi operanzi. Următoarele grupe de operatori sunt predefiniți în **VBScript**:

- aritmetici;
- de comparație;
- de concatenare;
- logici.

Operatori aritmetici

Semnificație	Simbol	Exemplu	Remarcă
adunare	+	a+b	a şi b de orice tip numeric.
scădere	-	a-b	a şi b de orice tip numeric.
înmulțire	*	a*b	a şi b de orice tip numeric.
împărțire	/	a/b	a şi b de orice tip numeric .
împărțire întreagă	\	a\b	Înainte de împărțire a şi b sunt rotunjite la valori întregi, rezultatul este tot un întreg.
ridicare la putere	^	a^b	a poate fi orice număr, b numai unul întreg.
modulo	Mod	a Mod b	a şi b se rotunjesc la întregi după care se calculează restul împărțirii lui a la b.

Operatori de comparaţie

Se folosesc pentru compararea expresiilor.

Semnificaţie	Simbol	Exemplu	Remarcă
mai mic	<	a<b	Expresia din exemplu ia valoarea `True` dacă `a < b`, în caz contrar ia valoarea `False`.
mai mic sau egal	<=	a<=b	Expresia din exemplu ia valoarea `True` dacă `a <= b`, în caz contrar ia valoarea `False`.
mai mare	>	a>b	Expresia din exemplu ia valoarea `True` dacă `a > b`, în caz contrar ia valoarea `False`.
mai mare sau egal	>=	a>=b	Expresia din exemplu ia valoarea `True` dacă `a >= b`, în caz contrar ia valoarea `False`.
egal	=	a=b	Expresia din exemplu ia valoarea `True` dacă `a = b`, în caz contrar ia valoarea `False`.
diferit	<>	a<>b	Expresia din exemplu ia valoarea `True` dacă `a <> b`, în caz contrar ia valoarea `False`.

Dacă în exemplele de mai sus, una dintre variabilele `a` sau `b` are valoarea `Null`, expresia de comparaţie ia şi ea valorea `Null`. Variabilele comparate trebuie să fie de tip numeric.

Operatori de concatenare

Se folosesc în cazul şirurilor şi au ca efect alăturarea a două şiruri. Valoarea întoarsă este un nou şir care conţine şirurile operanzi alăturaţi.

Semnificaţie	Simbol	Exemplu	Remarcă
concatenare	&	a&b	`a` şi `b` trebuie să fie şiruri, altfel se convertesc în şiruri automat.
concatenare	+	a+b	Dacă ambele variabile sunt de tipul şir, operatorul de adunare are efect de concatenare.

Operatori logici

Matematicianul britanic George Boole a inventat un sistem matematic bazat pe valorile de adevăr, adevărat (`True`), fals (`False`) şi funcţiile `ŞI` (**AND**), `SAU` (**OR**) şi `NU` (**NOT**). Funcţiile booleene sunt prezentate prin tabele de adevăr. Tabelele de adevăr prezintă toate posibilităţile combinaţiilor de intrare (sunt scrise pe fond gri în tabelele ce urmează), corespunzătoare celor de ieşire, pentru o funcţie booleană. În **VBScript**, operatori cu aceeaşi scriere sunt folosiţi pentru operaţiile logice şi pe biţi. Operatorii pe biţi tratează operanzii sub forma unui vector de biţi şi nu ca un singur număr. Ei acţionează asupra celor N biţi (care sunt reprezentarea binară a valorilor operanzilor) folosind funcţii boolene pentru a produce câte un rezultat la nivelul fiecăruia dintre cei N biţi. Dacă operanzii sunt valori logice, notaţiile folosite în tabelele de adevăr pentru valorile logice sunt cele de `True` şi `False`. Dacă operatorii sunt pe biţi, notaţia folosită în tabelele de adevăr este aceea de 0 pentru `False` şi de 1 pentru `True`.

Semnificație	Simbol	Exemplu	Remarcă
ŞI logic sau pe biți	And	a > b And b > c	Expresiile asupra cărora acționează operatorul sunt valori de adevăr; în acest caz se execută ŞI logic.
		a And b	a şi b sunt numere, operatorul ŞI va lucra la nivel de biți.
SAU logic sau pe biți	Or	a > b Or b> c	Expresiile asupra cărora acționează operatorul sunt valori de adevăr; în acest caz se execută SAU logic.
		a Or b	a şi b sunt numere, operatorul SAU va lucra la nivel de biți.
NU logic sau pe biți	Not	Not (a > b)	NU logic.
		Not a	NU pe biți.

Tabela de adevăr pentru And **logic**
a şi b sunt valori de adevăr True sau False.

And	True	False	Null
True	True	False	Null
False	False	False	False
Null	Null	False	Null

Tabela de adevăr pentru And **pe biți**
a şi b sunt numere şi se lucrează cu reprezentarea binară - şiruri de biți (0, 1).

And	1	0
1	1	0
0	0	0

Tabela de adevăr pentru Or **logic**
a şi b sunt valori de adevăr True sau False.

Or	True	False	Null
True	True	True	True
False	True	False	False
Null	True	Null	Null

Tabela de adevăr pentru Or **pe biți**
a şi b sunt numere şi se lucrează cu reprezentarea binară - şiruri de biți (0, 1).

Or	1	0
1	1	1
0	1	0

Tabela de adevăr pentru Not **logic**
a este True sau False.

a	Not(a)
True	False
False	True
Null	Null

Tabela de adevăr pentru Not **pe biți**
a este număr şi se lucrează cu reprezentarea binară - şiruri de biți (0, 1).

a	Not(a)
1	0
0	1

Operatorii logici lucrează cu valori de adevăr logice şi produc un rezultat care este tot o valoare logică, astfel **True** And **True** = **True**. Operatorii pe biți, după cum am mai spus deja, folosesc descompunerea binară a valorilor numerice ale operanzilor şi acționează asupra pozițiilor

identice de biți din cei doi operanzi, modificând bitul corespunzător numai pentru acea poziție. Rezultatul lor este un număr. De exemplu, în cazul lui `15 And 9` după descompunerea binară a operanzilor avem `1101 And 1001 = 1001 = 9`.

Instrucţiuni de ramificare

Toate limbajele moderne de programare au instrucțiuni pentru execuția condiționată a codului.

If ... Then

Instrucțiunea `If ... Then` are mai multe variante, dar toate au la bază aceeaşi sintaxă, o condiție urmată de `Then`, apoi urmează codul de executat dacă condiția ia valoarea de adevăr `True`. Ultima linie a lui `If` este `End If`, aceasta marcând terminarea blocului asupra căruia acționează `If`-ul. Iată câteva exemple:

```
Dim i
i=7
If i=7 Then
      'orice grup de instructiuni care se executa numai daca i=7
End If
```

O variantă puțin mai complexă de `If` poate conține şi o porțiune de `Else`:

```
Dim i
i=7
If i=7 Then
      'un grup de instructiuni care se executa daca i=7
Else
      'un alt grup de instructiuni care se executa daca i<>7
End If
```


Cea mai complexă variantă este aceea în care apare şi **`ElseIf`**:

```
Dim i
i=2
If i=1 Then
      'un grup de instructiuni care se executa daca i=1
ElseIf i=2
      'un alt grup de instructiuni care se executa daca i=2
Else
      'un alt grup de instructiuni care se executa daca i<>1 si i<>2
      'adica daca primele doua conditii nu au fost adevarate (True)
End If
```

Select ... Case

Atunci când se formează structuri **`If ... ElseIf ...Else ...End If`** lungi este de preferat folosirea instrucțiunii **`Select ... Case`**. Aceasta permite selectarea unei opțiuni din mai multe variante. Ultima porțiune de cod se poate rescrie astfel:

```
Dim i
i=2
Select Case i
      Case 1
            'un grup de instructiuni care se executa daca i=1
      Case 2
            'un alt grup de instructiuni care se executa daca i=2
      Case Else
            'un alt grup de instructiuni care se executa daca i<>1 si i<>2
            'adica daca primele doua conditii nu au fost adevarate (True)
End Select
```

Atunci când fiecare variantă are o singură valoare posibilă **`Select ... Case`** nu ajută prea mult în comparație cu **`If ... ElseIf ...Else ...End If`**, există însă şi cazuri în care pentru mai multe valori diferite trebuie executată aceeaşi instrucțiune, aici, **`Select ... Case`** devine utilă. Fie secvența de cod:

```
If TipGrad = "Preparator" Or TipGrad = "Asistent" _
  Or TipGrad = "Sef lucrari" Or TipGrad = "Conferentiar" Then
      ' nu primesc salar de merit
Else
      ' primesc salar de merit
End If
```

Rescrierea ei cu **`Select ... Case`** face, cel puțin, citirea mai uşoară:

```
Select Case TipGrad
Case "Preparator", "Asistent", "Sef lucrari", "Conferentiar"
      ' nu primesc salar de merit
Case Else
      ' primesc salar de merit
End Case
```

Instrucţiuni de ciclare

Deseori trebuie să reluăm un grup de operații de mai multe ori. **VBScript** are instrucțiunea **`For Each ... Next`** pentru parcurgerea obiectelor din colecțiile ASP intrinseci. O denumire alternativă este cea de iterație printre obiectele colecției. Atunci când nu se lucrează cu obiecte, **VBScript** pune la dispoziția programatorului instrucțiuni de ciclare clasice care au un corespondent în majoritatea altor limbaje de programare. Pentru un număr de reluări fixat se va folosi ciclul **`For ... Next`**, atunci când ciclarea se face sub controlul unei expresii care ia valoarea de adevăr **`True`** sau **`False`** se vor folosi ciclurile **`While ... Wend`** şi **`Do ... While`**. `While ... Wend` se execută atât timp cât o condiție ia valoarea `True` , iar `Do ... While` se execută atât timp cât o condiție este `False` .

For ... Next

Dacă numărul de reluări este predeterminat ciclul `For ... Next` este cel mai potivit. Acesta foloseşte o variabilă pentru controlarea numărului de reluări. Programatorul trebuie să specifice numele variabilei de control, valoarea de început, valoarea de sfârşit şi, opțional, pasul de creştere a variabilei de control. Codul care se va relua trebuie plasat între instrucțiunile `For` şi `Next`. Forma generală a instrucțiunii este:

```
For <Variabilă de control> = <Valoare de început> To <Valoare de sfârşit> Step <Pas>
      ' instrucțiuni
Next <Variabilă de control>
```

Tradițional, programatorii folosesc numele de variabilă `i` (i întreg) pentru **<Variabilă de control>**. Ciclurile pot fi imbricate (între instrucțiunile unui ciclu se poate scrie un alt ciclu). Opțional, pentru creşterea clarității codului, se poate scrie numele variabilei de control şi după **Next**. Pasul implicit este de 1. Deoarece creşterea cu o unitate a unei valori se numeşte incrementare, uneori **<Pas>** mai este numit şi increment. Iată un exemplu:

```
<HTML>
<HEAD>
<TITLE>Cod client 7 - ciclul For ... Next</TITLE>
</HEAD>
<BODY>
```

```
      <SCRIPT Language="VBScript">
            Dim i
            For i = 1 To 5 Step 2
                  Document.Write "i = " & i & "<BR>"
            Next
      </SCRIPT>
</BODY>
```

La terminarea instrucțiunilor de reluat **VBScript** creşte valoarea lui **<Variabilă de control>** cu cea definită în **<Pas>**, această reluare continuînd până la atingerea **<Valoare de sfârșit>**. Valorile variabilei de control şi a pasului nu trebuie să fie neapărat pozitive după cum se observă din exemplul următor:

```
Dim i
For i = 7 To -1 Step -2
      Document.Write "i = " & i & "<BR>"
Next
```

While ... Wend

Instrucțiunea `While ... Wend` se execută atât timp cât instrucțiunea condițională următoare lui `While` este `True`. Forma generală este:

```
While <Condiție>
      'instrucțiuni
Wend
```


Iată un exemplu care este echivalentul primului exemplu de ciclu `For ... Next`:

```
<HTML>
<HEAD>
<TITLE>Cod client 8 - ciclul While ... Wend</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Dim i
            i=1
            While i <= 5
                  Document.Write "i = " & i & "<BR>"
                  i=i+1
            Wend
      </SCRIPT>
</BODY>
```

O altă aplicație a ciclului `While ... Wend` este:

```
Dim s
s = "a"
While s <> "abcdefg"
      Document.Write "s = " & s & "<BR>"
      s = s & Chr(Asc(Right(s,1))+1)
Wend
```

Ciclul `While ... Wend` este o variantă mai simplă de `For ... Next`, fiind păstrat în limbaj numai din motive de compatibilitate cu versiunile lui mai vechi.

Do ... While

Ciclul `Do ... While` este asemănător lui `While ... Wend`, dar condiția de reluare a ciclului poate fi scrisă atât înaintea, cât şi la sfârşitul ciclului. Apare astfel, posibilitatea de execuție cel puțin o singură data a ciclului. Iată formele generale:

```
Do While <Condiție>
        'instrucțiuni
Loop

Do
        'instrucțiuni
Loop While <Condiție>
```

Există şi o variantă în care **<Condiție>** nu se scrie ea fiind folosită în cazul ciclurilor infinite:

```
Do
        'instrucțiuni
Loop
```

Această ultimă variantă este utilă atunci când se execută pe server, unde valoarea `Server.ScriptTimeout` limitează execuția, în timp, a script-urilor.

Varianta **Do ... Loop While <Condiție>** va fi executată cel puțin o singură dată întrucât condiția testată pentru reluarea ciclului se află după instrucțiunile de reluat.

Operaţii cu şiruri

Limbajul Visual Basic are o gamă variată de funcții pentru manipulare a şirurilor. Limbajul **VBScript**, pe lângă faptul că moşteneşte aceste funcții, are şi câteva extensii foarte utile. **VBScript** poate căuta şi extrage subşiruri, compara şi descompune un şir într-un tablou de subşiruri cu ajutorul delimitatorilor. Deşi operațiile cu şiruri sunt mult simplificate prin funcții speciale, manipularea şirurilor presupune copierea unor blocuri continue de memorie dintr-o regiune de RAM în alta. Minimizarea acestor operații duce la creşterea vitezei de lucru a programului.

Funcţii obişnuite pentru manipularea şirurilor moştenite din Visual Basic

Funcțiile `Left`, `Right` şi `Mid` întorc partea stângă, dreaptă respectiv o secțiune de mijloc a şirului sursă. Pentru `Left` şi `Right` trebuie specificat numărul de caractere care se întorc. Pentru `Mid` se va specifica poziția de pornire şi opțional lungimea. Iată câteva exemple:

```
Left("Te salut!, 3)           ' intoarce "Te "
Left("Te salut!, 33)          ' intoarce "Te salut!"
Right("Te salut!", 3)         ' intoarce "ut!"
Right("Te salut!", 43)        ' intoarce "Te salut!"
Mid("Te salut!", 3, 3)        ' intoarce "sal"
Mid("Te salut!", 3)           ' intoarce "salut!"
Mid("Te salut!", 3, 23)       ' intoarce "salut!"
```

Funcțiile `Instr` şi `InstrRev` permit găsirea poziției unui subşir într-un şir. Ambele funcții întorc poziția de început a primului caracter din subşir în şir.

```
Instr("Te salut!", "sal")     ' intoarce 4
```

Ambele funcții accepta o poziție de început şi un parametru de comparație ce specifică dacă în comparație se ține cont de scrierea cu litere mari sau mici. Pentru valoarea 0 (`VbBinaryCompare`) comparația se face ținând cont de scrierea cu litere mari sau mici, iar pentru valoarea 1 (`vbTextCompare`) nu. Spre deosebire de `Instr`, `InstrRev` începe căutarea de la capătul şirului.

```
Instr(1,"Te salut!", "sal", vbBinaryCompare)    ' întoarce 4
Instr(1,"Te salut!", "Sal", vbBinaryCompare)    ' întoarce 0
Instr(1,"Te salut!", "sal", vbTextCompare)      ' întoarce 4
```

```
InstrRev("Te salut!", "sal",9 ,VbBinaryCompare) ' întoarce 4
InstrRev("Te salut!", "Sal",9 ,VbBinaryCompare) ' întoarce 0
InstrRev("Te salut!", "sal",9 ,vbTextCompare)   ' întoarce 4
```

Observați că ordinea parametrilor prin care este specificată poziția de plecare nu este aceeaşi, iar răspunsul este dat întotdeauna față de primul element din şir. Pentru ca `InstrRev` să parcurgă tot şirul, parametrul de început trebuie să fie lungimea şirului.

Expresia de căutare

VBScript a preluat din **JScript** o foarte puternică metodă de căutare şi găsire a expresiilor obişnuite, în comparație cu aceea oferită de `Instr`.

O expresie de căutare este formată dintr-un text împreună cu unele caractere speciale sau din comenzi care descriu modul în care se va face căutarea. Pentru inițalizarea unei căutări într-o expresie de căutare trebuie creat un obiect `RegExp` şi inițializată proprietatea `Pattern` după cum urmează:

```
Set expob = New RegExp
expob.Pattern = "ta"
```

Pot fi realizate căutări globale, care găsesc toate aparițiile, prin folosirea proprietății `Global`, iar pentru ignorarea scrierii cu litere mari şi mici se poate folosi proprietatea `IgnoreCase`. După setarea proprietăților obiectului `RegExp` se trece la căutare prin folosirea metodei `Text` sau `Execute.Test` ce realizează testarea existenței subşirului, în timp ce `Execute` determină poziția şi numărul lui de apariții. Rezultatul lui `Test` este `True` dacă subşirul apare în şirul de căutat, altfel întoarce `False`. În continuare se prezintă un exemplu de lucru cu `Execute`. Rezultatele sunt vizualizate în **Figura 35**.


```
<HTML>
<HEAD>
<TITLE>Cod client - Expresia obisnuita</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Dim s
            Dim expob
            Dim rezultate
            Dim unrezultat
            Dim i
            s = "Acesta este sirul - Expresie obisnuita - " & _
                "care va fi parcurs in cautare"
            Set expob = New RegExp
            expob.Pattern = "ta"
            expob.IgnoreCase = True
            expob.Global=True
            Set rezultate = expob.Execute(s)
            Document.Write "In urma verificarii sirului:" & "<BR>"
            Document.Write  "<PRE>&quot;" & s & "&quot;<BR>"
            Document.Write  " "
            For i=0 to Len(s)
                  Document.Write Right(i,1)
            Next
            Document.Write  "</PRE><BR>"
            For Each unrezultat In rezultate
                  Document.Write "Subsirul &quot;" & unrezultat.Value & _
                  "&quot; de lungime " & unrezultat.Length & _
                  " a fost gasit in pozitia " & _
                  unrezultat.FirstIndex & "<BR>"
            Next
      </SCRIPT>
```

```
</BODY>
```

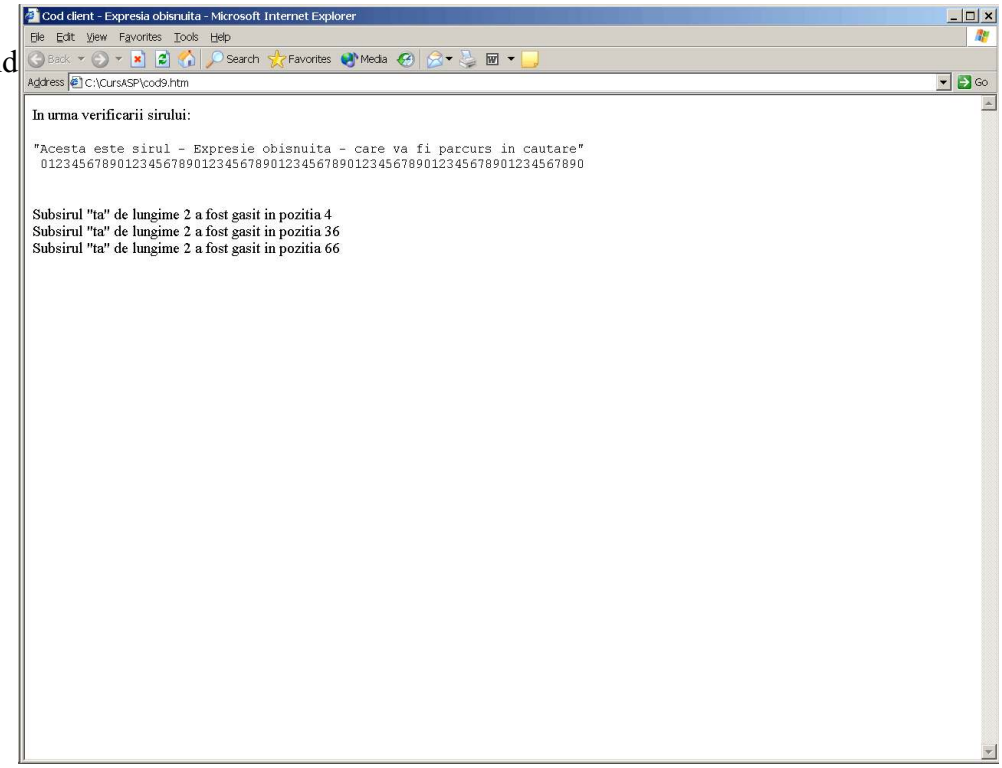


Figure 35 Căutarea folosind "Expresia de căutare"

Metoda `Execute` întoarce o colecție `rezultate`. Fiecare obiect din `rezultate` este un `rezultat` cu proprietățile `Value`, `Length` şi `FirstIndex` care definesc subşirul căutat, lungimea lui şi poziția lui în şir.

Puterea expresiei obişnuite constă în posibilitatea căutării unor caractere speciale. Tabelul următor prezintă câteva dintre caracterele speciale şi secvențele care pot fi folosite într-o expresie de căutare.

Caracter	Descriere
\	Specifică faptul că următorul caracter fie este unul special, fie este exact cel care s-a scris. Întrucât unele caractere au o semnificație specială ele trebuie "scoase" în afara semnificației speciale; aceasta se face precedând caracterul cu \, câteva dintre aceste caractere sunt: `( ) [ ] \*. $ .` Câteva dintre caracterele speciale sunt: `\b \B \d \D \n.`
^	Limitează comparația la începutul unui rând sau şir.
$	Limitează comparația la sfîrşitul unui rând sau şir.
*	Specifică caracterul din fața lui * de 0 sau de mai multe ori. De exemplu, pentru `ma*` se va găsi atât `mama` cât şi `mic`.
+	Specifică caracterul din fața lui + de unul sau de mai multe ori. De exemplu, pentru `ma+` se va găsi `mama`, dar nu se va găsi şi `mic`.

Caracter	Descriere
`?`	Specifică caracterul din fața lui `?` de 0 sau o singură dată. De exemplu, pentru `ma?` se va găsi `mama`, şi `mic`, dar nu se va găsi `maa`.
`.`	Specifică un singur caracter, oricare ar fi el (cu excepția celui de trecere la linie nouă).
`a|b`	Specifică pe `a` sau pe `b`.
`{n}`	Specifică repetarea unui caracter de `n` ori. De exemplu, `e{2}` găseşte perechea de `e` din `aceea` şi dar nu şi `e`-ul din `acel`.
`[abc]`	Specifică oricare caracter dintre parantezele drepte.
`[^abc]`	Specifică oricare dintre caracterele care nu sunt scrise între parantezele drepte.
`[a-z]`	Specifică un caracter din domeniul definit; în varianta `[a-z]` orice literă mică din alfabet.
`\b`	Specifică un delimitator de cuvânt cum ar fi spațiul, punctul, tabulatorul. De exemplu, `a\b` specifică pe al doilea `a` din `Ea este mama mea.`
`\B`	Specifică un text care nu se află lângă un delimitator de cuvânt cum ar fi spațiul, punctul, tabulatorul. De exemplu, `\Bm` specifică al doilea `m` din `Ea este mama mea.`
`\d`	Specifică orice caracter care este cifră, la fel cu `[0-9]`.
`\D`	Specifică orice caracter care nu este cifră, la fel cu `[^0-9]`.
`\n`	Specifică caracterul de trecere la linie nouă (`vbCrLf - newline`).

Clase VBScript

Terminologie

- *programare orientată pe obiecte* - conceptul de bază în această tehnologie de programare este aceea de obiect care este o structura de dată încapsulată cu mulțimea rutinelor, denumite metode, care pot opera cu datele respective. Manipulările datelor pot fi efectuate numai prin aceste metode care sunt comune tuturor obiectelor ce sunt instanțieri ale aceleiaşi clase. Fiecare clasă are o poziție în ierarhia (mulțimea claselor împreună cu relațiile între clase) claselor. Metodele unei clase pot fi transferate în josul ierarhiei unei subclase sau pot fi moştenite de la o superclasă.
- *clasă* - prototipul unui obiect într-un limbaj de programare orientat pe obiecte, analogul tipului derivat din limbajele procedurale. O clasă mai poate fi considerată şi o mulțime de obiecte care au o structură şi un comportament comun. Structura clasei este determinată de variabilele prin care se reprezintă starea obiectului din acea clasă, iar comportamentul clasei este dat de mulțimea metodelor asociate clasei.
- *obiect* - în programarea orientată pe obiecte este o instanță unică a unei structuri de date definită conform unui şablon dat de clasa din care face parte.

În **VBScript**, o clasă este formată din proprietăți şi cod care lucrează împreună formând o unitate de program distinctă. Majoritatea claselor conțin proprietăți, care pot fi citite sau scrise, şi

metode, care sunt funcţii sau subprograme ce pot fi apelate. Clasele pot conţine proprietăţi şi metode ascunse (private) sau vizibile (publice). O proprietate este o caracteristică a unui obiect, iar o metodă reprezintă o acţiune care poate fi efectuată de obiect. Clasa reprezintă un şablon. O variabilă de tipul clasă (uneori numită şi instanţiere) este un obiect care se creează pe baza şablonului, el având caracteristicile descrise de acesta.

Definirea membrilor dată

Când se atribuie o valoare unei proprietăţi din obiect, valoarea este stocată într-o variabilă internă care este accesibilă numai respectivului obiect. Un utilizator al obiectului poate manipula doar proprietăţile expuse. Astfel, datele specifice obiectului sunt protejate contra manipulărilor directe şi, eventual, greşite.

Primul pas este specificarea acestor variabile interne clasei care se mai numesc şi membri dată. De obicei, se declară câte o variabilă internă pentru fiecare dată de stocat în obiect respectând regulile:

- fiecare variabilă se declară folosind cuvântul cheie `Private` pentru a face variabilele disponibile numai în interiorul acelei clase;
- tradiţional, numele membrilor dată se prefixează cu `m` pentru a indica faptul că este vorba despre un membru al unei clase.

În exemplul care va fi prezentat acestea sunt:


```
Private mNume
Private mPrenume
Private mAdresa
Private dTelefoane
```


Implementarea proprietăţilor clasei

Al doilea pas constă în definirea proprietăţilor clasei. La fel ca şi în cazul membrilor dată, ideea stă în crearea unei proprietăţi pentru fiecare articol de informaţie stocată. Acest pas are două etape pentru fiecare proprietate:

- stabilirea mecanismului pentru întoarcerea valorii curente a proprietăţii şi
- un mecanism, opţional, pentru modificarea valorii proprietăţii.

Aceste "mecanisme" se implementează printr-un grup special de funcţii folosind cuvintele cheie `Property Get` şi `Property Set`.

Property Get

`Property Get` se foloseşte pentru întoarcerea valorii curente a unei proprietăţi. Formatul general pentru această proprietate este:

Public Property Get **numeproprietate**( )
    [*instrucţiuni*]
    numeproprietate = *mmembrudată*
End Property

Folosirea lui `Public` face ca această procedură sau proprietate să fie vizibilă pentru procese externe. **numeproprietate** este numele proprietăţii, iar *mmembrudată* este variabila membru dată care stochează valoarea curentă a proprietăţii. Dacă dorim, se pot adăuga instrucţiuni în plus procedurii, deşi în cazul lui `Property Get` este destul de rar. În exemplul care va fi prezentat acestea sunt:

```
Public Property Get Nume()
      Nume=mNume
End Property

Public Property Get Prenume()
      Prenume=mPrenume
End Property

Public Property Get Adresa()
      Adresa=mAdresa
End Property
```

Property Let

Cu excepția cazului în care doriți ca proprietatea să poată fi doar citită, trebuie definită o procedură care permite atribuirea unei noi valori proprietății. În acest scop se va folosi o procedură `Property Let` care are forma generală:

Public Property Let **numeproprietate**(***ValoareNouă***)
    [*instrucțiuni*]
    mmembrudată = ***ValoareNouă***
End Property

La fel ca şi mai înainte **numeproprietate** este numele proprietății, de asemenea, ***ValoareNouă*** este noua valoare pe care utilizatorul vrea s-o atribuie proprietății, iar *mmembrudată* este variabila membru dată a cărei valoare se va modifica. Şi aici pot fi adăugate instrucțiuni în plus. Acestea testează, de obicei, noua valoare înainte de a modifica membrul dată. În exemplul care va fi prezentat acestea sunt:


```
Public Property Let Nume(s)
      mNume=s
End Property

Public Property Let Prenume(s)
      mPrenume=s
End Property

Public Property Let Adresa(s)
      mAdresa=s
End Property
```

Crearea metodelor clasei

Orice obiect are şi un grup de metode care acționează asupra obiectului. În acest scop, în cadrul clasei, trebuie să definim proceduri `Sub` sau `Function` de tipul `Public` pentru ca să fie vizibile extern. În cazul exemplului care se va prezenta există procedură `Sub` cu numele `AdaugaTelefon` care adaugă un nou număr de telefon.

Definirea evenimentelor clasei

Clasele definite de utilizator suportă lucrul cu două evenimente `Initialize` şi `Terminate`.

Evenimentul Initialize

Acesta apare atunci când se realizează instanțierea obiectului de tipul clasă, adică atunci când se declară o variabilă obiect de tipul clasei după cum urmează:

```
Dim Pers1
Set Pers1 = New Persoana 'evenimentul Initialize
```

Pentru a folosi acest eveniment se adaugă clasei o procedură `Private` care foloseşte următorul

format:

Private Sub **Class_Initialize**
    [codul de initializare se pune aici]
End Sub

În cazul nostru, procedura care urmează inițializează, creează obiectul `Scripting.Dictionary` şi stochează o referință la acesta în `dTelefoane`.

```
Private Sub Class_Initialize
      Set dTelefoane = CreateObject("Scripting.Dictionary")
      Document.Write "Obiectul Scripting.Dictionary a fost initializat<BR>"
End Sub
```

Evenimentul Terminate

Acest eveniment apare atunci când toate referințele la obiectul de tipul clasă sunt setate la `Nothing`. Se foloseşte atunci când clasa a alocat dinamic memorie şi aceasta trebuie eliberată. Pentru a folosi acest eveniment se scrie, în clasă, o procedură `Private` de forma:

Private Sub **Class_Terminate**
    [*codul pentru terminare se pune aici*]
End Sub

În cazul aplicației care urmează se va elibera spațiul alocat pentru obiectul `Scripting.Dictionary` prin codul:

```
Private Sub Class_Terminate
      Set dTelefoane = Nothing
      Document.Write "Obiectul Scripting.Dictionary a fost descarcat.<BR>"
End Sub
```

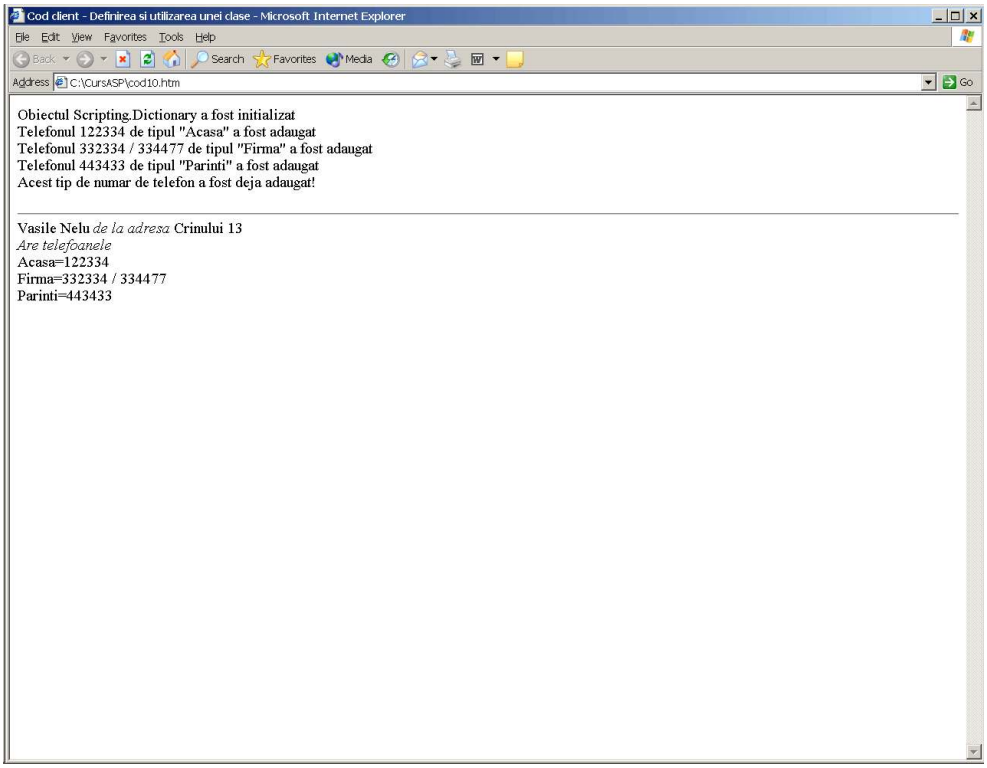
Aplicația care urmează defineşte clasa `Persoana` pentru a stoca numele, prenumele, adresa şi telefoanele unei persoane.

```
<HTML>
<HEAD>
<TITLE>Cod client - Definirea si utilizarea unei clase</TITLE>
</HEAD>
<BODY>
      <SCRIPT Language="VBScript">
            Option Explicit
            Class Persoana
                  Private mNume
                  Private mPrenume
                  Private mAdresa
                  Private dTelefoane

                  Private Sub Class_Initialize
                        Set dTelefoane = _
                        CreateObject("Scripting.Dictionary")
                        Document.Write  "Obiectul Scripting.Dictionary" & _
                        " a fost initializat<BR>"
                  End Sub

                  Private Sub Class_Terminate
                        Set dTelefoane = Nothing
                        Document.Write  "Obiectul Scripting.Dictionary" & _
                        " a fost descarcat<BR>"
```

```
                End Sub

                Public Property Let Nume(s)
                      mNume=s
                End Property
                Public Property Get Nume()
                      Nume=mNume
                End Property

                Public Property Let Prenume(s)
                      mPrenume=s
                End Property
                Public Property Get Prenume()
                      Prenume=mPrenume
                End Property

                Public Property Let Adresa(s)
                      mAdresa=s
                End Property
                Public Property Get Adresa()
                      Adresa=mAdresa
                End Property

                Public Sub AdaugaTelefon(Tip, Numar)
                      If dTelefoane.Exists(Tip) Then
                            Document.write "Acest tip de numar" & _
                            " de telefon a fost deja adaugat!<BR>"
                      Else
                            dTelefoane.Add Tip,Numar
                            Document.write "Telefonul " & Numar & _
                            " de tipul &quot;" & Tip & _
                            "&quot; a fost adaugat<BR>"
                      End If
                End Sub

                Public Property Get Telefoane()
                      Dim t
                      Dim s
                      For Each t In dTelefoane.Keys
                            s = s & t & "=" & dTelefoane(t) & "<BR>"
                      Next
                      Telefoane=s
                End Property
          End Class

          Dim Pers1
          Set Pers1=New Persoana

          Pers1.Nume="Vasile"
          Pers1.Prenume="Nelu"
          Pers1.Adresa="Crinului 13"
          Pers1.AdaugaTelefon "Acasa","122334"
          Pers1.AdaugaTelefon "Firma","332334 / 334477"
          Pers1.AdaugaTelefon "Parinti","443433"
          Pers1.AdaugaTelefon "Parinti","443433"
          Document.Write  "<BR><HR>"
          Document.Write  Pers1.Nume & " "
          Document.Write  Pers1.Prenume & " <I>de la adresa</I> "
          Document.Write  Pers1.Adresa & "<BR><I>Are telefoanele</I><BR>"
          Document.Write  Pers1.Telefoane
     </SCRIPT>
</BODY>
```


În **Figura 36** se prezintă modul în care IE afişează pagina corespunzătoare codului de mai sus.

Figure 36 Exemplu de utilizarea a claselor în VBScript cu HTML



Obiecte client

Navigatorul IE înțelege, după cum ați văzut deja, limbajul **VBScript**. Astfel, se pot implementa programe care să valideze date, să realizeze personalizarea paginilor în funcție de oră sau de versiunea navigatorului folosit, să realizeze comunicație între obiectele unei pagini etc. Un script foarte simplu poate afişa un mesaj într-o fereastră de dialog. Unul mai complicat, poate fi chiar o aplicație.

Inserarea script-ului în pagina Web

Instrucțiunile **VBScript** se scriu direct în pagina Web, putând fi citite de orice persoană care doreşte să vizualizeze codul sursă al paginii. Pentru aceasta se va folosi marcajul `SCRIPT` cu atributul `LANGUAGE`. Forma lui generală este:

```
<SCRIPT LANGUAGE="limbaj">
      aici se va scrie codul
</SCRIPT>
```

Pentru limbajul **VBScript** se va scrie `<SCRIPT LANGUAGE="VBScript">`, iar pentru limbajul **JavaScript** `<SCRIPT LANGUAGE="JScript">`.

Pentru procedurile `Sub` sau `Function` scrise între marcajele `<SCRIPT>` şi `</SCRIPT>` navigatorul va interpreta imediat instrucțiunile. În exemplul următor există o procedură `Sub` cu numele `Salut`. La nivelul ei se apelează `MsgBox` pentru afişarea mesajului "`Salutare ...`". Deoarece `Call Salut` este în afara procedurii, ea va fi executată imediat de navigator, iar mesajul va fi afişat imediat cum se deschide pagina.


```
<HTML>
<HEAD>
<TITLE>Script care saluta</TITLE>

</HEAD>
<BODY>
      <SCRIPT LANGUAGE="VBScript">
            Sub Salut
                  MsgBox "Salutare ...!"
            End Sub
            Call Salut
      </SCRIPT>
</BODY>
```

Ierarhia obiectelor de script

Un model de obiecte asigură un mecanism care face o aplicație să fie uşor programabilă. Ierarhia obiectelor de script este un model de obiecte care permite manipularea obiectele de pe suprafața paginii şi a navigatorului în care se vizualizează pagina. La nivelul cel mai de sus se află obiectul `Window`. Sub acesta urmează încă nivelurile prezentate în tabelul următor.

Obiect	Descriere
`Document`	Reprezintă documentul curent al ferestrei.
`Location`	Reprezintă URL-ul documentului curent.
`Histoy`	Reprezintă lista de istoric a navigatorului.
`Frame`	Reprezintă un tablou de cadre în fereastra curentă.

Obiect	Descriere
`Script`	Reprezintă script-ul în curs de utilizare.
`Navigator`	Reprezintă aplicația de navigare (de exemplu IE) folosită pentru afişarea paginii Web.

Obiectul Window

Obiectul `Window` este la rădăcina ierarhiei. Ca atare, el este implicit atunci când nu se foloseşte un prefix care să descrie obiectul. Astfel, de exemplu, `Window.name` şi `name` sunt echivalente. În cele ce urmează se prezintă câteva dinte proprietățile şi metodele semnificative pentru acest obiect.

Proprietăţile obiectului Window

Proprietățile obiectului `Window` definesc modul de apariție a ferestrei şi felul în care se raportează el la mulțimea cadrelor.

Proprietate	Descriere
`defaultStatus`	Determină textul implicit afişat în linia de stare a navigatorului (Textul care se afişează atunci când navigatorul aşteaptă date de intrare).
`name`	Întoarce numele ferestrei. Ferestrele primesc nume atunci când se folosesc cadre (cu atributul `NAME` în marcajul `FRAME`). O fereastra mai poate primi nume la deschiderea ei cu metoda `open` sau prin folosirea atributului `TARGET` în marcajul `A HREF`.
`opener`	Întoarce obiectul `window` care reprezintă fereastra care a deschis fereastra curentă.
`parent`	Întoarce obiectul `window` care reprezintă fereastra părinte a ferestrei curente (Părintele este cadrul în care fereastra curentă este stocată).
`self`	Întoarce un obiect `window` care reprezintă fereastra curentă.
`status`	Setează textul din linia de stare a navigatorului
`top`	Întoarce un obiect `window` care reprezintă cea mai de sus fereastră în tabloul cadrelor curente.

Metodele obiectului Window

Obiectul `Window` (`fereastră`) poate manipula fereastra navigatorului prin metodele:

Metodă	Descriere
`alert`	Afişează o fereastră de dialog pentru alertă.
`close`	Închide fereastra.

Metodă	Descriere
`confirm`	Afişează o fereastră de dialog cu un mesaj şi cu butoanele **OK**, **Cancel**. De exemplu, în cazul codului `rezultat = confim("Doriti sa transmiteti formularul server-ului?)`, dacă se apasă **OK**, `confirm` întoarce `True`, dacă se apasă **Cancel**, `confirm` întoarce `False`.
`navigate`	Face navigarea la fereastra specificată prin URL. Iată un exemplu: `navigate "http://www.utcluj.ro/".`
`setTimeOut`	Evaluează o expresie după trecerea unei durate exprimate în milisecunde. Forma generală este `setTimeOut(Cod, Milisecunde)`. Primul argument este obligatoriu şi este fie un pointer la funcție, fie un şir cu numele funcției de rulat. Al doilea argument este şi el obligatoriu fiind un întreg care specifică durata aproximativă, în milisecunde, până la evaluarea expresiei. Expresia este evaluată o singură dată. Iată un exemplu: `Call window.setTimeout("mutaText()", 50)`
`open`	Deschide o fereastră nouă. Sintaxa este: `window.open(URL, Destinație, Opțiuni)`, unde: - `URL`: Un şir care specifică URL-ul de afişat în noua fereastră; - `Destinație`: Numele noii ferestre deschise. - `Opțiuni`: O lista de valori separate prin virgulă care determină proprietățile noii ferestre după cum urmează: **`toolbar`** - Valoarea `yes` sau `no` determină afişarea sau nu, în noua fereastră, a barei cu instrumente; **`location`** - Valoarea `yes` sau `no` determină afişarea sau nu, în noua fereastră, a barei de locație; **`status`** - Valoarea `yes` sau `no` determină afişarea sau nu, în noua fereastră, a liniei de stare; **`menubar`** - Valoarea `yes` sau `no` determină afişarea sau nu, în noua fereastră, a barei de meniu; **`scrollbars`** - Valoarea `yes` sau `no` determină afişarea sau nu, în noua fereastră, a barelor de navigație; **`resizeable`** - Valoarea `yes` sau `no` determină dacă noua fereastră este sau nu redimensionabilă; **`width`** - Lățimea noii ferestre în pixeli; **`height`** - Înălțimea noii ferestre în pixeli; **`top`** - Poziția coordonatei Y a colțului stânga-sus al ferestrei. **`left`** - Poziția coordonatei X a colțului stânga-sus al ferestrei. Iată un exemplu: `window.open "http://www.yahoo.com", "Frame2", "toolbar=no, location=no"`

Metodă	Descriere
prompt	Afişează o fereastră de dialog prin care utilizatorul poate introduce o valoare. Sintaxa este: `prompt(mesaj,implicit)` - `mesaj` - mesajul afişat în fereastra de dialog; - `implicit` - valoarea iniţială afişată în cutia de text.

Evenimentele obiectului Window

Următoarele evenimente sunt suportate de obiectul `Window`:

Eveniment	Descriere
onLoad	Se declanşează atunci când conţinutul ferestrei este încărcat.
onUnload	Se declanşează atunci când conţinutul ferestrei este descărcat.

Evenimentele se definesc în marcajul `BODY` după cum se vede în exemplul următor:

```
<HTML>
<HEAD>
<TITLE>Script verifica daca un sir contine numai litere</TITLE>
</HEAD>
<BODY onLoad="Intra()" onUnload="Afara()">
      <SCRIPT LANGUAGE="VBScript">
            Sub Intra()
                  MsgBox "Ma bucur ca ai (re)venit."
            End Sub
            Sub Afara()
                  MsgBox "Pe curind ..." & Chr(10) & Chr(13) & _
                  "si numai cele bune pana atunci!"
            End Sub
      </SCRIPT>
</BODY>
```


Obiectul Document

Obiectul `Document` reprezintă pagina Web curentă încărcată în navigator. Este, de asemenea, container pentru obiectele din pagină, aici incluzându-se obiectele `Link`, `Forms` şi `Location`. Spre deosebire de obiectul `Window`, pentru a putea lucra cu proprietăţile şi evenimentele obiectului `Document` acestea trebuie precedate de cuvântul `document` (de exemplu, `document.title`).

Proprietăţile obiectului Document

Obiectul `Document` are proprietăţi care permit personalizarea prin program a modului de afişare a paginii. Aici sunt incluse culoarea fondului, culoarea hiperlegăturilor, titlul documentului etc.

Proprietate	Descriere

bgColor	Întoarce sau setează culoarea fondului documentului. Pentru setarea valorii, se foloseşte fie un şir, fie o valoare RGB, după cum urmează:
	`document.bgColor="white"`
	`document.bgColor="#FFFFFF"`

fgColor	Întoarce sau setează culoarea textului documentului.

Proprietate	Descriere
`lastModified`	Întoarce un şir cu data la care s-a făcut ultima modificare a documentului.
`linkColor`	Întoarce sau setează culoarea hiperlegăturilor din document.
`location`	Întoarce obiectul `Location`. Proprietățile lui permit specificarea unor părți (`protocol`, `host`, `pathname`) ale URL-ului corespunzătoare documentului.
`title`	Întoarce titlul documentului (textul cuprins în marcajul `TITLE`).
`vLinkColor`	Întoarce sau setează culoarea hiperlegăturilor vizitate din document.

Colecţiile obiectului Document

Obiectele colecției grupează mai multe obiecte de acelaşi tip în vederea unei referiri cât mai simple.

Colecție	Descriere
`all`	Întoarce o referință la colecția care stochează toate elementele documentului. Pentru a referi un element cu numele `Par1` se scrie `document.all("Par1")`.
`forms`	Întoarce obiectul `Forms` care este colecția tuturor formularelor din document. Modul de lucru cu acest obiect se va discuta în paragraful următor.
`links`	Întoarce obiectul `Link` care este colecția tuturor hiperlegăturilor din document. Fiecare element al acestei colecții este un obiect `Link`, care are, la rândul lui, proprietăți şi metode specifice fiecărei hiperlegături (de exemplu, `href`, `protocol`, `host` etc.).
`anchors`	Întoarce obiectul **`Anchors`** care este colecția tuturor elementelor de ancorare din document. **`document.anchors.length`** permite determinarea numărului de elemente de ancorare ale documentului.

În continuare urmează două exemple în care se folosesc colecțiile obiectului `Documents`. Primul arată modul în care se poate schimba conținutul unui paragraf şi culoarea de fond a documentului, iar al doilea arată modul în care se poate realiza plimbarea unui grup de texte pe suprafața documentului. Cele două aplicații vor fi salvate cu extensie de documente HTML şi vor putea fi rulate numai din IE.

```
<% Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY bgColor="lightgreen">
<P ID="p1">Acesta texul corespunzator marcajului de paragraf.</P>
</BODY>
</HTML>
<SCRIPT Language="VBScript">
   Dim Par1
   Dim i
   Set Par1 = document.all("p1")
```

```
   i=inputbox("Clic pe OK si fii atent la modificari ..." _
    ,"Modificare text paragraf si culoare de fond",document.bgColor)
   If i <> "" Then
      Par1.innertext = "Textul paragrafului a fost modificat din script."
      document.bgColor=i
   End If
</SCRIPT>

<% Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<P ID="p1" style="position: absolute; left: 640">Aceast text se va deplasa ✂
de la drepta la stanga pe ecran.</P>
</BODY>
<SCRIPT Language="VBScript">
   Dim C
   C=0
   a=array("primul text", "al doilea text","al treilea text")
   Call window.setTimeout("mutaText()", 50)
   function mutaText()
      Dim Poz
      Dim E1

      Set E1 = document.all("p1")
      Poz = E1.style.left
      If right(Poz, 2) = "px" Then
         Poz = left(Poz, Len(Poz) - 2)
      End If
      E1.style.left = Poz - 5
      If CLng(Poz) >= -200 Then
         Call window.setTimeout("mutaText()", 50)
      Else
        E1.innerText = a(C)
        E1.style.left = "640px"
            If C < UBound(a) Then
                  C=C+1
            Else
                  C=0
            End If

        Call window.setTimeout("mutaText()", 50)
      End If
   End Function
</SCRIPT>
```


Metodele obiectului Document

Documentele sunt considerate de către navigator obiecte statice. După ce conţinutul documentului a fost scris nu se mai pot face modificări ale lui (excepţie face conţinutul formularelor). Obiectul `Document` are mai multe metode, însă cea mai importantă dintre ele este `write`. Aceasta scrie un text în document. `document.write` va suprascrie orice document care există deja în fereastra navigatorului. Forma generală este:

```
document.write text
```

`text` este un şir care conţine textul care se doreşte a fi inserat. Utilizarea acestei metode are următoarele restricţii:

- textul este scris din poziţia în care metoda write apare în document;
- write trebuie să fie executată atunci când navigatorul analizează script-ul. Din acest motiv `write` se scrie fie în afara unei proceduri (dar în interiorul marcajului `SCRIPT`), fie într-o procedură care urmează să fie apelată;

- şirul de afişat poate conține marcaje HTML, iar navigatorul le va lua în considerare la afişare.

Un exemplu de utilizare a lui `write` se prezintă în continuare:

```
<HTML>
<HEAD>
<TITLE>Cod client - document.write</TITLE>
</HEAD>
<BODY>
      <P>O linie de text nevinovata ...
      <SCRIPT LANGUAGE="VBScript">
            Function OrasiData()
                  OrasiData=Time() & " - " & Date() & "."
            End Function

            Sub Salut()
                  Dim ora
                  ora=Hour(Now())
                  If ora < 12 Then
                        document.write "Buna dimineata!"
                  ElseIf ora < 16 Then
                        document.write "Buna ziua!"
                  Else
                        document.write "Noapte buna ..."
                  End If
            End Sub
      </SCRIPT>
      <P>Inca o linie de text la fel de nevinovata ...
      <SCRIPT LANGUAGE="VBScript">
            document.write "<P>Bine <B>ati</B> venit<BR>"
            Call Salut
            document.write "<P><P>Pagina a fost accesata la: " & _
             OrasiData & "<BR>"
      </SCRIPT>
</BODY>
```


Obiectul Navigator

Obiectul `Navigator` furnizează informații cu privire la navigatorul folosit pentru afişarea documentului. Câteva dintre proprietățile lui sunt:

Proprietate	Descriere
`appCodeName`	Întoarce numele codului folosit de navigator (de obicei "Mozilla").
`appName`	Întoarce numele aplicației navigator (de exemplu, Microsoft Internet Explorer).
`appVersion`	Întoarce numărul de versiune a navigatorului. Aici este inclus şi sistemul de operare.
`userAgent`	Întoarce agentul utilizator al navigatorului (deseori `appCodeName` combinat cu `appVersion`).

```
<HTML>
<HEAD>
<TITLE>Cod client -document.navigator</TITLE>
</HEAD>
<BODY>
      <SCRIPT LANGUAGE="VBScript">
            document.write navigator.appCodeName & "<BR>"
```

```
            document.write navigator.appName & "<BR>"
            document.write navigator.appVersion & "<BR>"
            document.write navigator.userAgent & "<BR>"
      </SCRIPT>
</BODY>
```

Obiectul Form

Formularele HTML pun accentul prelucrărilor pe server. Astfel, server-ul este cel ce trebuie să extragă datele, să le valideze, să le interpreteze şi, în final, să formuleze pagina Web de răspuns. În acest sens, codul **VBScript** poate uşura sarcina server-ului prin realizarea următoarelor prelucrări la client:

- validarea datelor înainte de trimiterea lor către server. Controalele formularelor pot fi accesate din **VBScript** şi trimiterea datelor către server poate fi întârziată până când valorile introduse sunt cele corecte;
- anumite calcule pot fi realizate pe client, fără a mai fi necesară trasmiterea acestora ca parametri server-ului.

Obiectul `Form` este o reprezentare a formularelor din pagina Web. Aceasta înseamnă că include fiecare obiect între marcajele `<FORM>`, `</FORM>`. Pentru pagini cu mai multe formulare colecția `Forms` reprezintă toate obiectele `Form` ale paginii.

Pentru referirea unui formular din cod sunt posibile două variante:

- se va folosi un indice cu obiectul `Forms`, de exemplu, `documents.Forms(0)` pentru a referi primul formular al paginii;
- se va folosi numele formularului aşa cum acesta este definit in atributul `NAME` al marcajului `FORM`. De exemplu dacă acesta ar fi `<FORM NAME="Frm1">`, formularul poate fi referit prin `document.Frm1`.

Proprietăţile obiectului Form

Majoritatea proprietăților obiectului `Form` corespund unor atribute ce pot fi incluse în marcajul `FORM`. Descrierea lor este făcută în tabelul următor.

Proprietate	Descriere
`action`	Întoarce sau setează URL-ul care se foloseşte pentru transferul formularului. Este echivalentul atributului `ACTION` din marcajul `FORM`.
`encoding`	Întoarce sau setează modul de codificare al formularului (de exemplu, `text/html`). Este echivalentul atributului `ENCODING` din marcajul `FORM`.
`method`	Întoarce sau seteză metoda folosită pentru transferul formularului. Este echivalentul atributului `METHOD` din marcajul `FORM`.
`target`	Setează fereastra destinație pentru formularul de răspuns. Este echivalentul atributului `TARGET` din marcajul `FORM`.

Următoarea secvență de cod setează proprietățile de mai sus pentru transferul unui formular:

```
document.Frm1.action ="http://www.server.ro/cgi-bin/program.exe"
document.Frm1.encoding="text/html"
document.Frm1.method="POST"
document.Frm1.target="FerRaspuns"
```

Transferul (submit) formularelor

Conținutul formularului este transferat server-ului dacă utilizatorul apasă un buton de transfer (`submit`). Această acțiune se poate realiza oricând din **VBScript** prin folosirea evenimentului `OnSubmit`. Evenimentul apare automat la transferul formularului către server şi permite definirea unei proceduri de tratare a evenimentului de transfer. Să presupunem că o funcție `Validare()`, care poate întroarce doar valorile `True` şi `False`, se doreşte a fi folosită pentru validarea conținutului formularului înainte de transfer. Procedura de tratare a evenimentului va fi scrisă astfel:

```
document.Frm1.OnSubmit="Validare()"
```

Dacă funcția `Validare()` întoarce valoarea `True` trasferul către server va fi realizat. Dacă valoarea întoarsă este `False` transferul este abandonat, utilizatorul rămânând cu formularul de transferat deschis pe ecran.

Manipularea controalelor unui formular

Obiectul `Form` are o proprietate, `elements`, care este colecția tuturor controalelor formularului. Colecția include, pe lângă controalele HTML, şi toate obiectele inserate cu marcajul `OBJECT` (acestea sunt, de exemplu, contoalele ActiveX). Colecția `elements` poate fi folosită pentru referirea unui control prin indice (primul control are indicele 0, al doilea 1 etc.) însă codul astfel scris este greu de citit. O altă variantă este folosirea numelui controlului definit prin atributul `NAME` sau, în cazul marcajului, `OBJECT` al atributului `ID`.

Proprietăţile controalelor

Între proprietățile diferitelor tipuri de controale există asemănări şi diferențe. Unele sunt comune la mai multe controale, iar altele sunt specifice unui anumit tip. Tabelul care urmează încearcă să prezinte o variantă simplificată a lor.

Proprietate	Descriere

`checked`	Pentru un buton de opțiune sau de validare, întoarce sau setează starea controlului (1 sau `True` înseamnă activ, 0 sau `False` dezactivat). De exemplu, codul care urmează se poate folosi pentru schimbarea stării unui buton de validare:
	`Set bv = document.Frm1.ButonValidare1`
	`bv. checked = Not bv. checked`

`defaultChecked`	Întoarce sau setează starea implicită a unui buton de validare sau de opțiune.
`defaultValue`	Întoarce sau setează valoarea implicită a unui control.
`form`	Întoarce obiectul `Form` din care face parte controlul.
`length`	În cazul unei liste, întoarce numărul de articole ale ei.
`name`	Întoarce sau setează numele controlului.
`value`	Întoarce sau setează valoarea curentă a unui control.
`selectedIndex`	În cazul unei liste, întoarce numărul de ordine corespunzător articolului selectat.

Metodele controalelor

Următoarele metode sunt accesibile pentru controalele formularelor:

Metodă	Descriere
`blur`	Anulează focusul controlului în cauză.
`click`	Clic pe control.
`focus`	Predă focalizarea (focus-ul) controlului în cauză.
`select`	Selectează conţinutul unui control de tipul `TEXT`, `TEXTAREA` sau `PASSWORD`.

Evenimentele controalelor

Porţiunile de cod sunt executate de navigator în ordinea scrierii lor. O altă variantă de scriere a codului constă în plasarea lui în proceduri de tratare a evenimentelor. În acest caz secvenţa de cod va fi rulată la fiecare apariţie a evenimentului declanşator, avantajele fiind următoarele:

- proceduri de tratare a evenimentelor pot fi create pentru majoritatea controalelor HTML;
- aceeaşi procedură poate trata evenimentele mai multor controale;
- se pot folosi limbaje de script diferite într-o singură pagină. De exemplu o procedură de tratare a evenimentului se poate scrie în **VBScript**, iar o alta în **JavaScript**.

Există două metode pentru specificarea unei proceduri de tratare a evenimetelor. Prima metodă foloseşte atributul **`EVENT`** al marcajului **`INPUT`**, iar a doua atributele **`FOR`** şi **`EVENT`** ale marcajului **`SCRIPT`**. În exemplul care urmează **`EVENT`** este numele evenimentului care se va prinde, adică **`ONCLICK`**. Fiind vorba de un buton, evenimentul corespunzător apăsării butonului se va executa la fiecare clic pe acesta.

```
<INPUT TYPE="BUTTON" VALUE="Verifica" ONCLICK="Verifica">
```

În această variantă se foloseşte **`SCRIPT`** cu atributele **`FOR`** şi **`EVENT`**, **`FOR`** specifică numele controlului, iar **`EVENT`** numele evenimentului care va fi prins de procedură.

```
<INPUT NAME="Nume" VALUE="Aici se va introduce numele" SIZE ="40">
<SCRIPT FOR="Nume" EVENT="onChange" LANGUAGE="VBScript">
      'Call Verifica
      alert("Valoarea s-a modificat")
</SCRIPT>
```

Această variantă este de preferat întrucât asigură o lizibilitate mai bună a codului. Tabelul care urmează prezintă un sumar al evenimentelor care pot fi prinse prin metodele descrise mai sus.

Eveniment	Descriere
`ONBLUR`	Declanşat atunci când controlul specificat pierde focus-ul (se face clic pe un alt control).
`ONCHANGE`	Declanşat atunci când valoarea controlului este modificată.
`ONCLICK`	Declanşat atunci când utilizatorul face clic pe control.

Eveniment	Descriere
ONFOCUS	Declanşat atunci când controlul primeşte focus-ul.
ONSELECT	Declanşat atunci când conţinutul controalelor TEXT, TEXTAREA sau PASSWORD este selectat.

Codul care urmează permite verificarea utilizării numai a literelor într-un şir de caractere introdus de utilizator.

```
<HTML>
<HEAD>
<TITLE>Script verifica daca un sir contine numai litere</TITLE>
</HEAD>
<BODY>
      <FORM NAME="Frm1">
            <P>
                  Cum te numesti?: <INPUT NAME="Nume" VALUE="Aici" & _
                   " se va introduce numele" SIZE ="40">

                  <SCRIPT FOR="Nume" EVENT="onChange" LANGUAGE="VBScript">
                        'Call Verifica
                        alert("Valoarea s-a modificat")
                  </SCRIPT>
            </P>
            <P>
            <INPUT TYPE="BUTTON" VALUE="VERIFICA" ONCLICK="Verifica">
            </P>
      </FORM>

      <SCRIPT LANGUAGE = "VBScript">
            Dim litere
            litere = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ "

            Function esteSirNull(s)
                  esteSirNull = len(s) = 0
            End Function

            Function esteLitera(c)
                  esteLitera = instr(1, litere, c, vbTextCompare) > 0
            End Function

            Function numaiLitere(s)
               Dim i
               For i = 1 to Len(s)
                  If Not esteLitera(mid(s,i,1)) Then
                        numaiLitere = False
                     Exit Function
                  End If
               Next
               numaiLitere = True
            End Function

            Sub Verifica()
                  Set ctrl = Document.Frm1.Nume

                  If esteSirNull(ctrl.Value) Then
                        alert("Numele NU poate fi vid!")
                        ctrl.focus
                  ElseIf Not numaiLitere(ctrl.Value) Then
                        alert("Numele poate contine numai: litere mici," & _
                         "litere mari si spatiu")
                        ctrl.focus
                  End If
```

```
            End Sub
      </SCRIPT>
</BODY>
```

Exemplu: Utilizarea evenimentelor asociate controalelor

Codul care urmează va fi stocat într-un fişier `scr1.html`. În acelaşi director vor mai fi stocate fişierele `ora.vbs`, `normal.jpg` şi `peste.jpg`. Fişierul `ora.vbs` este un fişier ce conţine un cod **VBScript** care va fi utilizat în pagina de Web, dar este stocat în afara paginii. Fişierele `normal.jpg` şi `peste.jpg` conţin câte o imagine ce va fi folosită pe post de buton. Imaginea `normal.jpg` va fi afişată în condiţii normale, iar imaginea `peste.jpg` se afişează pe locul primeia dacă poziţionăm cursorul peste aceasta (vezi **Figura 37**). Efectul este cel de avertizare a utilizatorului ca este poziţionat pe suprafaţa unei imagini (cu rol de buton).

Fişierul `ora.vbs`:

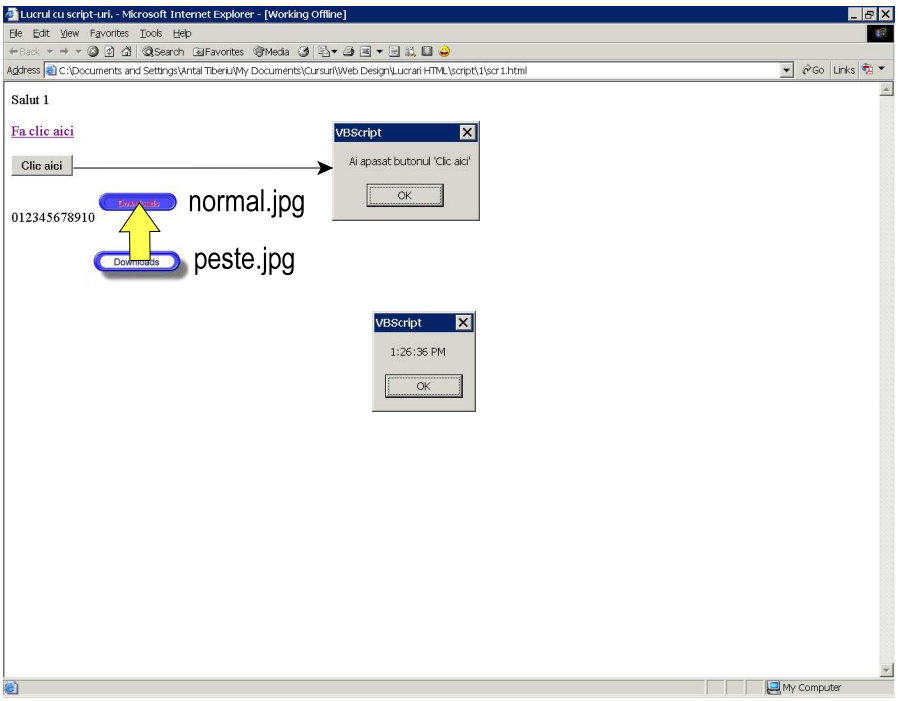
```
function ora()
      ora=time
end function

msgbox(ora())
```

Fişierul `scr1.html`:


```
<HTML>
<HEAD>
  <TITLE>Lucrul cu script-uri.</TITLE>
</HEAD>
<BODY>

<SCRIPT LANGUAGE="vbscript">
            document.write "Salut 1"
</SCRIPT>

<SCRIPT LANGUAGE="vbscript" SRC="ora.vbs">
</SCRIPT>

<P>
<A HREF="http://www.utcluj.ro" onclick="msgbox(Date())">Fa clic aici</A>
<P>

<INPUT NAME="nClic" TYPE="BUTTON" VALUE="Clic aici">
<SCRIPT LANGUAGE="VBScript" EVENT="onclick" for="nClic">
            msgbox "Ai apasat butonul 'Clic aici'"
</SCRIPT>

<P>
<SCRIPT LANGUAGE="vbscript">
            Dim i
            For i=0 to 10
                  document.write i
            Next
</SCRIPT>

<IMG SRC = "normal.jpg" name="img">
<SCRIPT LANGUAGE="VBScript" EVENT="onmouseover" FOR="img">
            document.img.src = "peste.jpg"
</SCRIPT>
<SCRIPT LANGUAGE="VBScript" EVENT="onmouseout" FOR="img">
            document.img.src = "normal.jpg"
</SCRIPT>

</BODY>
```

```
</HTML>
```

Figure 37
Aplicații cu Document



Codul este executat secvențial. Prima oară se afișează `"Salut 1"`, apoi se încarcă şi execută codul din `ora.vbs`, apoi codul care conține ciclul `For`. Celelalte secvențe de cod sunt rulate în funcție de acțiunile utilizatorului ce pot declanşa următoarele evenimente: clic pe hiperlegătură (**`onclick`**), clic pe butonul `"Clic aici"` (**`onclick`**) sau poziționare pe butonul `"Downloads"` (**`onmouseover`**), apoi poziționare în afara lui (**`onmouseout`**).

Exemplu: Aplicaţie pentru calculul dobânzii bancare

Programul conține procedura `Sub` cu numele `CalculDobanda`, un formular cu numele `frmDob` şi o procedura de tratare a evenimentului `onClick` a butonului cu numele `bCalcul`. În **Figura 38** se prezintă interfața paginii de calcul a dobânzii bancare, iar în **Figura 39** se observă modul de prezentare a rezultatelor.

```
<HTML>
<HEAD>
<TITLE>Formular pentru calculul dobanzilor</TITLE>

</HEAD>
<BODY>
      <SCRIPT LANGUAGE="VBScript">
            Sub CalculDobanda
                  Set frm = Document.frmDob
                  suma = frm.Suma.Value
                  Dobandaan=frm.DobandaAn.Value
                  luni = frm.Luni.Value
                  Dobandapeluna=suma*Dobandaan/12/100
                  Dobandatotala=Dobandapeluna*luni
                  Document.write "La suma de: " & suma & " lei<BR>"
                  Document.write "cu dobanda pe an de: " & Dobandaan &"%<BR>"
                  Document.write "timp de: " & luni & " luni<BR>"
                  Document.write "Dobanda pe luna este: <I>" _
                   & Dobandapeluna & "</I><BR>"
                  Document.write "iar dobanda totala este: <I>" _
```

```
                & Dobandatotala & "</I><BR>"
        End Sub
    </SCRIPT>

    <FORM NAME="frmDob">
        <H3>Calculator pentru dobanzi bancare</H3>
        Suma: <INPUT TYPE=TEXT NAME="Suma" VALUE="1000000"><BR>
        Dobanda pe an: <INPUT TYPE=TEXT NAME="DobandaAn" VALUE="29"> %<BR>
        Numarul de luni: <INPUT TYPE=TEXT NAME="Luni" VALUE="12"><BR>
        <INPUT TYPE=HIDDEN NAME="Rez" VALUE=""><BR>
        <INPUT TYPE=BUTTON NAME="bCalcul" VALUE="Calcul">

        <SCRIPT FOR="bCalcul" EVENT="onClick" LANGUAGE="VBScript">
            Set frm = Document.frmDob
            If frm.Suma.Value = "" Then
                alert ("Suma pentru care se va " & _
              "calcula dobanda nu poate fi 0!")
                frm.Suma.focus
            ElseIf frm.DobandaAn.Value = "" Then
                alert ("Dobanda anuala data de " & _
              "banca nu poate fi 0%!")
                frm.DobandaAn.focus
            ElseIf frm.Luni.Value = "" Then
                alert ("Numarul de luni pentru " & _
              "care se va calcula dobanda lunara nu poate fi 0!")
              frm.Luni.focus
            Else
              Call CalculDobanda
            End If
        </SCRIPT>
    </FORM>
</BODY>
```

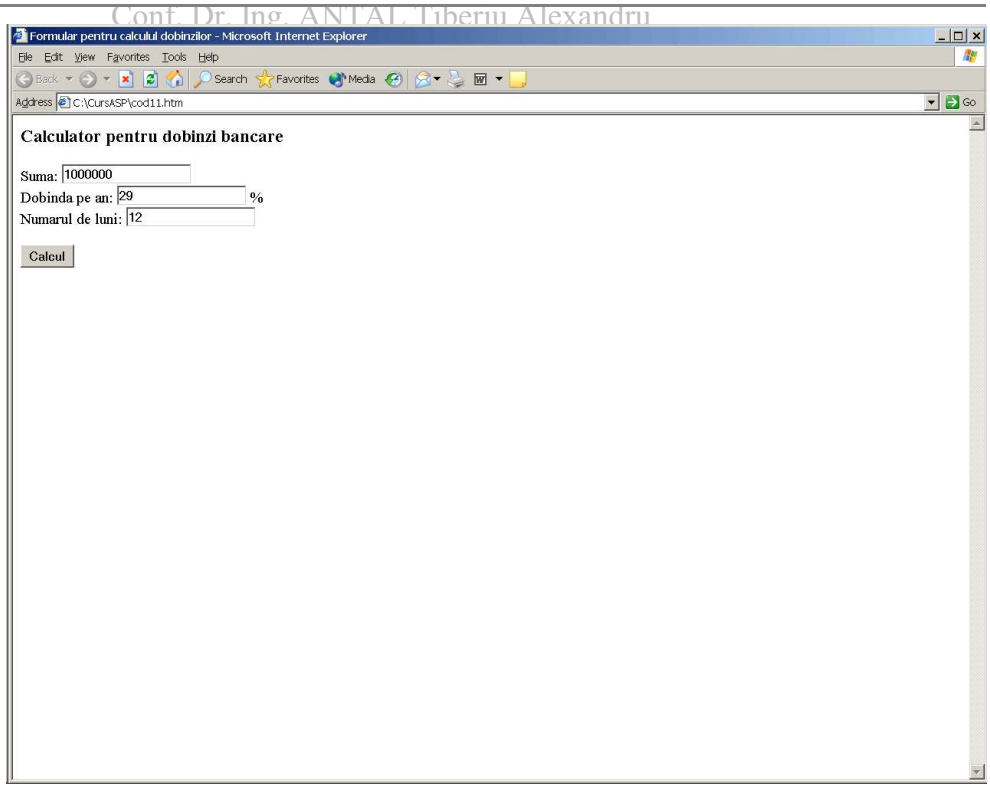



Figure 38 Interfața programului de calcul a dobânzilor

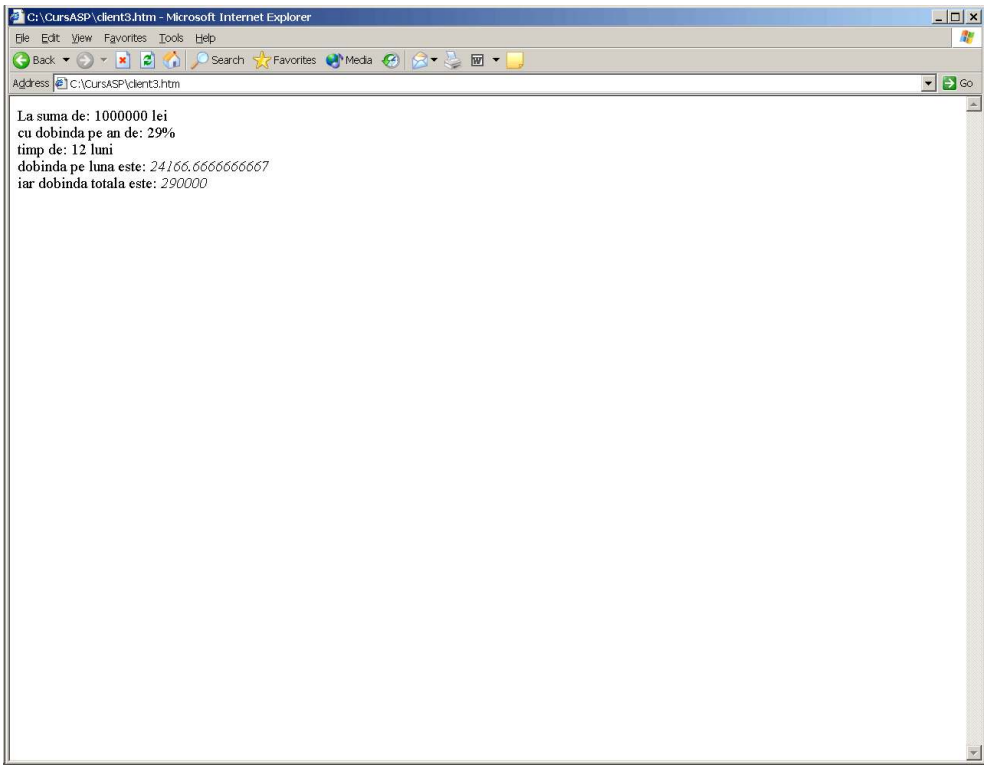


Figure 39 Rezultatele afişate de programul pentru calculul dobânzii

Exemplu: Accesul la conţinutul paginii de Web, din VBScript, folosind obiectul Document
Aplicaţia următoare este formată din fişierele `obWindow1_v2.html` şi `f1.jpg`, ce trebuie să fie stocate în acelaşi director. La deschiderea paginii de Web poza `"f1.jpg"`şi textul roşu `"linia de text 3"` se deplasează de la stânga la dreapta. Secvenţa de cod care realizează această mişcare se poate utiliza deci, atât pentru texte, cât şi pentru imagini.

Fişierul `obWindow1_v2.html`:

```
<HTML>
<STYLE TYPE="text/css">
      p.rosu {color:red;}
</STYLE>

<SCRIPT LANGUAGE="VBScript" SRC="muta.vbs">

function muta()
      Dim c
      Dim el

      set el = document.all("p1")
      c=el.style.left
      c=cint(left(c,len(c)-2))
      c=c+2
      if c < 200 then
            el.style.left=c&"px"
            call window.setTimeout("muta()",50)
      else
            el.style.left="1px"
            call window.setTimeout("muta()",50)
      end if
end function

function verifica()
```

```
      Dim s
      s="Numele: " & document.form1.text1.value & "<BR>"
      s=s+"Adresa: " & document.form1.text2.value & "<BR>"
      s=s+"radio 1: " & document.form1.radio1(0).checked  & "<BR>"
      s=s+"radio 2: " & document.form1.radio1(1).checked  & "<BR>"
      s=s+"radio 3: " & document.form1.radio1(2).checked  & "<BR>"
      if (document.form1.checkbox1(0).checked) then
            s=s+"check 1: " & document.form1.checkbox1(0).value  & "<BR>"
      end if
      if (document.form1.checkbox1(1).checked) then
            s=s+"check 1: " & document.form1.checkbox1(1).value  & "<BR>"
      end if
      if (document.form1.checkbox1(2).checked) then
            s=s+"check 1: " & document.form1.checkbox1(2).value  & "<BR>"
      end if

      if (document.form1.select1.selectedIndex =0) then
            s=s+"lista 1: " & document.form1.select1(0).value & "<BR>"
      end if
      if (document.form1.select1.selectedIndex =1) then
            s=s+"lista 2: " & document.form1.select1(1).value  & "<BR>"
      end if
      if (document.form1.select1.selectedIndex =2) then
            s=s+"lista 3: " & document.form1.select1(2).value  & "<BR>"
      end if

      document.write s
end function
```



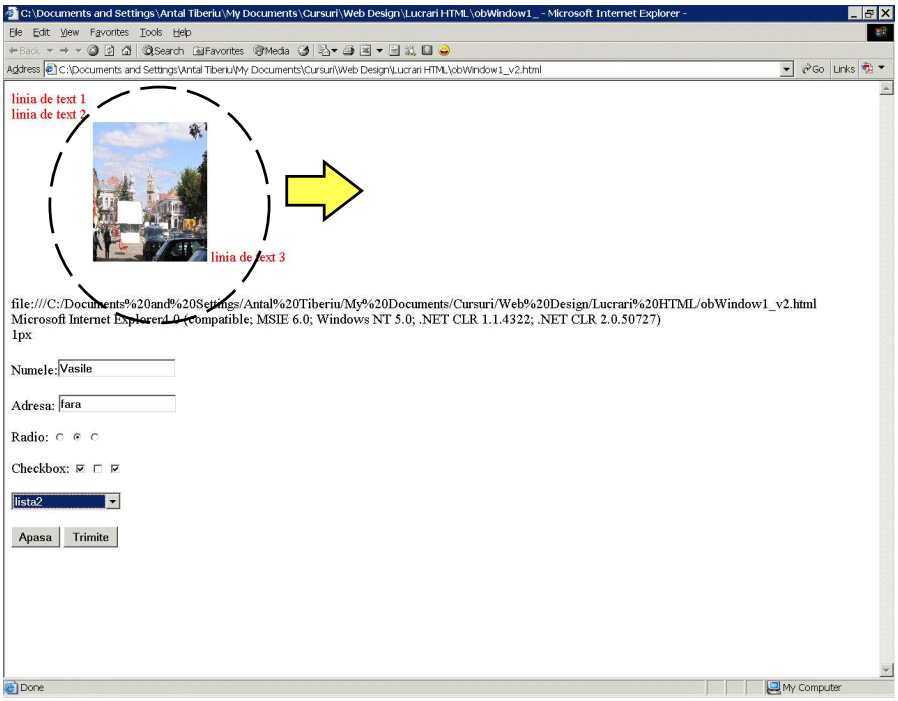
```
function Test()
      if len(document.form1.text1.value) = 0 then
            alert("Formularul nu poate fi trimis daca numele este vid!")
            Test = False
      else
          Test = True
      end if
end function
</SCRIPT>
```



```
<HEAD>
  <TITLE></TITLE>
</HEAD>
<BODY>
<P class=rosu>
linia de text 1
<BR>
linia de text 2
<BR>
<SPAN ID="p1" style="LEFT: 1px; POSITION: relative">
<IMG HEIGHT=200 WIDTH=164 ALT ="" SRC="./f1.jpg" BORDER=0>
linia de text 3
</span>

</P>
<BR>
<script language="VBScript">
 window.document.write document.location & "<BR>"
document.write window.navigator.appName, window.navigator.appVersion & "<BR>"
 document.write document.all("p1").style.left
 call window.setTimeout("muta()",50)
</script>

<FORM method=post name=form1 enctype="text/plain" action="mailto:unu@doi.ro"
>
      <P>Numele:<INPUT id=text1 name=text1> </P>
      <P>Adresa: <INPUT id=text2 name=text2></P>
      <P>Radio:
            <INPUT type=radio name=radio1 value=1>
```

```
            <INPUT type=radio name=radio1 value=2>
            <INPUT type=radio name=radio1 value=3></P>
      <P>Checkbox:
            <INPUT type=checkbox name=checkbox1 value=unu>
            <INPUT type=checkbox name=checkbox1 value=doi>
            <INPUT type=checkbox name=checkbox1 value=trei></P>
      <P>   <SELECT size=1 name=select1 style="WIDTH: 158px">
                  <OPTION value=l1 selected>lista1</OPTION>
                  <OPTION value=l2>lista2</OPTION>
                  <OPTION value=l3>lista3</OPTION>
            </SELECT></P>
<P> <INPUT type=button value=Apasa name=button1 onclick="verifica">
      <INPUT type=button value="Trimite" name=button2 onclick="trimite1">
</P>
</FORM>
```

Figure 40
Accesarea conținutului paginii de Web cu VBScript




```
<SCRIPT LANGUAGE="VBScript">
function trimite1()
      'document.form1.enctype="text/plain"
      'document.form1.action="mailto:unu@doi.ro"
      if Test() then
            document.forms(0).submit
      end if
end function
</SCRIPT>
</BODY>
</HTML>
```

La apăsarea butonului "`Apasa`" (vezi **Figura 40**) se vor extrage toare datele din formular şi se vor afişa în navigator sub forma:

```
Numele: Vasile
Adresa: fara
radio 1: False
radio 2: True
radio 3: False
check 1: unu
check 1: trei
```

```
lista 2: l2
```

Dacă se apasă butonul "Trimite" se verifică dacă numele este vid. Dacă este vid se afişează un mesaj de eroare altfel, conținutul formularului se trimite la o adresă de e-mail.

Marcajul <OBJECT>

O pagină Web poate conține şi controale ActiveX definite cu ajutorul marcajului HTML <OBJECT>. Controalele ActiveX sunt miniprograme care permit transformarea unei pagini Web statice în una dinamică şi interactivă. În acest fel pagina Web devine o aplicație care poate comunica cu utilizatorul. Marcajul <OBJECT> are o mulțime de atribute prin care se controlează tipul obiectului, aspectul lui, locația, înălțimea, lungimea etc. Câteva dintre aceste atribute se prezintă în tabelul următor.

Atribut	Descriere
ALIGN=*aliniere*	Setează alinierea obiectului în pagină. Valori de aliniere pot fi LEFT, RIGHT sau CENTER.
BORDER=*n*	Dacă obiectul este o hiperlegătură, atributul specifică grosimea chenarului.
CLASSID=*id*	O valoare care identifică implementarea obiectului.
DATA=*cale*	Identifică datele folosite de obiect.
HEIGTH=*n*	Înălțimea dorită pentru obiect.
HSPACE=*n*	Distanța între obiect şi orice text sau imagine la stânga sau la dreapta lui.
ID=*NumeObiect*	Un şir care identifică obiectul. Valoarea este folosită de programe pentru referirea obiectului.
NAME=*NumeControl*	Numele obiectului în formular. Când formularul este transmis această valoare se transmite împreună cu datele.
STANDBY=*mesaj*	Mesajul care se afişează atât timp cât obiectul se încarcă.
TYPE=*tip*	Tipul mediului de Internet al datelor.
USEMAP=*cale*	O imagine care se va folosi împreună cu obiectul.
VSPACE=*n*	Distanța între obiect şi orice text sau imagine deasupra sau sub acesta.
WIDTH=*n*	Lățimea dorită pentru obiect.

Dacă controlul ActiveX este înregistrat în registrul Windows atunci sintaxa pentru atributul CLASSID este *CLASSID:identificator_de_clasa*. Cel mai simplu mod de lucru pentru inserarea acestor controale într-o pagină Web este folosirea lui **Microsoft Visual InterDev**, care este parte a lui **Microsoft Visual Studio**. În continuare voi rescrie aplicația anterioară cu **Microsoft Visual InterDev**. Cu ajutorul acestei componente software voi insera două controale ActiveX de tipul TextBox pentru a putea afişa pe aceeaşi pagină rezultatele obținute în urma calculelor. De asemenea, voi folosi un tabel pentru a realiza o alinierea pe linii şi pe coloane mai estetică.

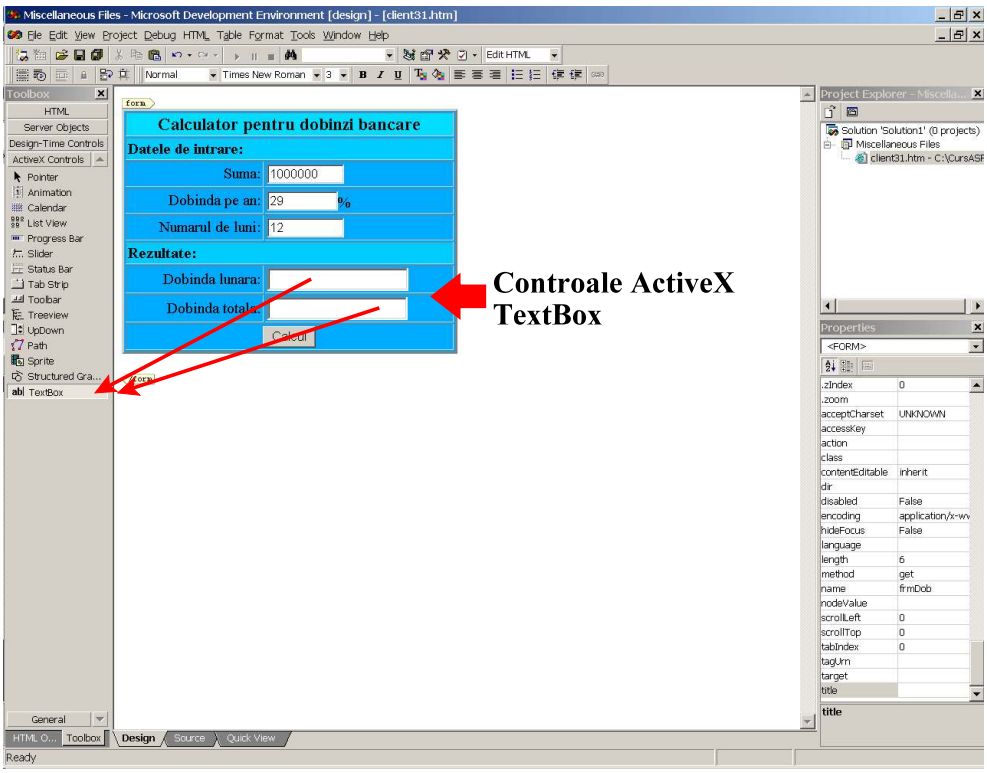


Figure 41
Noua interfața a programului de calcul a dobânzii

Pentru noua interfață trebuie să creați un tabel cu 2 coloane și 9 rânduri. În rândurile 7 și 8 veți insera câte un control ActiveX de tipul `TextBox`. Controlul din linia 7 va avea numele `dlunara`, iar cel din linia 8, `dtotala`. Codul HTML corespunzător noii paginii este prezentat în continuare, iar în **Figura 41**, se prezintă modul de afișare de a paginii de către IE.

NOTĂ

În codul următor, simbolul ✂ s-a folosit pentru a marca o întrerupere de linie, linia următoare acestui simbol se va scrie pe aceeași linie cu cea care conține simbolul.

```
<HTML>
<HEAD>
<TITLE>Formular pentru calculul dobinzilor</TITLE>
</HEAD>
<BODY>
      <SCRIPT LANGUAGE="VBScript">
            Sub CalculDobanda
                  Set frm = Document.frmDob
                  suma = frm.Suma.Value
                  Dobandaan=frm.DobandaAn.Value
                  luni = frm.Luni.Value
                  Dobandapeluna=suma*Dobandaan/12/100
                  Dobandatotala=Dobandapeluna*luni
                  Document.dtotala.Value = Dobandatotala
                  Document.dlunara.Value = Dobandapeluna
            End Sub
      </SCRIPT>

      <FORM NAME="frmDob">
            <TABLE  cellSpacing=2  cellPadding=2  width="50%"  border=3✂
BGCOLOR="#00aaff">
```

```
                  <TR>
                        <TD COLSPAN="2" ALIGN="middle" BGCOLOR="#00ddff">✄
<BIG><B>Calculator pentru dobinzi bancare</B></BIG> </TD>
                  </TR>
                  <TR>
                        <TD  COLSPAN="2"  ALIGN="left"  BGCOLOR="#00ccff">✄
<B>Datele de intrare:</B> </TD>
                  </TR>
                  <TR>
                        <TD ALIGN="right">Suma:</TD>
                  <TD> <INPUT NAME="Suma" VALUE="1000000" SIZE=7 ></TD>
                  </TR>
                  <TR>
                        <TD ALIGN="right">Dobanda pe an:</TD>
                        <TD><INPUT NAME="DobandaAn" VALUE="29" SIZE=6>%</TD>
                  </TR>
                  <TR>
                        <TD ALIGN="right">Numarul de luni:</TD>
                        <TD><INPUT NAME="Luni" VALUE="12" SIZE=7></TD></TR>
                  <TR>
                        <TD COLSPAN="2"✄
 BGCOLOR="#00ccff"><B>Rezultate:</B></TD>
                  </TR>
                  <TR>
                        <TD ALIGN="right"> Dobanda lunara:</TD>
                        <TD>
      <OBJECT style="LEFT: 0px; WIDTH: 184px; TOP: 0px; HEIGHT: 30px"
      classid=clsid:8BD21D10-EC42-11CE-9E0D-00AA006002F3 name=dlunara>
      <PARAM NAME="VariousPropertyBits" VALUE="746604571">
      <PARAM NAME="BackColor" VALUE="2147483653">
      <PARAM NAME="ForeColor" VALUE="2147483656">
      <PARAM NAME="MaxLength" VALUE="0">
      <PARAM NAME="BorderStyle" VALUE="0">
      <PARAM NAME="ScrollBars" VALUE="0">
      <PARAM NAME="DisplayStyle" VALUE="1">
      <PARAM NAME="MousePointer" VALUE="0">
      <PARAM NAME="Size" VALUE="3895;635">
      <PARAM NAME="PasswordChar" VALUE="0">
      <PARAM NAME="ListWidth" VALUE="0">
      <PARAM NAME="BoundColumn" VALUE="1">
      <PARAM NAME="TextColumn" VALUE="65535">
      <PARAM NAME="ColumnCount" VALUE="1">
      <PARAM NAME="ListRows" VALUE="8">
      <PARAM NAME="cColumnInfo" VALUE="0">
      <PARAM NAME="MatchEntry" VALUE="2">
      <PARAM NAME="ListStyle" VALUE="0">
      <PARAM NAME="ShowDropButtonWhen" VALUE="0">
      <PARAM NAME="ShowListWhen" VALUE="1">
      <PARAM NAME="DropButtonStyle" VALUE="1">
      <PARAM NAME="MultiSelect" VALUE="0">
      <PARAM NAME="Value" VALUE="">
      <PARAM NAME="Caption" VALUE="">
      <PARAM NAME="PicturePosition" VALUE="458753">
      <PARAM NAME="BorderColor" VALUE="2147483654">
      <PARAM NAME="SpecialEffect" VALUE="2">
      <PARAM NAME="Accelerator" VALUE="0">
      <PARAM NAME="GroupName" VALUE="">
      <PARAM NAME="FontName" VALUE="Times New Roman">
      <PARAM NAME="FontEffects" VALUE="1073741824">
      <PARAM NAME="FontHeight" VALUE="240">
      <PARAM NAME="FontOffset" VALUE="0">
      <PARAM NAME="FontCharSet" VALUE="0">
      <PARAM NAME="FontPitchAndFamily" VALUE="2">
      <PARAM NAME="ParagraphAlign" VALUE="2">
      <PARAM NAME="FontWeight" VALUE="400"></OBJECT>
                  </TD>
                  </TR>
                  <TR>
```

```
                        <TD ALIGN="right">Dobanda totala:</TD>
                        <TD>
        <OBJECT style="LEFT: 0px; WIDTH: 183px; TOP: 0px; HEIGHT: 30px"✂
classid="clsid:8BD21D10-EC42-11CE-9E0D-00AA006002F3"
      name=dtotala>
      <PARAM NAME="VariousPropertyBits" VALUE="746604571">
      <PARAM NAME="BackColor" VALUE="2147483653">
      <PARAM NAME="ForeColor" VALUE="2147483656">
      <PARAM NAME="MaxLength" VALUE="0">
      <PARAM NAME="BorderStyle" VALUE="0">
      <PARAM NAME="ScrollBars" VALUE="0">
      <PARAM NAME="DisplayStyle" VALUE="1">
      <PARAM NAME="MousePointer" VALUE="0">
      <PARAM NAME="Size" VALUE="3874;635">
      <PARAM NAME="PasswordChar" VALUE="0">
      <PARAM NAME="ListWidth" VALUE="0">
      <PARAM NAME="BoundColumn" VALUE="1">
      <PARAM NAME="TextColumn" VALUE="65535">
      <PARAM NAME="ColumnCount" VALUE="1">
      <PARAM NAME="ListRows" VALUE="8">
      <PARAM NAME="cColumnInfo" VALUE="0">
      <PARAM NAME="MatchEntry" VALUE="2">
      <PARAM NAME="ListStyle" VALUE="0">
      <PARAM NAME="ShowDropButtonWhen" VALUE="0">
      <PARAM NAME="ShowListWhen" VALUE="1">
      <PARAM NAME="DropButtonStyle" VALUE="1">
      <PARAM NAME="MultiSelect" VALUE="0">
      <PARAM NAME="Value" VALUE="">
      <PARAM NAME="Caption" VALUE="">
      <PARAM NAME="PicturePosition" VALUE="458753">
      <PARAM NAME="BorderColor" VALUE="2147483654">
      <PARAM NAME="SpecialEffect" VALUE="2">
      <PARAM NAME="Accelerator" VALUE="0">
      <PARAM NAME="GroupName" VALUE="">
      <PARAM NAME="FontName" VALUE="Times New Roman">
      <PARAM NAME="FontEffects" VALUE="1073741824">
      <PARAM NAME="FontHeight" VALUE="240">
      <PARAM NAME="FontOffset" VALUE="0">
      <PARAM NAME="FontCharSet" VALUE="0">
      <PARAM NAME="FontPitchAndFamily" VALUE="2">
      <PARAM NAME="ParagraphAlign" VALUE="2">
      <PARAM NAME="FontWeight" VALUE="400"></OBJECT>
                        </TD>
                  </TR>
                  <TR>
                        <TD COLSPAN="2" ALIGN="CENTER"><INPUT TYPE=button✂
NAME="bCalcul" VALUE="Calcul"></TD>
                  </TR>
            </TABLE>
            <SCRIPT FOR="bCalcul" EVENT="onClick" LANGUAGE="VBScript">
                  Set frm = Document.frmDob
                  If frm.Suma.Value = "" Then
                      alert ("Suma pentru care se va " & _
                    "calcula dobanda nu poate fi 0!")
                      frm.Suma.focus
                  ElseIf frm.DobandaAn.Value = "" Then
                      alert ("Dobanda anuala data de " & _
                    "banca nu poate fi 0%!")
                      frm.DobandaAn.focus
                  ElseIf frm.Luni.Value = "" Then
                      alert ("Numarul de luni pentru " & _
                    "care se va calcula dobanda lunara nu poate fi 0!")
                    frm.Luni.focus
                  Else
                    Call CalculDobanda
                  End If
            </SCRIPT>
```

```
        </FORM>
</BODY>
```

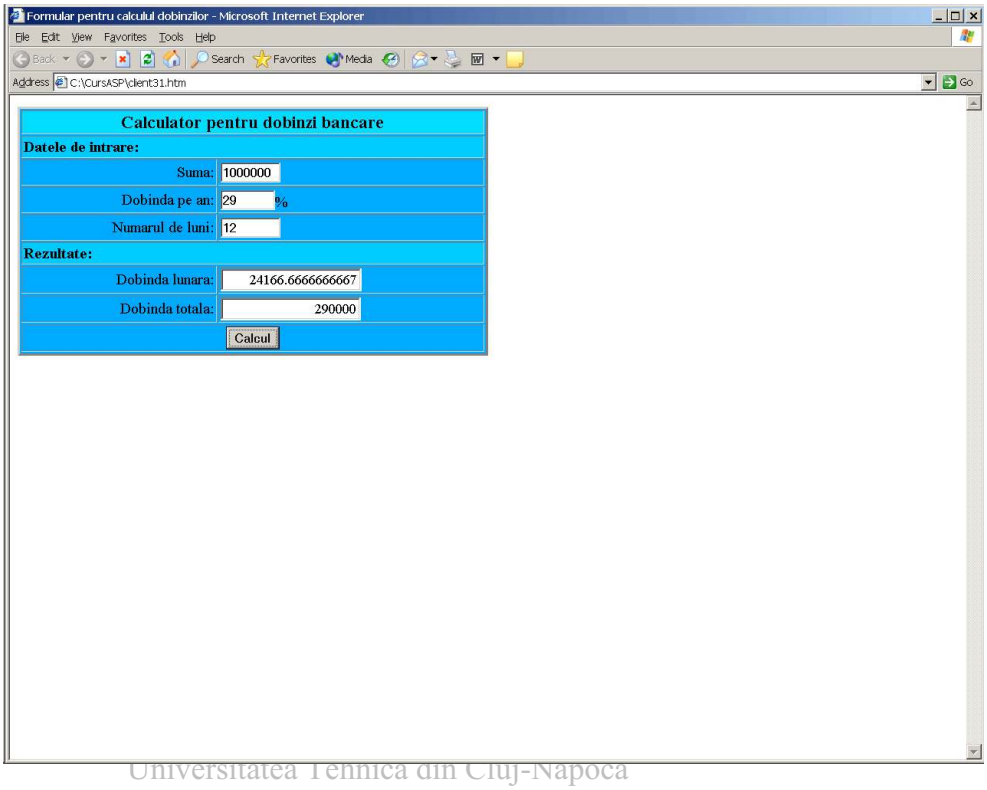


Figure 42 Modul de afişare a rezultatelor folosind controale ActiveX

Obiecte server

ASP are şapte obiecte care se pot folosi pentru a:

- trimite HTML şi date navigatorului;
- extrage informaţii din navigator;
- comunica cu server-ul;
- stoca date ale aplicaţiei sau ale unui utilizator;
- trata erorile.

Din punctul de vedere al utilizatorului aceste obiecte expun proprietăţi şi metode care pot fi accesate sau rulate. Lista celor şapte obiecte se prezintă în tabelul următor:

Obiect	Descriere
`Response`	Utilizat pentru trimiterea informaţiilor la client.
`Request`	Utilizat pentru extragerea informaţiilor trimise de client printr-o cerere.
`Server`	Utilizat pentru a comunica cu server-ul.
`Application`	Utilizat pentru a stoca informaţii despre aplicaţie.
`Session`	Utilizat pentru a stoca informaţii particulare unei instanţe de navigator (uneori, aceasta corespunde unui singur utilizator).
`ObjectContext`	Utilizat pentru a iniţia sau a controla tranzacţii şi pentru a crea noi obiecte prin Microsoft Transaction Server (MTS).
`ASPError`	Utilizat pentru a obţine informaţii cu privire la eroarea care apare în timp ce motorul ASP prelucrează un script.

Până acum, pentru a rula o secvenţă de cod pe client, am folosit marcajul `SCRIPT` cu următoarea formă de scriere `<SCRIPT LANGUAGE="VBScript"> ... </SCRIPT>`. Un exemplu simplu de astfel de cod este:

```
<SCRIPT LANGUAGE="VBScript">
      Document.Write time
</SCRIPT>
```

Pentru ca o secvenţă de cod să fie executată pe server-ul de Web avem două variante de scriere:

- secvenţa de cod se scrie între delimitatorii `<% %>`;
- în marcajul `SCRIPT`, care va cuprinde secvenţa de cod, se va folosi atributul `RUNAT=SERVER`.

Fie codul:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Response.Write "Un text ..."
%>
</BODY>
</HTML>
```

deşi, la prima vedere, pare a fi în regulă la începutul lui ar trebui să stea linia:

```
<%@ Language=VBScript %>
```

Aceasta, prin caracterul `@`, spune motorului ASP că limbajul de script implicit folosit în pagină este `VBScript`.

NOTĂ

Pentru toate secvenţele de cod care urmează este nevoie să aveţi instalat un server de Web (PWS sau IIS). Fişierele care se vor rula pe server vor avea extensia ".asp" şi nu pot fi deschise direct din navigator.

Obiectul Response

Când un navigator cere date unui server acesta răspunde fie direct, cu datele cerute, fie cu o redirectare, fie cu o eroare. Răspunsul are două porţiuni principale, antetul şi corpul. Antetul răspunsului (`response header`) conţine directive şi/sau informaţii despre conţinut cum sunt: tipul, data expirării, cookies etc. Corpul răspunsului (`response body`) conţine datele.

În cazul cererilor HTML, răspunsul este dat direct de server-ul de Web, care citeşte şi întoarce conţinutul paginii HTML. În cazul paginilor ASP, datele răspunsului sunt generate prin obiectul ASP, numit, `Response`.

Trimiterea unui răspuns cu obiectul Response

Pentru a putea rula pe server aplicaţia care urmează, aceasta trebuie să fie stocată într-un director virtual. Dacă fişierul se numeşte `aspr1.asp` şi este stocat în directorul virtual `CursASP` se poate folosi următorul URL pentru rularea lui:

```
http://server/CursASP/aspr1.asp
```

Mai sus, în locul lui `server`, trebuie să scrieţi numele server-ului pe care se va rula codul ASP. Dacă nu cunoaşteţi numele server-ului (în general este acelaşi cu cel al calculatorului) se poate folosi numele generic de `localhost` pentru server-e Web locale (acesta va rula pe calculatorul pe care se află şi codul ASP corespunzător). Astfel, se poate scrie:

```
http://localhost/CursASP/aspr1.asp
```

Conţinutul fişierului `aspr1.asp` se prezintă în continuare:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Response.Write "Un text ..." & "<BR>"
      Response.Write "afisat cu ""Response.write""" & "<BR>"
%>
</BODY>
</HTML>
```

Dacă în loc de textul:

```
Un text ...
afisat cu "Response.write"
```

veţi primi un mesaj de eroare, verificaţi dacă server-ul de Web este încărcat şi funcţional, apoi dacă are instalată partea de ASP şi dacă `CursASP` este director virtual.

Utilizarea variabilelor în răspunsul dat de server

`Response.write` permite şi utilizarea variabilelor. Pentru aceasta variabila trebuie declarată cu `Dim` (`Dim` vine de la `dimension`, iar în **VBScript** această procedură nu este obligatorie), apoi trebuie să i se atribuie şi o valoare înainte de afişare. Codul care urmează prezintă modul în care se pot afişa ghilimelele şi modalitatea de folosire a lui `<% Option explicit %>` pentru a primi o eroare de sintaxă dacă se utilizează o variabilă care nu a fost declarată cu `Dim`.

```
<%@ Language=VBScript %>
<% Option explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim unSir
      unSir = "Un text ... <BR>afisat cu ""Response.write""<BR>"
      Response.write unSir
%>
</BODY>
</HTML>
```


De exemplu, dacă în codul de mai sus în loc de `Response.write unSir`, se va scrie, din grabă, `Response.write unir` şi dacă linia `Option explicit` este prezentă următorul mesaj de eroare va fi afişat:

```
Error Type:
Microsoft VBScript runtime (0x800A01F4)
Variable is undefined: 'unir'
/CursASP/aspr2.asp, line 10
```

Majoritatea paginilor ASP sunt o mixtură de HTML şi cod script. Iată un exemplu de acest fel:

```
<%@ Language=VBScript %>
<% Option explicit %>
<%
      Dim Cine
      Dim Cind
      Cine="Vasile Ionel"
      Cind=Time
%>
<HTML>
<HEAD>
<TITLE>O mixtura de HTML si cod VBScript</TITLE>
</HEAD>
<BODY>
Omul in cauza se numeste <%=Cine%> si a fost pe aici la ora: <%=Cind%>
</BODY>
</HTML>
```

Întrucât semnul = din codul `<%=Cine%>` este o formă de scriere prescurtată a lui `Response.write`, codul scris desfăşurat este `<%Response.write Cine%>`.

Colecţia Response.Cookies

Activitatea de programare nu se poate desfăşura fără variabile. S-a văzut deja modul în care se

poate crea şi utiliza o variabilă în codul HTML. Una dintre problemele programării ASP este aceea de păstrare a valorilor variabilelor la schimbarea paginilor, deseori denumirea folosită în acest scop fiind aceea de păstrare a stării aplicației. Din punctul de vedere al motorului ASP, după ce o pagină a fost prelucrată complet, toate variabilele folosite în pagină sunt uitate. Există mai multe metode de păstrare a valorilor variabilelor, cea mai veche dintre ele fiind cea care se prezintă în continuare. Un cookie, în cea mai simplă varintă, este o pereche de forma `cheie=valoare` (de exemplu, `CineNume=Vasile`).

Cea mai simplă modalitate de creare a unui cookie este folosirea de HTML cu un antet care conține perechea cookie după cum se observă în exemplul care urmează:

```
<META HTTP-EQUIV="set-cookie" CONTENTS="CineNume=Vasile; path=/;>
```

Se observă că, din punctul de vedere al server-ului, un cookie este o informație în plus în antetul paginii. Din punctul de vedere al clientului, un cookie este o modalitate de a stoca o cantitate mică de date pe client. Cantitatea de date care poate fi stocată este dependentă de navigator, dar majoritatea navigatoarelor moderne acceptă cel mult 1024 de caractere pe post de date.

O altă metodă de creare a unui cookie este colecția `Cookie` a obiectului `Response`. O colecție este un obiect care conține o listă de articole. Colecția `Cookie` conține lista perechilor cookie `cheie=valoare` pe care dorim să le stocăm pe client. Pentru a seta un cookie scriem:


```
Response.Cookie("CineNume")="Vasile"
```


iar pentru a şterge valoarea lui scriem:

```
Response.Cookie("CineNume")=""
```

Este posibilă şi crearea unui cookie pe mai multe nivele pentru stocarea de valori multiple. În acest caz, fiecare sub-valoare este asociată unei sub-chei. Iată un exemplu:

```
Response.Cookie("Cine")("Nume")="Vasile"
Response.Cookie("Cine")("Prenume")="Ionel"
```

Câteva aspecte legate de cookie-uri sunt prezentate în tabelul următor:

Aspect	Descriere
un cookie este specific unui site	După crearea unui cookie, navigatorul întoarce cookie-ul pentru fiecare cerere adresată site-ului. La fiecare cerere de URL a navigatorului el verifică lista de cookie-uri şi dacă găseşte unul stocat va trimite valorile (stocate) corespunzătoare acelui site către server.
sunt sigure	Navigatorul trimite cookie-urile doar dacă acestea sunt specifice unui anumit domeniu şi server, adică un alt site nu va putea citi setările de cookie făcute de noi pentru un site al nostru. Parametrul path permite setarea de cookie-uri specifice unor anumite directoare din site. În acest caz navigatorul va trimite cookie-urile doar dacă în URL-ul dorit, domeniul, server-ul şi calea sunt cele dorite.

Aspect	Descriere
expiră în timp	Pentru fiecare cookie se poate seta o dată şi o oră de expirare, după care acesta nu mai este valid. Dacă data de expirare este setată, navigatorul salvează cookie-ul pe disc. În lipsa acestei setări, navigatorul va ține cookie-ul în memorie, iar el va expira atunci când se închide navigatorul.
pot fi securizate	Comunicația normală între navigatoare şi server-e se face prin text ASCII, aici fiind incluse şi cookie-urile. Programatorii, deseori, folosesc cookie-urile pentru a stoca date sensibile, de exemplu parole. Persoane rău intenționate, cu puțin efort, pot citi aceste valori. Din acest motiv se poate face ca un cookie să fie trimis printr-o conexiune securizată (de exemplu, via HTTPS). În acest caz navigatorul nu va trimite cookie-ul printr-o conexiune fără securitate. Dacă cookie-ul se transmite printr-o conexiune securizată el este încriptat şi nu mai poate fi citit prea uşor de hacker-i.

Metoda Response.AddHeader

În cazul în care dorim să vizualizăm conținutul unei pagini majoritatea navigatoarelor au opțiunea **View Source**. Pentru informațiile care se transferă prin antet, cum este un cookie, nu există însă o soluție de vizualizare directă. Există totuşi posibilitatea de a defini conținutul antetului cu ajutorul metodei `Response.AddHeader`. Tot ceea ce face această metodă se poate scrie şi direct în HTML, însă aceasta se poate realiza prin cod.

Codul care urmează foloseşte marcajul `SPAN`. Acesta face parte dintre mecanismele generice pentru structurarea documentelor, fiind folosit mult de către FrontPage şi DreamWeaver. `SPAN` este folosit în continuare pentru a denumi o porțiune a textului de afişat. Astfel se va defini o zonă de text care va putea fi identificată printr-un nume. Atributul `ID` folosit cu `SPAN` dă un nume unic elementului în scopul de a-l putea referi ulterior prin cod. În acest fel se defineşte un nou obiect. În codul următor el se numeşte `TimpRedirectare`, pe care altfel, în HTML, nu am putea să-l definim ca o entitate distinctă şi identificabilă din cod. Pentru exemplificarea redirectării se folosesc două fişiere, `aspr4.asp` şi `asp41.asp` ale căror coduri se prezintă în continuare. Redirectarea are loc ca urmare a liniei `<% Response.AddHeader "Refresh","7; URL= aspr41.asp"%>`. Observați că ";" se foloseşte pentru a separa valorile multiple la fel ca şi în HTML. Datorită modului de scriere al URL-ului redirectarea va reuşi doar dacă cele două fişiere sunt stocate în acelaşi director.

Fişierul `aspr4.asp`:

```
<%@ Language=VBScript %>
<% Option explicit %>
<% Response.Buffer = True %>
<% Response.AddHeader "Refresh","7;URL=aspr41.asp"%>

<!--META HTTP-EQUIV="REFRESH" CONTENT="7;
URL=http://localhost/CursASP/aspr41.asp"&gt;-->

<HTML>
<HEAD>
<TITLE>Exemplu de redirectare cu Response.AddHeader</TITLE>

<SCRIPT LANGUAGE="VBScript">
```

```
call window.setTimeout("timer",1000,"VBScript")
Function timer()
      Dim  Timp
      Timp = document.all("TimpRedirectare").innertext
      Timp = Timp - 1
      document.all("TimpRedirectare").innerText = Timp
      If (Timp > 0) Then
            call window.setTimeout("timer",1000,"VBScript")
      End If
End function

</SCRIPT>
</HEAD>
<BODY>
Pagina pe care o cautati a fost mutata. Noua locatie este:
<A HREF="http://localhost/CursASP/aspr41.asp">aspr41.asp</A>.
<P>
Daca navigatorul suporta redirectarea noua locatie
se va deschide automat in <SPAN ID="TimpRedirectare">7</SPAN> secunde.
</BODY>
</HTML>
```

Fişierul `aspr41.asp`:

```
<HTML>
<HEAD>
</HEAD>
<BODY>

<P>Ati fost redirectat pe pagina &quot;aspr41.asp&quot;.</P>

</BODY>
</HTML>
```


Funcția `timer` se execută pe client. Deoarece aceasta este scrisă în **VBScript**, clientul trebuie să fie IE. Ea permite afişarea timpului rămas până la redirectare cu ajutorul contorului `TimpRedirectare`, însă nu realizează redirectarea. Linia a treia de cod `<% Response.Buffer = True %>` este esențială pentru funcționarea redirectării după ce server-ul a transmis deja o porțiune de cod HTML. Tipic, server-ul trimite mai întâi antetul (`header`), apoi corpul (`body`). După ce server-ul a transmis antetul, acesta nu mai poate fi schimbat deoarece el este deja pe calculatorul clientului. Dacă apare situația schimbării antetului, ca urmare a prelucrării unui script ieşirea trebuie stocată într-un buffer înainte ca acesta să înceapă să fie trimis clientului. Răspunsul buffer-at va conține atât antetul, cât şi corpul. În acest fel ne putem răzgândi cu privire la pagina care va fi deschisă. Se poate trimite un antet cu redirectare către navigator.

Metoda Response.Redirect

Metoda `Response.AddHeader` este utilă atunci când se cunoaşte, în avans, că trebuie să se facă o redirectare şi se mai cunoaşte URL-ul de redirectare. Există însă cazuri când redirectarea trebuie făcută dinamic, pe baza unor decizii luate în cod. În acest caz trebuie folosită metoda `Response.Redirect`. De exemplu, pentru a realiza o redirectare către pagina Universității Tehnice din Cluj-Napoca se scrie:

```
Response.Redirect "http://www.utcluj.ro/"
```

Codul de redirectare este executat imediat cum se ajunge la el, din acest motiv server-ul nu mai execută codul care urmează după această linie şi nu va trimite nici codul HTML din buffer. Dacă se uită scrierea liniei `<% Response.Buffer = True %>` la începutul paginii, server-ul va genera o eroare când încearcă să execute redirectarea, iar mesajul afişat va fi ceva de forma:

```
Response object error 'ASP 0156 : 80004005'

Header Error

/CursASP/nume.asp, line 23
The HTTP headers are already written to the client browser. Any HTTP
header modifications must be made before writing page content.
```

Proprietăţile Response.Expires şi Response.ExpiresAbsolute

Terminologie

- *cache* - denumire folosită pentru o memorie mică şi rapidă care stochează datele accesate recent. Deseori este folosită pentru accesul la memorie a procesorului, pentru copii locale ale datelor accesibile printr-o rețea etc.
- *proxy server* - un proces care furnizează un cache de articole pentru alte server-e care se presupune că sunt mai încete şi cu un cost de acces mai mare. În particular, termenul se foloseşte pentru un server de Web care acceptă URL-uri cu un prefix special. Atunci când primeşte o cerere de astfel de URL, extrage prefixul şi caută URL-ul obținul în cache-ul local. Dacă acesta este găsit, răpunsul se întoarce imediat. Altfel va fi adus de pe un alt server, se salvează în cache, apoi se întoarce celui care l-a cerut. În general, cache-ul are un algoritm de expirare şi se goleşte pe baza vârstei, mărimii şi istoriei de accesare a documentelor din el.

Paginile ASP au conținutul dinamic, iar cele HTML îl au static. S-a depus un efort mare pentru îmbunătățirea timpului de răspuns în cazul paginilor HTML pe Web. Metoda cea mai des folosită este mutarea paginii mai aproape de navigatorul care îi cere conținutul. De aceea, majoritatea navigatoarelor formează un cache pe disc, cu paginile accesate. Prima încărcare a paginii va dura mai mult deoarece pagina se transferă de pe server. După aceea, ea se stochează local şi toate accesele ulterioare la aceeaşi pagină se vor face prin încărcarea din cache, deci mult mai rapid. Această strategie de lucru este perfectă pentru pagini HTML, ca urma a conținutului lor static, dar nu merge în cazul paginilor ASP. Paginile ASP se modifică în urma schimbării surselor sau din cauza parametrilor de intrare modificați de un anumit utilizator. Un proxy server foloseşte cache pentru pagini pentru a evita un transfer lung de la server-ul care ține pagina originală. Dacă navigatorul afişează o pagină din cache aceasta poate să fie expirată sau poate să nu fie ceea ce şi-a dorit utilizatorul. Codul ASP de pe server-ul care ține pagina originală nu va fi rulat deoarece proxy-ul interceptează cererea înainte ca ea să ajungă la server. Obiectul `Response` are câteva proprietăți pentru a rezolva aceste probleme.

Proprietatea `Response.Expires` defineşte durata de timp, în minute, în care pagina este validă. După trecerea minutelor pagina va expira. Până când pagina expiră este luată din cache, apoi se reîncarcă de pe server. De exemplu, `Response.Expires=50` spune navigatorului că pagina expiră după 50 de minute. Proprietatea `Response.ExpiresAbsolute` lucrează asemănător cu `Response.Expires`, doar că aici se poate specifica şi o dată. De exemplu, `Response.ExpiresAbsolute = #July 28, 2007#` (într-un şir, caracterele # se folosesc pentru a defini o dată) specifică navigatorului că pagina va expira în 28 iulie 2007. Problema acestor proprietăți este că nu funcționează întotdeauna, deseori din cauza navigatoarelor (navigatoarele lui Netscape se fac vinovate cel mai des de un comportament ciudat în acest sens), caz în care aplicația ASP nu va lucra, nici ea, corect.

Cache-ul realizat de navigator mai poartă denumirea de pre-caching. O altă cauză a nefuncționării corecte poate fi cache-ul făcut de proxy care nu interpretează corect timpul şi data de expirare. Pentru a forța proxy-ul să aducă o copie nouă a paginii se va folosi proprietatea `Response.CacheControl`. Proprietatea poate lua două valori, `Public` şi `Private`. Implicită este

valoarea `Private`, caz în care nu se doreşte ca server-ul să stocheze în cache pagina. Din păcate nici această proprietate nu este luată în considerare de toate proxy server-ele, aşa că cea mai simplă modalitate de a forța reîncărcarea paginii la fiecare accesare este ca ea să fie expirată încă de la primul acces la ea. Un exemplu de lucru în acest sens este hiperlegătura:

```
<A HREF="pagina.asp?UniqueURL=1/1/1990 1:1:11 AM">
      Clic aici pentru pagina.asp
</A>
```

Obiectul Request

Obiectul `Response` se foloseşte pentru a trimite un conținut navigatorului. Obiectul `Request` este opusul primului el fiind folosit pentru a extrage un conținut din navigator. `Request` are mai multe colecții care încapsulează informațiile trimise de navigator la fiecare cerere. Atât server-ul cât şi navigatorul tratează fiecare cerere ca şi o comunicație nouă, aici fiind incluse toate informațiile necesare derulării comunicației (adresă de IP, tip de informație cerută, conținutul formularelor, variabilele `QueryString`, informații specifice navigatorului). Obiectul `Request` se foloseşte pentru citirea acestor informații de cinci colecții distincte care se prezintă în ordine alfabetică în tabelul următor:

Colecție	Descriere
`ClientCertificates`	Stochează informații legate de securitate.
`Cookies`	Conține valorile de cookie trimise de navigator.
`Form`	Conține informațiile introduse de utilizator în controalele de intrare şi pe cele stocate de aplicație în variabilele formularului.
`QueryString`	Conține informațiile trimise împreună cu URL-ul.
`ServerVariables`	Conține informațiile pe care server-ul le transferă automat cu fiecare cerere.

Toate colecțiile au următoarele elemente comune:

Denumire	Descriere
`Count`	Proprietatea stochează numărul de elemente ale colecției. Primul element are indicele 1 (la tablouri acesta era 0), iar valoarea ei se citeşte prin `Variabilă= NumeColecție.Count`
`Item`	Metoda permite extragerea unui articol din colecție specificat prin nume sau indice după următoarele sintaxe: `Variabilă= NumeColecție.Item("NumeCheie")` sau `Variabilă= NumeColecție.Item(Index)`
`Key`	Metoda permite extragerea valorii unei chei pe baza unui index. În practică se foloseşte rar, sintaxa fiind: `Variabilă= NumeColecție.Key)`

Toate colecțiile ASP au proprietatea implicită `Item`, iar atunci când ne referim la aceasta ea nu mai trebuie scrisă.

Colecţia Request.ClientCertificates

Această colecție conține certificatele de securitate (valori binare unice) pe care navigatorul le

transferă atunci când cere informații de la un site securizat prin protocolul HTTP. Nu toate navigatoarele suportă lucrul cu certificate de securitate. Server-ul foloseşte certificatul de securitate pentru identificarea utilizatorilor. Colecția `Request.ClientCertificate` este întotdeauna vidă atunci când navigatorul cere informații prin protocolul HTTP standard. Pentru obținerea de valori ale certificatelor de securitate trebuie realizată o conexiune de site securizat la server prin IIS.

Colecţia Request.Cookies

Navigatorul trimite cookie-urile, în antet, prin variabila `HTTP_COOKIE`. Pentru citirea cookie-urilor se va folosi colecția `Request.Cookies` care este complementara colecției `Response.Cookies`. Dacă prin folosirea lui `Response.Cookies` este setat un cookie, acesta va fi inclus în următoarea cerere a navigatorului client. Dacă cookie-ul este şters el nu mai apare în colecția următoarei cereri. Dacă se modifică valoarea lui, valoarea din colecție se va modifica şi ea în următoarea cerere a navigatorului. Pentru a da o valoare unui cookie numit cheie se foloseşte sintaxa deja cunoscută `Response.Cookies("cheie")=valoare` ; pentru citirea acestei valori se va folosi sintaxa:

```
Dim V
V=Request.Cookies("cheie")
```

Pentru a parcurge întreaga colecție de cookie-uri se va folosi instrucțiunea `For Each ... Next`:


```
Dim V
For Each V in Request.Cookies
      Response.Write V & " = " Request.Cookies.Item(V)
Next
```

Exemplul funcționează numai dacă fiecare cookie din colecție ia câte o singură valoare. În cazul valorilor multiple trebuie folosită proprietatea `hasKeys` care întoarce `True` dacă există subchei. Exemplul următor prezintă şi o modalitate de accesare a unui cookie cu mai multe valori. Aplicația are la bază fişierele `asprq_tr.asp`, `asprq_st.asp`, `asprq_ex.asp` ale căror coduri se prezintă în continuare.

Fişierul `asprq_tr.asp`:

```
<HTML>
<BODY>
<FORM ACTION="asprq_st.asp" METHOD="POST">
<H3>Introdu datele tale:</H3>
<TABLE COLS="2" BORDER="2">
      <TR>
            <TD>NUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Nume" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>PRENUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Prenume" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>Tipul de utilizator:</TD>
            <TD>
                  <SELECT NAME="Tip">
                  <OPTION VALUE="ADMINISTRATOR">Administrator
                  <OPTION VALUE="VIZITATOR">Vizitator
                  <OPTION VALUE="UTILIZATOR">Utilizator
```

```
            </TD>
      </TR>
      <TR>
            <TD ALIGN="CENTER"> <INPUT TYPE="SUBMIT" VALUE="Trimite"></TD>
            <TD ALIGN="CENTER"> <INPUT TYPE="RESET" VALUE="Reset"></TD>
      </TR>
</TABLE>
</FORM>
</BODY>
```

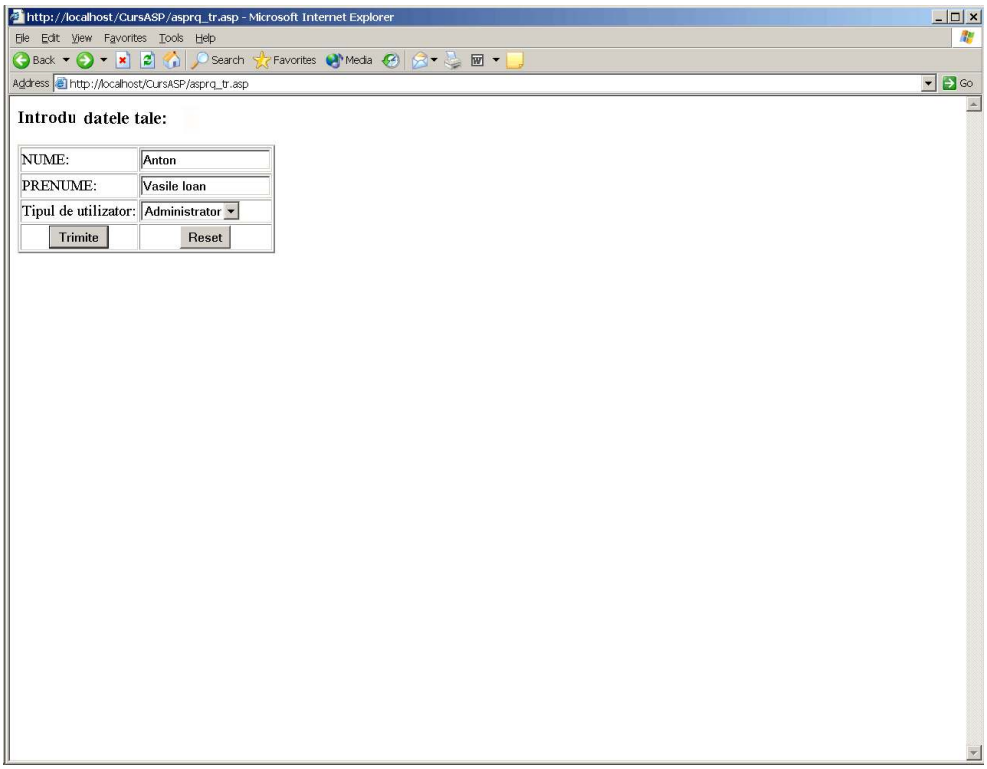
Fişierul `asprq_st.asp`:

```
<%
      Response.Cookies("Persoana")("Nume")= Request.Form("Nume")
      Response.Cookies("Persoana")("Prenume")= Request.Form("Prenume")
      Response.Cookies("Persoana").Expires="July 28, 2004"
      Response.Cookies("Tip")= Request.Form("Tip")
      Response.Cookies("Tip").Expires="July 28, 2004"
%>
<HTML>
<BODY>
<P>Cookie-urile sint stocate!
<P>Click pe <A HREF = "asprq_ex.asp">asprq_ex.asp</A> pentru afisarea✂
rezultatelor stocate.
</BODY>
```

Fişierul `asprq_ex.asp`: http://www.east.utcluj.ro/mb/mep/antal


```
<%@ Language=VBScript %>
<% Option Explicit %>
<%
      Dim Nume
      Dim Prenume
      Dim Tip
      Nume = Request.Cookies("Persoana")("Nume")
      Prenume = Request.Cookies("Persoana")("Prenume")
      Tip = Request.Cookies("Tip")
%>
<HTML>
<BODY>
      Salutare <%=UCase(Nume)%>   <%=Prenume%>, ai drepturile de acces✂
de <%=Tip%>!<BR><BR>
<%
      Dim V
      Dim SubV
      For Each V In Request.Cookies
         If Request.Cookies(V).hasKeys Then
            Response.Write V  & "<BR>"
            For Each SubV in Request.Cookies(V)
               Response.Write "      " & _
               SubV & " = " & Request.Cookies(V)(SubV) & "<BR>"
            Next
         Else
            Response.Write V & " = " & Request.Cookies(V) & "<BR>"
         End If
      Next
%><BR>
<%
      Response.Write "HTTP_COOKIE = " & Request.ServerVariables✂
 ("HTTP_COOKIE") &  "<BR>"
%></BODY></HTML>
```

Figure 43 Fişierul asprq_tr.asp

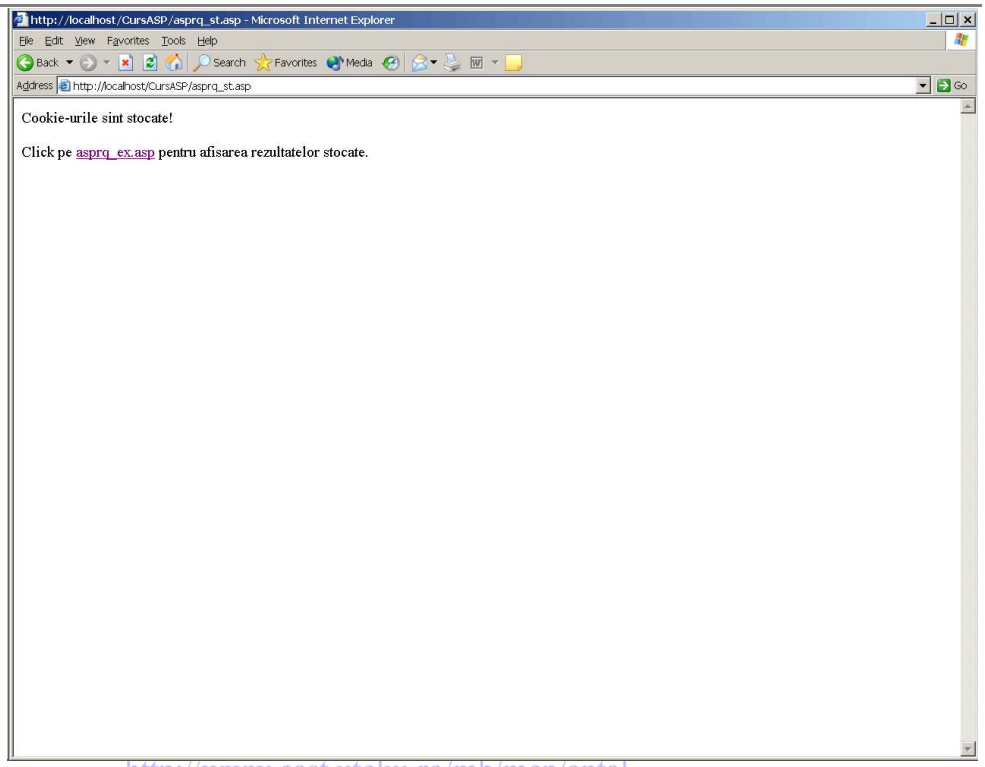


Prima oară se va deschide pagina `asprs_tr.asp`, care va fi completată aşa cum se vede în **Figura 43**. Apoi, după apăsarea lui **Trimite**, pe ecran se afişează pagina din **Figura 44**. Secvenţa de cod care urmează este o implementare recursivă a codului de afişare a valorilor de cookie din `asprq_ex.asp`.

Ea are avantajul funcţionării corecte, indiferent de numărul subcheilor care s-au definit.

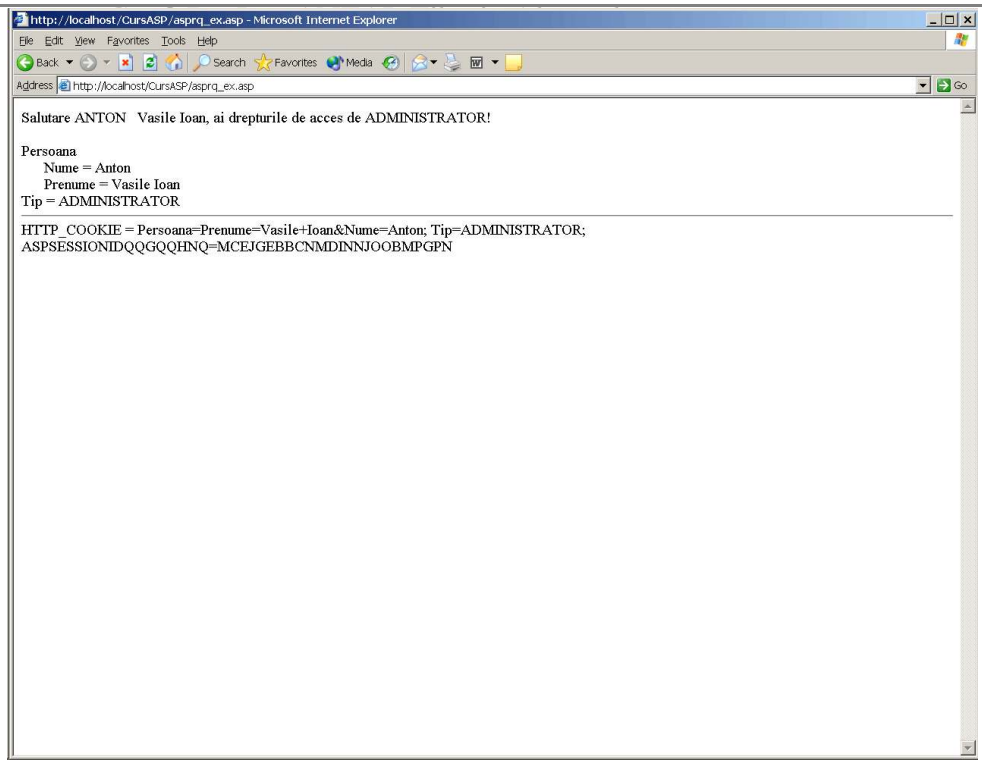
```
<%
      Sub afisareCookie(C)
        Dim V
        For Each V In C
          If IsObject(C(V)) then
              If C(V).hasKeys Then
                Response.Write V  & "<BR>"
                Call afisareCookie(C(V))
              Else
                Response.Write V & " = " & C(V) & "<BR>"
              End If
           Else
            Response.Write "      " & V & " =✂
 " & C(V) & "<BR>"
           End If
        Next
      End Sub
      Dim C
      Set C = Request.Cookies
      afisareCookie(C)

%>
```

Figure 44
Fişierul asprq_st.asp



Aici, se face clic pe hiperlegătură, iar pe ecran apare pagina din **Figura 45**.

Figure 45
Fişierul asprq_ex.asp



Colecţia Request.Forms

Când navigatorul transferă datele unui formular către server cu metoda `POST`, motorul ASP separă şi stochează datele formularului în colecţia `Request.Form`. Metoda implicită a colecţiei este `Item`, din acest motiv în loc de `Request.Form.Item("Telefon")` se poate scrie

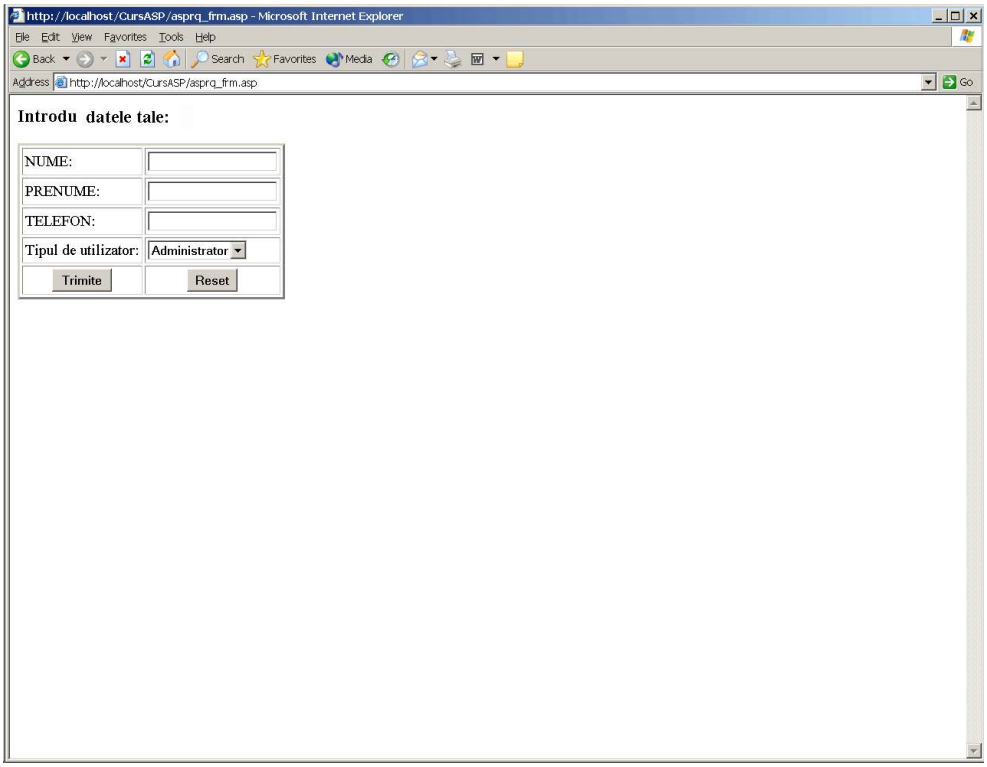
`Request.Form("Telefon")`. Numărul de articole stocate în colecție se poate găsi prin `Request.Form.Count`. Dacă formularul conține mai multe controale cu acelaşi nume, de exemplu `Telefon`, atunci acestea pot fi referite prin sintaxa:

```
Request.Form("Telefon")(1)
Request.Form("Telefon")(2)
```

Numărul de valori distincte date aceluiaşi control se găseşte prin `Request.Form("Telefon").Count`. Pentru afişarea tuturor cheilor şi valorilor de cheie din colecția `Request.Form` se poate folosi codul:

```
Dim i
For i=1 To Request.Form.Count
    Response.Write Request.Form.Key(i) & " = " & Request.Form.Key(i) & "<BR>"
Next
```

Figure 46 Fişierul asprq_frm.asp



Cel mai simplu mod de a învăța lucrul cu această colecție este folosirea unui formular cu buton de transfer (`submit`) care va realiza transferul însăşi către formularul cu pricina. Codul corespunzător imaginii din **Figura 46** se prezintă în continuare.

```
<%@ Language=VBScript %>
<% Option Explicit %>
<%
      Function numaiCaractere(sir)
            Dim i
            if Len(sir) = 0 Then
                  numaiCaractere = False
                  Exit Function
            End if
            For i = 1 To Len(sir)
                  If instr(1, "abcdefghijklmnopqrstuvwxyz'", _
                        mid(sir,i,1), vbTextCompare) = 0 Then
```

```
                        numaiCaractere = False
                        Exit Function
                  End If
            Next
            numaiCaractere = True
    End Function

    Function numaiCifre(sir)
            Dim i
            If Len(sir) = 0 Then
                  numaiCifre = False
                  Exit Function
            End If
            For i = 1 To Len(sir)
                  If instr(1, "0123456789-()", _
                        mid(sir,i,1), vbTextCompare) = 0 Then
                        numaiCifre = False
                        Exit Function
                  End If
            Next
            numaiCifre = True
    End Function

      If Request.ServerVariables("REQUEST_METHOD") = "POST" Then
            Dim eroare
            Dim nume
            Dim prenume
            Dim telefon
            Dim tip
            nume = Request.Form("Nume")
            prenume  = Request.Form("Prenume")
            telefon = Request.Form("Telefon")
            tip = Request.Form("Tip")
            eroare=""

            If Not numaiCaractere(nume) Then
                  eroare = "Campul NUME nu poate fi vid si " & _
                  "poate contine doar litere!<BR>"
            End If
            If Not numaiCaractere(prenume) Then
                  eroare = eroare & "Campul PRENUME nu poate fi vid " & _
                  "si poate contine doar litere!<BR>"
            End If
            If Not numaiCifre(telefon) Then
                  eroare = eroare & "Campul TELEFON nu poate fi vid " & _
                  "si poate contine cifre, - sau ( )!<BR>"
            End If

            If Len(eroare) = 0 Then
                  Response.Write "Nume = " & nume & "<BR>"
                  Response.Write "Prenume = " & prenume & "<BR>"
                  Response.Write "Telefon = " & telefon & "<BR>"
                  Response.Write "Tip = " & tip & "<BR>"
                  Response.End
            End If
      End If
%>

<HTML>
<BODY>
<FORM NAME="Frm1" ACTION="asprq_frm.asp" METHOD="POST">
<H3>Introdu datele tale:</H3>
<%
If Len(eroare) > 0 Then
      Response.Write "<P><FONT COLOR=""RED"">" & eroare & "</FONT></P>"
End If
%>
```

```
<P>
<TABLE COLS="2" BORDER="3" CELLPADDING="3" CELLSPACING="3">
      <TR>
            <TD>NUME:</TD>
            <TD>     <INPUT     TYPE="TEXT"     NAME="Nume"     SIZE="20"✂
VALUE="<%=nume%>"></TD>
      </TR>

      <TR>
            <TD>PRENUME:</TD>
            <TD>     <INPUT     TYPE="TEXT"     NAME="Prenume"     SIZE="20"✂
VALUE="<%=prenume%>"></TD>
      </TR>
      <TR>
            <TD>TELEFON:</TD>
            <TD>     <INPUT     TYPE="TEXT"     NAME="Telefon"     SIZE="20"✂
VALUE="<%=telefon%>"></TD>
      </TR>

      <TR>
            <TD>Tipul de utilizator:</TD>
            <TD>
                  <SELECT NAME="Tip">
                  <OPTION VALUE="ADMINISTRATOR" <% if Tip = "ADMINISTRATOR"✂
Then Response.Write "SELECTED"%> >Administrator
                  <OPTION VALUE="VIZITATOR" <% if Tip = "VIZITATOR" Then✂
Response.Write "SELECTED"%>>Vizitator
                  <OPTION VALUE="UTILIZATOR" <% if Tip = "UTILIZATOR" Then✂
Response.Write "SELECTED"%>>Utilizator
            </TD>
      </TR>
      <TR>
            <TD ALIGN="CENTER"> <INPUT TYPE="SUBMIT" VALUE="Trimite"></TD>
            <TD ALIGN="CENTER"> <INPUT TYPE="RESET" VALUE="Reset"></TD>
      </TR>
</TABLE>
</FORM>
</BODY>
```


Variabila `REQUEST_METHOD` este folosită pentru a detecta dacă formularul a fost transmis cu metoda `POST`. Rolul exemplului este didactic. După ce formularul a fost transmis către el însuşi, se verifică dacă `Nume` şi `Prenune` există şi conţin doar litere şi spaţii, apoi se verifică dacă `Telefon` există şi conţine doar cifre sau caracterele `-`, `(`, `)`. Dacă în urma verificărilor una sau mai multe dintre condiţiile impuse nu sunt îndeplinite atunci se generează mesajele de eroare corespunzătoare. Dacă variabila `eroare` este vidă (conţine şirul nul) înseamnă ca nu au fost erori şi rezultatele pot fi afişate. Codul de mai sus funcţionează numai dacă formularul se transferă cu metoda `POST` şi are dezavantajul că verificările se fac pe server. O variantă mai realistă este codul care urmează.

```
<HTML>
<BODY>

<SCRIPT LANGUAGE="JavaScript">
      function TestJS()
      {
            return TestVB();
      }
</SCRIPT>

<SCRIPT LANGUAGE="VBScript">
      Dim eroare

      Function TestVB()
            eroare=""
```

```
            If Not numaiCaractere(Trim(Frm1.Nume.Value)) Then
                  eroare = "Campul NUME nu poate fi vid si " & _
                  "poate contine doar litere!"& chr(10) & chr(13)
            End If
            If Not numaiCaractere(Trim(Frm1.Prenume.Value)) Then
                  eroare = eroare & "Campul PRENUME nu poate fi vid " & _
                  "si poate contine doar litere!" & chr(10) & chr(13)
            End If
            If Not numaiCifre(Trim(Frm1.Telefon.Value)) Then
                  eroare = eroare & "Campul TELEFON nu poate fi vid " & _
                  "si poate contine cifre, - sau ( )!" & chr(10) & chr(13)
            End If

            If Len(eroare) = 0 Then
                  TestVB = True
            Else
                  document.all("er").innerText = eroare
                  TestVB = False
            End If
      End Function

      Function numaiCaractere(sir)
            Dim i
            if Len(sir) = 0 Then
                  numaiCaractere = False
                  Exit Function
            End if
            For i = 1 To Len(sir)
                  If instr(1, "abcdefghijklmnopqrstuvwxyz'", _
                        mid(sir,i,1), vbTextCompare) = 0 Then
                        numaiCaractere = False
                        Exit Function
                  End If
            Next
            numaiCaractere = True
    End Function

    Function numaiCifre(sir)
            Dim i
            If Len(sir) = 0 Then
                  numaiCifre = False
                  Exit Function
            End If
            For i = 1 To Len(sir)
                  If instr(1, "0123456789-()", _
                        mid(sir,i,1), vbTextCompare) = 0 Then
                        numaiCifre = False
                        Exit Function
                  End If
            Next
            numaiCifre = True
      End Function

      Sub Sterge()
            Frm1.Reset
            document.all("er").innerText = ""
      End Sub

</SCRIPT>

<FORM    NAME="Frm1"    ACTION="asprq_qst.asp"    METHOD="GET"    ONSUBMIT
="return(TestJS());">
<H3>Introdu datele tale:</H3>
<FONT COLOR="RED" SIZE="2">
<SPAN ID="er"> </SPAN>
</FONT>
<P>
<TABLE COLS="2" BORDER="3" CELLPADDING="3" CELLSPACING="3">
```

```
      <TR>
            <TD>NUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Nume" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>PRENUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Prenume" SIZE="20"></TD>
      </TR>
      <TR>
            <TD>TELEFON:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Telefon" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>Tipul de utilizator:</TD>
            <TD>
                  <SELECT NAME="Tip">
                  <OPTION VALUE="ADMINISTRATOR" <% if Tip = "ADMINISTRATOR"✂
Then Response.Write "SELECTED"%>>Administrator
                  <OPTION VALUE="VIZITATOR" <% if Tip = "VIZITATOR" Then✂
Response.Write "SELECTED"%>>Vizitator
                  <OPTION VALUE="UTILIZATOR" <% if Tip = "UTILIZATOR" Then✂
Response.Write

"SELECTED"%>>Utilizator

            </TD>
      </TR>
      <TR>
            <TD ALIGN="CENTER"> <INPUT TYPE="SUBMIT" VALUE="Trimite"></TD>
            <TD ALIGN="CENTER"> <INPUT TYPE="BUTTON" VALUE="Reset"✂
ONCLICK="Sterge()"></TD>
      </TR>
</TABLE>
</FORM>
</BODY>
```


Spre deosebire de primul caz, aici toate verificările se fac pe client, iar transferul către server se face numai în cazul în care restricţiile impuse valorilor introduse sunt îndeplinite. Dezavantajul este acela că trebuie folosit puţin **JavaScript** pentru ca să poată funcţiona corect. Funcţia scrisă în **JScript** este:

```
function TestJS()
{
      return TestVB();
}
```

Observaţi că nu face altceva decât întoarce valoarea funcţiei `TestVB` care este scrisă în `VBScript`. În paragraful Transferul (submit) formularelor s-a prezentat modul de folosire a evenimentului de formular `ONSUBMIT` pentru anularea transferului către server. În condiţii normale acestui eveniment i se asociază o secvenţă de cod în **JScript**, în exemplul aceasta fiind `return(TestJS());`. Dacă valoarea întoarsă este `True` conţinutul formularului este transferat către server. Altfel, transferul este suspendat şi se afişează mesaje cu privire la erorile ce trebuie corectate. Întrucât aici se aşteaptă un `True` sau `False` generat din **JScript** a fost scrisă o funcţie care întoarce una dintre cele două valori obligatorii. Observaţi că există şi o altă funcţie nouă numită `Sterge` care are rolul de a şterge mesajele de eroare şi conţinuturile controalelor la apăsarea butonului **Reset** din formular.

Colecţia Request.QueryString

O metodă de transfer a datelor între client şi server este prin colecţia `Request.QueryString`.

Aceste date se transferă împreună cu cererea URL-ului sau, mai corect, ca o parte a acestuia după cum s-a discutat deja în Părţile componente ale URL. În cazul URL-ului,

```
http://www.unsite.ro/Programe/default.htm?Pagina=1&Paragraf=2
```

avem doi parametri, care vor deveni două articole ale colecţiei `Request.QueryString`. Ei mai pot fi găsiţi şi în variabila `QUERY_STRING`, dar într-o formă mai puţin accesibilă. La trimiterea datelor în acest caz metoda de transfer trebuie să fie `GET`. Variabila `REQUEST_METHOD` poate fi folosită pentru a verifica metoda de transfer folosită. Motorul ASP este cel care, prin colecţia `Request.QueryString`, analizează şirul primit şi îl transformă în perechi de tipul `nume=valoare`. Numele, în colecţia `Request.QueryString`, devin chei. Pentru extragerea unui articol individual din colecţie se poate folosi indicele sau numele cheii după cum urmează:

```
Dim V
V=Request.QueryString(1)      ' primul articol din colectie
V=Request.QueryString("Tip")  ' aici Tip este numele cheii
```

Pentru parcurgerea întregii colecţii se va folosi `For Each ...Next` după cum urmează:

```
Dim V
For Each V In Request.QueryString
      Response.Write V & "=" & Request.QueryString(V)
Next
```


Aplicaţia care urmează are două fişiere, `aspre_tr1.asp` şi `aspre_qst.asp`. Codul celor două fişiere se prezintă în continuare.

Fişierul `aspre_tr1.asp`:

```
<HTML>
<BODY>
<FORM ACTION="asprq_qst.asp" METHOD="GET">
<H3>Introdu datele tale:</H3>
<TABLE COLS="2" BORDER="2">
      <TR>
            <TD>NUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Nume" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>PRENUME:</TD>
            <TD> <INPUT TYPE="TEXT" NAME="Prenume" SIZE="20"></TD>
      </TR>

      <TR>
            <TD>Tipul de utilizator:</TD>
            <TD>
                  <SELECT NAME="Tip">
                  <OPTION VALUE="ADMINISTRATOR">Administrator
                  <OPTION VALUE="VIZITATOR">Vizitator
                  <OPTION VALUE="UTILIZATOR">Utilizator
            </TD>
      </TR>

      <TR>
            <TD ALIGN="CENTER"> <INPUT TYPE="SUBMIT" VALUE="Trimite"></TD>
            <TD ALIGN="CENTER"> <INPUT TYPE="RESET" VALUE="Reset"></TD>
      </TR>
</TABLE>
</FORM>
```

```
</BODY>
</HTML>
```

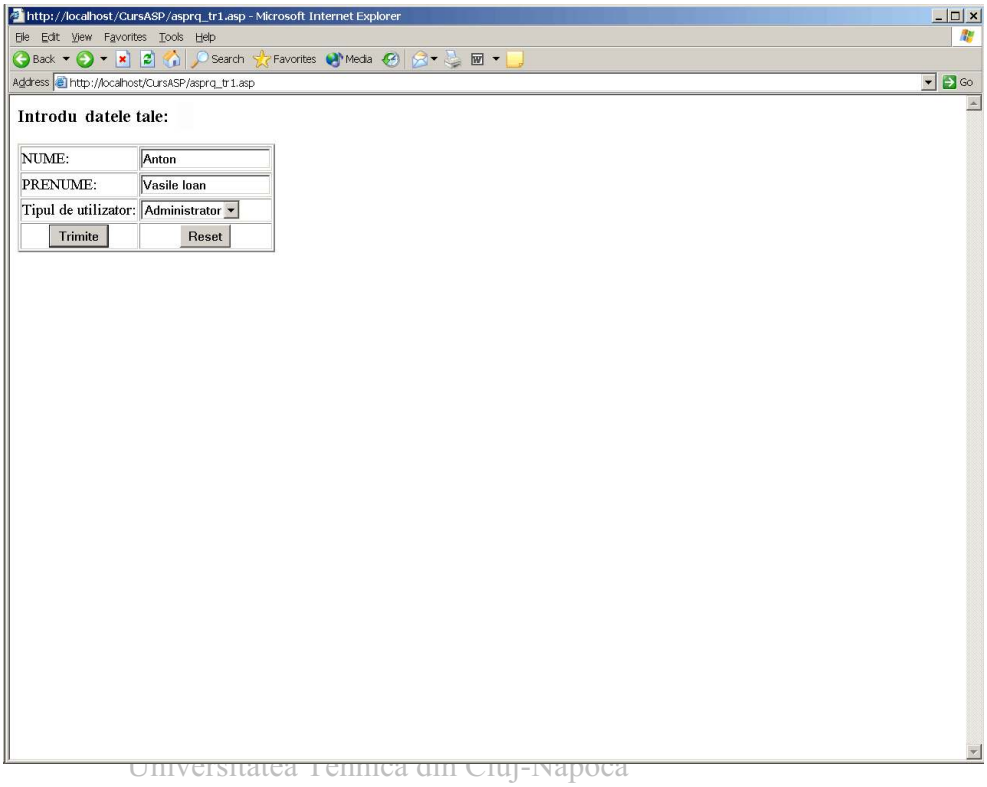


Figure 47 Fişierul asprq_tr1.asp

Fişierul `aspre_qst.asp`:

```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim V
      If Request.ServerVariables("REQUEST_METHOD") = "GET" Then
            For Each V In Request.QueryString
                  Response.Write V & "=" & Request.QueryString(V) & "<BR>"
            Next
      End If
%>
<HR>

<%
      Response.Write "QUERY_STRING = " &✂
 Request.ServerVariables("QUERY_STRING") & "<BR>"
%>
</BODY>
```

Datorită folosirii metodei de transfer `GET` la apăsarea butonului **Trimite**, afişat în **Figura 47** URL-ul, devine:

```
http://localhost/CursASP/asprq_qst.asp?Nume=Anton&Prenume=Vasile+Ioan&Tip= ✂
ADMINISTRATOR
```

Rezultatele sunt afişate în **Figura 48**.

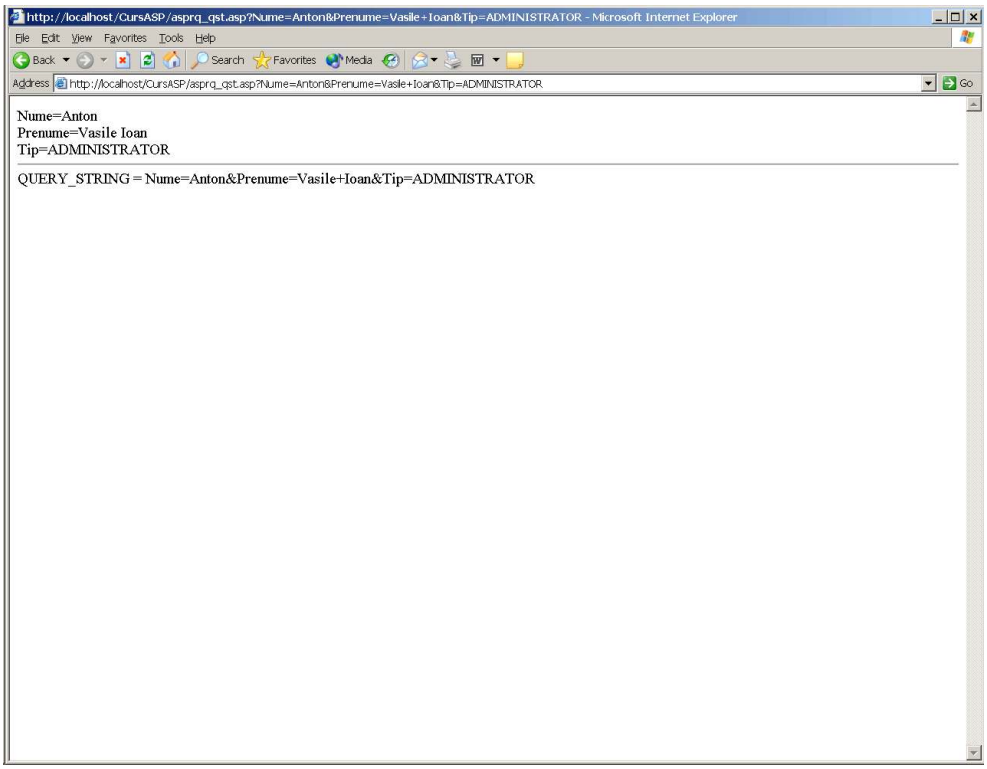


Figure 48
Fişierul asprq_qst.asp

Colecţia Request.ServerVariables

Colecția `Request.ServerVariables` conține informații standard trimise automat de navigator la fiecare cerere. Aplicația care urmează permite studiul acestora cu rezerva că lista este dependentă de versiunea de IIS şi, parțial, de locul în care s-a navigat înainte cu navigatorul.

```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>Lista de variabile Request.ServerVariables</TITLE>
</HEAD>
<BODY>
<H3 ALIGN="CENTER">Lista de variabile Request.ServerVariables</H3>
<TABLE WIDTH="90%" ALIGN="CENTER" BORDER="1" COLS="3">
      <THEAD>
            <TR>
                  <TH> </TH>
                  <TH>Numele cheii</TH>
                  <TH>Valoarea</TH>
            </TR>
      </THEAD>
      <TBODY>
      <%
            Dim Sv
            Dim c
            c=1
            For Each Sv In Request.ServerVariables
                  With Response
                        .Write "<TR>"
                        .Write "<TD>"
                        .Write c & ") "
                        .Write "</TD>"
                        .Write "<TD>"
                        .Write Sv
```

```
                        .Write "</TD>"
                        .Write "<TD>"
                        .Write Request.ServerVariables(Sv) & "<BR>"
                        .Write "</TD>"
                        .Write "</TR>"
                    End With
                    c=c+1
            Next
      %>
      </TBODY>
</TABLE>
</BODY>
</HTML>
```

Lista de variabile din colecția `Request.ServerVariables` afişată de codul asp de mai sus este:

Numele cheii	Valoarea
ALL_HTTP	HTTP_ACCEPT:*/* HTTP_ACCEPT_LANGUAGE:en-us HTTP_CONNECTION:Keep-Alive HTTP_HOST:localhost HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) HTTP_COOKIE:ASPSESSIONIDQQQGQTMC=IHMBNHFDPPKMDGPGEDLEAMOC HTTP_ACCEPT_ENCODING:gzip, deflate
ALL_RAW	Accept: */* Accept-Language: en-us Connection: Keep-Alive Host: localhost User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Cookie:ASPSESSIONIDQQQGQTMC=IHMBNHFDPPKMDGPGEDLEAMOC Accept-Encoding: gzip, deflate
APPL_MD_PATH	/LM/W3SVC/1/Root/CursASP
APPL_PHYSICAL_PATH	C:\CursASP\
AUTH_PASSWORD	
AUTH_TYPE	
AUTH_USER	
CERT_COOKIE	
CERT_FLAGS	
CERT_ISSUER	
CERT_KEYSIZE	
CERT_SECRETKEYSIZE	
CERT_SERIALNUMBER	
CERT_SERVER_ISSUER	
CERT_SERVER_SUBJECT	
CERT_SUBJECT	
CONTENT_LENGTH	0

Numele cheii	Valoarea
CONTENT_TYPE	
GATEWAY_INTERFACE	CGI/1.1
HTTPS	off
HTTPS_KEYSIZE	
HTTPS_SECRETKEYSIZE	
HTTPS_SERVER_ISSUER	
HTTPS_SERVER_SUBJECT	
INSTANCE_ID	1
INSTANCE_META_PATH	/LM/W3SVC/1
LOCAL_ADDR	127.0.0.1
LOGON_USER	
PATH_INFO	/cursasp/asprq1.asp
PATH_TRANSLATED	C:\CursASP\asprq1.asp
QUERY_STRING	
REMOTE_ADDR	127.0.0.1
REMOTE_HOST	127.0.0.1
REMOTE_USER	
REQUEST_METHOD	GET
SCRIPT_NAME	/cursaspasprq1.asp
SERVER_NAME	localhost
SERVER_PORT	80
SERVER_PORT_SECURE	0
SERVER_PROTOCOL	HTTP/1.1
SERVER_SOFTWARE	Microsoft-IIS/5.1
URL	/cursasp/asprq1.asp
HTTP_ACCEPT	*/*
HTTP_ACCEPT_LANGUAGE	en-us
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	localhost
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Numele cheii	Valoarea
HTTP_COOKIE	ASPSESSIONIDQQQGQTMC=IHMBNHFDPPKMDGPGEDLEAMOC
HTTP_ACCEPT_ENCODING	gzip, deflate

În funcție de tipul de aplicație care se scrie, unele dintre ele pot să fie utile. Câteva dintre cele pe care le-am folosit în aplicații le descriu în tabelul următor:

Cheie	Descriere
APPL_PHYSICAL_PATH	Calea către aplicația rulată, nu include numele aplicației. Valoarea se utilizează pentru ascunderea locului în care este stocată aplicația.
LOCAL_ADDR	Adresa IP a server-ului pe care se ține aplicația.
LOGON_USER	Numele de utilizator de rețea al navigatorului care cere resursa (numai dacă opțiunea de securitate IIS Allow Anonymous este inactivă).
PATH_INFO	Calea fizică spre fișierul cerut. Utilă dacă se face redirectare, însă trebuie să se cunoască numele inițial al fişierului care a fost cerut.
SERVER_NAME	Numele server-ului cerut de utilizator.
SERVER_PORT	Adresa de port IP, pentru server-e HTTP este, tipic, 80.

Obiectul Application

Terminologie

- *Sistem de operare (SO)* - un pachet de programe de nivel jos care planifică sarcinile (`task`-urile), alocă spațiul de stocare, realizează interfațarea cu perifericele şi prezintă o interfață implicită fiecărui utilizator atunci când nu se rulează un program aplicație. SO este deseori divizat în: nucleu (`kernel` - responsabil cu alocarea resurselor, interfațarea hardware de nivel jos, securitate etc.) care este întotdeauna prezent şi programe de sistem care folosesc facilități ale nucleului pentru realizarea unor sarcini de întreținere la un nivel mai înalt. Unele SO au interfețe grafice cu utilizatorul şi un sistem de ferestre, altele nu au aceste posibilități. Facilitățile sistemului de operare şi filozofia lui generală de proiectare au o influența puternică asupra stilului de programare şi a culturilor tehnice care se dezvoltă în legătură cu maşinile pe care acesta este rulat.
- *Job* - Denumire generică dată tuturor activităților implicate în realizarea unui proiect, de la început până la sfârşit, pe calculator. Un job poate include mai multe procese şi programe. În zilele de azi termenul este învechit el fiind legat de vremurile când un job putea fi definit printr-un fişier de comenzi ce conținea linii de program sursă intercalate cu instrucțiuni pentru controlul job-urilor în vederea parcurgerii unor multitudini de faze cum sunt compilarea, legarea, rularea şi tipărirea.
- *Multitasking* (alte denumiri: *multi-processing*, *multiprogramming*, *concurență*) - o tehnică folosită de SO pentru partajarea unui singur procesor între mai multe job-uri independente. Un SO multitasking trebuie să aibă un oarecare mod de protecție a unui task față de un altul. Astfel, un task nu poate interacționa într-un mod accidental sau neaşteptat cu un altul, în vederea modificării unor zone de memorie care nu-i aparțin. Job-

urile într-un SO multitasking pot aparține unuia sau mai multor utilizatori (*multiuser*). O variantă de multitasking mai simplă şi cu mai puține restricții între taskuri este multithreading-ul.

- *Session* (*sesiune*)- intervalul de timp în care un utilizator interacționează continuu cu o aplicație.

Din punctul de vedere al Web server-ului fiecare director virtual se consideră o aplicație. În ASP, fiecare aplicație se va referi prin obiectul `Application`. Obiectul este folosit pentru manipularea informațiilor care trebuie accesate de către toți utilizatorii unei aplicații. El este instanțiat atunci când primul utilizator vizitează directorul virtual din momentul pornirii Web server-ului. Obiectul poate fi folosit de către toți utilizatorii care vizitează site-ul şi rămâne funcțional până când toate sesiunile de lucru ale utilizatorilor expiră sau până la oprirea Web server-ului. El stochează perechi `cheie=valoare` la fel cu o parte dintre obiectele deja discutate, cu marea diferență că toți utilizatorii unui site vor manipula acelaşi obiect `Application`, în timp ce toate celelalte colecții sunt specifice unui anumit utilizator, fiind create de motorul ASP pentru o singură tranzacție sau sesiune de lucru. Deoarece toate sesiunile ASP partajează acelaşi obiect `Application` programatorul trebuie să fie familiarizat cu problemele potențiale caracteristice partajării datelor într-un mediu multiuser.

Deseori, Web server-ele trebuie să trateze cereri simultane, venite de la mai mulți utilizatori. Pentru urmărirea şi rezolvarea acestor cereri Web server-ul le stochează într-o coadă (`queue`). O coadă este o listă în care un articol nou se adaugă la un capăt, iar un articol se extrage de la celălat capăt. IIS stochează task-urile într-o coadă, apoi începe prelucrarea lor, dar nu trece la un task nou decât după ce-l termină pe cel curent. IIS face multitasking cu divizarea în timp a duratei de lucru a procesorului. Numărul de task-uri simultane pe care le poate prelucra depinde de numărul de thread-uri atribuite IIS-ului. Un thread este un singur task, dar un program poate folosi un singur thread sau mai multe în timpul execuției lui. Atunci când se creează mai multe thread-uri, calculatorul trebuie să poată comuta între ele. În acest scop el trebuie să salveze şi să refacă setarea fiecărui task pe măsură ce realizează comutarea. Procedura de salvare şi refacere a stării unui task este consumatoare de timp, încărcarea procesorului crescând odată cu creşterea numărului de task-uri, în timp ce perioada pentru execuția unui task rămâne constantă. În acest fel timpul de calcul, partea activă a task-urilor, scade. Din acest punct de vedere, IIS încearcă să stabilească un echilibru optim între timpul de execuție rapidă a unei cereri şi durata de prelucrare a mai multor cereri.

Variabile ale aplicaţiei - Application

Într-o aplicație clasică, variabilele globale sunt acelea care pot fi accesate şi modificate din orice porțiune de cod. Obiectul `Application` permite stocarea unor variabile globale aplicației ce vor putea fi accesate din orice pagină a aplicație şi de către oricare utilizator al acesteia. Variabilele globale se prezintă sub forma unei liste de articole, cu nume stocate în colecția `Contents`. Pentru adăugarea unui articol nou în listă se foloseşte sintaxa:

```
Application.Value("cheie")="valoare"
```

Deoarece proprietatea `Value` este cea implicită în locul liniei de mai sus se poate scrie:

```
Application("cheie")="valoare"
```

Extragerea unei valori stocate în listă se face prin:

```
Variabila = Application.Contents("cheie")
```

sau prin:

```
Variabila = Application.Value("cheie")
```

La fel ca şi în cazul celorlalte colecţii valorile pot fi accesate şi prin index:

```
Variabila = Application.Contents(1)
Variabila = Application.Value(1)
```

Pentru ştergerea valorii unui articol din listă pot fi folosite următoarele variante:

```
Application.Contents("cheie")=""
Application.Value("cheie")=Empty
Application.Value("cheie")=vbNullString
```

NOTĂ

Ştergerea articolelor funcţionează corect numai începând cu ASP 3 deşi, Microsoft, în documentaţia MSDN, care vine pentru Visual Studio 6 susţine că aceasta este posibilă şi în versiunile anterioare (din cele scrise acolo nu funcţionează metodele `Remove` **şi** `RemoveAll` **ale colecţiei** `Contents`**).**

Pentru afişarea tuturor articolelor din colecţia `Contents` se poate folosi secvenţa de cod:


```
For i=1 to Application.Contents.Count
      Response.Write Application.Contents(i) & "<BR>"
Next
```


În exemplul care urmează se prezintă modul în care se pot crea şi iniţializa un grup de articole ale colecţiei `Contents` din `Application`. În pagina `aspapp1.asp` se realizează crearea dinamică a şapte variabile globale care vor fi accesibile pentru orice pagină a aplicaţiei şi pentru toţi utilizatorii acestora. Apoi, valorile acestor variabile vor fi incrementate şi în final, afişate. Tehnica de incrementare prezentată este des folosită pentru a număra utilizatorii care s-au legat la o aplicaţie. După lansarea primei pagini, se va lansa şi pagina `aspapp2.asp` . Observaţi că aceasta va cunoaşte toate variabilele globale create în prima pagină şi va afişa valorile lor actualizate conform ultimelor modificări aduse în prima pagină.

Fişierul `aspapp1.asp`:

```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim i
      For i=1 to 7
            Application("Cheie"&i)=i 'Valoarea cheii
      Next

      Response.Write "<P>Valorile dupa initializare:<BR>"
      For i=1 to Application.Contents.Count
            Response.Write Application.Contents(i) & "<BR>"
      Next

      For i=1 to Application.Contents.Count
```

```
            Application.Contents(i) = Application.Contents(i) + 1
      Next
      Response.Write "<P>Valorile dupa modificare:<BR>"
      For i=1 to Application.Contents.Count
            Response.Write Application.Contents(i) & "<BR>"
      Next

      Application.Contents("Cheie3")=""
      Application.Contents("Cheie5")=Empty
      Response.Write "<P>Valorile dupa a doua modificare:<BR>"
      For i=1 to Application.Contents.Count
            Response.Write Application.Contents(i) & "<BR>"
      Next

%>
</BODY>
</HTML>
```

Fişierul `aspapp2.asp`:

```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim i

      Response.Write "<P>Valorile aplicatiei intr-o noua pagina<BR>"
      For i=1 to Application.Contents.Count
            Response.Write Application.Contents(i) & "<BR>"
      Next

%>
</BODY>
</HTML>
```


Metodele Application.Lock şi Application.Unlock

Fiecare aplicație IIS are un singur obiect `Application` care este partajat între toți utilizatorii. Orice sesiune de lucru a utilizatorului cu aplicația poate accesa valorile obiectului `Application`, dar este de dorit ca numai o singură sesiune să poată modifica valorile la un moment dat. Din acest motiv obiectul `Application` poate fi blocat (`lock`) înainte de adăugarea sau modificarea unei valori, apoi deblocat (`unlock`) după ce s-au terminat modificările dorite. În lipsa blocării obiectului `Application` este posibil ca mai mulți utilizatori, prin script-urile paginilor încărcate, să încerce modificarea simultană a unor valori stocate în el. Această încercare va genera erori. Implicit, dacă se uită deblocarea, la terminarea paginii aceasta ar trebui să se facă automat. Dacă un script încearcă să acceseze obiectul `Application` şi el este blocat de un alt proces, va aştepta până când obiectul `Application` va fi deblocat. Dacă script-ul nu poate accesa obiectul `Application` mai mult timp, după o perioadă de aşteptare, motorul ASP va genera un cod de eroare. Secvența de cod care urmează permite blocarea şi deblocarea obiectul `Application`:

```
Application.Lock
Application("cheie")=valoare
Application.Unlock
```

Aplicația `aspapp3.asp`, ce va fi prezentată în continuare, realizează contorizarea numărului deschiderilor paginii. În **Figura 49**, se observă că pagina (aplicația) este deschisă, local, de cinci ori, în fiecare frereastră fiind afişat numărul deschiderilor curente.

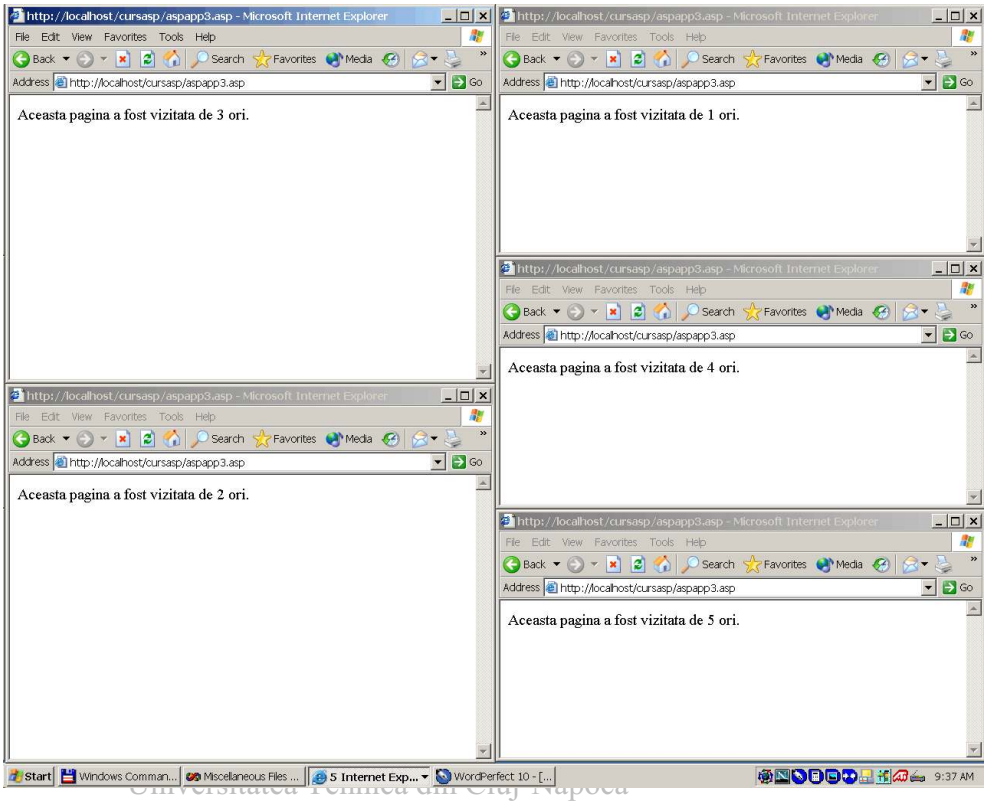


Figure 49 Contorizarea numărului deschiderilor aplicației aspapp3.asp


```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
Dim contViz
Application.Lock
Application("Contor") = Application("Contor") + 1
contViz = Application("Contor")
Application.Unlock
Response.Write "Aceasta pagina a fost vizitata de " & contViz & " ori."
%>
</BODY>
</HTML>
```

Obiectul Session

Unul dintre cele mai importante obiecte ASP este `Session`. ASP realizează menținerea stării unui anumit utilizator, iar obiectul `Session` va rămâne vizibil tot timpul sesiunii utilizatorului. El stochează valori asociate unui navigator, obiectul `Session` este asemenea celui `Application` cu excepția că o singură sesiune are accesul exclusiv la acesta. Dacă la un moment dat sunt 100 de utilizatori pe o pagină, vor exista 100 de obiecte `Session`, distincte, care rulează. Obiectul `Session` permite crearea unor proprietăți globale personale unui anumit utilizator care vor putea fi folosite numai de acel utilizator. Astfel, apare posibilitatea creării unor pagini HTML individualizate în mod dinamic, acesta fiind unul dintre motivele pentru care a fost creat ASP. Un nou obiect `Session` este creat de fiecare dată când un nou utilizator vizitează site-ul de Web şi este distrus atunci când el părăseşte site-ul sau dacă apare inactivitate cu durată de timp mai mare decât cea setată în proprietatea de expirare a obiectului `Session`. O sesiune ASP este specifică unui cookie dat. Relația produce puțină confuzie deoarece un cookie şi un navigator sunt

entități distincte. Trebuie reținut că motorul ASP identifică sesiunile prin căutarea cookie-ului `ASPSESSIONID`. Ca urmare a modului de identificare a sesiunilor folosit de motorul ASP pot să apară probleme în cazul în care navigatorul este setat să refuze cookie-urile.

Fazele unei sesiuni

Începutul sesiunii

O sesiune ASP începe atunci când navigatorul cere o pagină ASP stocată într-un director virtual (care este o aplicație IIS). Nu contează care pagină a directorului virtual este cerută, tot ceea ce contează este ca aceasta să fie o pagină cu extensia ASP, stocată direct în directorul virtual sau într-un subdirector. Server-ul redirectează imediat utilizatorul către fişierul cu numele `global.asa`, generează antetul de cookie şi rulează codul corespunzător (dacă există) evenimentului `Session_OnStart` din acest fişier. Cheia de cookie este de forma `ASPSESSIONIDXXXXXXXX` , iar valoarea ei este o serie încriptată de litere şi cifre pseudo-aleatoare. Un exemplu ar fi :

```
ASPSESSIONIDQQQGQTMC=IHMBNHFDPPKMDGPGEDLEAMOC
```

Identificarea sesiunii

Tranzacțiile HTTP nu păstrează starea. Acesta este motivul pentru care server-ul asociază fiecărui navigator un cod de identificare unic în răspunsul la prima cerere formulată de navigator. Datorită lui navigatorul se poate identifica pe server, în cazul cererilor ulterioare. Deci, fiecare navigator primeşte un cookie `SessionID` cu o valoare corespunzătoare. Orice informație stocată pe server legată de un anumit navigator va conține şi o copie a valorii lui `SessionID`. Când navigatorul cere o nouă pagină, server-ul va găsi informațiile legate de navigator mai repede pe baza valorii `SessionID` deja stocate. Fiecare valoare a cookie-ului de indentificare este unică între două opriri ale IIS-ului. Dacă navigatorul acceptă cookie-ul, el va fi trimis tuturor cererilor ulterioare ale acelui site. Motorul ASP stochează lista cookie-urilor. Când navigatorul trimite cookie-ul `ASPSESSIONID`, motorul ASP caută valoarea corespunzătore din listă. Dacă aceasta există, motorul poate asocia un obiect `Session` existent unui navigator prin căutare în lista obiectelor `Session`. Deci, motivul pentru care cookie-ul `SessionID` este atât de important constă în indentificarea, de către server, a navigatorului, în vederea stocării şi extragerii unor valori specifice navigatorului. În lipsa lui, navigatorul rămâne necunoscut şi nu va putea fi identificat în cazul unei vizite ulterioare. Astfel, server-ul nu va putea extrage valorile specifice acelui navigator.

Terminarea sesiunii

Terminarea unei sesiuni se poate realiza în următoarele feluri:

- Server-ul abandonează sesiunea după scurgerea unui interval de timp de inactivitate. Durata intervalului se poate defini prin proprietatea `Session.Timeout`.
- În codul ASP s-a ajuns la metoda `Session.Abandon`.
- IIS a fost oprit, motiv pentru care toate aplicațiile şi sesiunile corespunzătoare se opresc şi ele.
- S-a modificat fişierul `global.asa`. După salvarea lui, următoarea cerere adresată server-ului va forța IIS să oprească aplicația, motiv pentru care toate sesiunile vor fi închise şi ele.
- Navigatorul refuză cookie-ul `SessionID`. Toate navigatoarele permit alegerea acestei variante. Activarea ei însă nu opreşte server-ul din crearea unei noi sesiuni (asta pentru că server-ul nu are de unde să ştie, în avans, că navigatorul refuză cookie-urile), însă va opri sesiunea să continue după prima cerere.

Proprietatea Session.TimeOut

Atunci când server-ul creează cookie-ul `SessionID`, el mai creează şi îi asociază şi o durată de existență. Implicit, această durată este de 10 minute pe IIS 5. Se poate însă specifica şi o altă valoare folosind proprietatea `Session.TimeOut`. Valoarea stocată, sau cea implicită, descreşte până la 0, moment în care sesiunea este terminată de ASP, toate informațiile specifice ei fiind pierdute. Motorul ASP resetează durata de existență de fiecare dată când primeşte o cerere de pagină din partea navigatorului cu un anumit `SessionID`. Din acest motiv, atât timp cât navigatorul cere pagini noi, sesiunea va rămâne activă.

Scopul proprietății este cel de protejare a server-ului contra stocării unor date asociate unui navigator pe o durată prea mare. Ideal ar fi ca server-ul să ştie de undeva că sesiunea s-a terminat, dar din cauza că HTTP nu permite stocarea stărilor, acest lucru nu se poate întâmpla.

Un caz nefericit apare atunci când, ca urmarea a unei valori setate în `Session.TimeOut`, cookie-ul `SessionID` expiră, dar navigatorul continuă să-i mai stocheze valoarea. În acest caz `SessionID`-ul avut de navigator nu mai este valid şi toate datele asociate sesiunii nu mai există. ASP va permite ca navigatorul să păstreze aceeaşi valoare de cookie `SessionID`, dar aceasta va fi asociată unui nou obiect `Session`.

Metoda Session.Abandon

Metoda permite controlul parțial al utilizatorului asupra sesiunii de lucru cu aplicația. În general, ea se va pune pe unul dintre butoanele de părăsire a sesiunii. Atunci când utilizatorul face clic pe buton sesiunea va fi terminată. Terminarea nu este instantanee, ci este întârziată, având loc numai după prelucrarea paginii curente şi a instrucțiunilor din metoda `Session_OnEnd` a fişierului `global.asa`.

Fişierul global.asa

Fişierul `global.asa` are o utilizarea specială în ASP. El trebuie să fie plasat în rădăcina directorului virtual şi gestionează evenimentele obiectelor `Application` şi `Session`. El mai poate fi utilizat pentru crearea de variabile `Application` şi `Session`. Obiectele `Application` şi `Session` au evenimente declanşate de fazele de pornire şi de terminare a lor. Motorul ASP va rula secvențele de cod corespunzătoare evenimentelor din tabelul următor:

Momentul în care apare	Denumire eveniment
Pornirea aplicației	`Application_OnStart`
Pornirea sesiunii	`Session_OnStart`
Terminarea sesiunii	`Session_OnEnd`
Terminarea aplicației	`Application_OnEnd`

Codul corespunzător celor patru evenimente se scrie în procedurile corespunzătoare de tratare a evenimentelor. Nu este obligatorie scrierea de cod pentru aceste evenimente. Toate secvențele de cod script trebuie însă să fie cuprinse între marcajele `<SCRIPT LANGUAGE=VBScript RUNAT=Server> </SCRIPT>`. În afara tratării acestor evenimente, `global.asa`, mai poate conține definiții de obiecte statice, prin folosirea marcajului `<OBJECT> </OBJECT>`, la nivelul obiectului `Application`, `Session` sau la nivel de bibliotecă de tipuri. O bibliotecă de tipuri (`Type Library`) este un fişier care conține o listă de proprietăți şi de metode. ASP foloseşte bibliotecile de tipuri pentru a forma lista numelor de metode, proprietăți şi tipuri de parametri la care un obiect ştie să răspundă. **VB** şi **Visual InterDev** folosesc aceste informații de *Intellisense* la

afişarea listei pentru completarea automată a codului.

Motorul ASP parcurge etapele de iniţializare pentru aplicaţie şi sesiunile de lucru după cum urmează:

- Primul utilizator care doreşte să acceseze o pagină a directorului virtual face ca motorul ASP să găsească fişierul `global.asa` în directorul rădăcină al aplicaţiei. Server-ul va parcurge aceleaşi etape indiferent dacă fişierul `global.asa` există sau nu.
- Imediat după aceasta se creează obiectul `Application`, iar ASP lansează procedura `Application_OnStart` (dacă aceasta există).
- După terminarea evenimentului `Application_OnStart`, motorul ASP creează un nou `SessionID`, scrie antetul de cookie `SessionID`, apoi creează un nou obiect `Session` şi porneşte cronometrul de expirare pe baza valorii din `Session.Timeout`.
- Imediat după crearea noului obiect `Session`, este apelată procedura de tratare a evenimentului `Session_OnStart` (dacă aceasta există).
- La expirarea sesiunii, sau la apelarea metodei `Session.Abandon`, motorul ASP apelează procedura de tratare a evenimentului `Session_OnEnd` (dacă aceasta există), apoi distruge obiectul `Session`.
- După ce motorul ASP distruge ultimul obiect `Session`, el apelează procedura de tratare a evenimentului `Application_OnEnd` (dacă aceasta există).

Fişierul mai poate fi folosit pentru iniţializarea variabilelor globale şi pentru colectarea de meta-informaţii cu privire la aplicaţie. Nu este obligatorie iniţializarea variabilelor globale în `global.asa`, aceasta putându-se face şi de pe prima pagină a aplicaţiei.

Una dintre utilizările curente ale fişierului `global.asa` este iniţializarea şirului de conectare la o baza de date, deoarece administratorul server-ului poate modifica numele sau locul de stocare al bazei de date. Înscrierea acestora în cod obligă ca, la fiecare acţiune de acest fel din partea administratorului, să modificăm anumite porţiuni şi ale codului ASP. Întreţinerea paginii devine mai greoaie din acest motiv. Dacă acest şir este stocat în `global.asa` eventualele modificări trebuie operate numai asupra acestui fişier. O aplicaţie formată din fişierele `global.asa` şi `aspses1.asp` se prezintă în continuare. Scopul ei este de a prezenta modalitatea de contorizare a vizitatorilor unui site şi de a arăta modul de lucru cu obiectele `Application` şi `Session`.

Fişierul `global.asa`:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
'*** Uneori acest fisier va trebui modificat ***
'contVizit - variabila globala care contorizeaza nr. de utilizatori
'startApp - variabila globala care tine data si ora pornirii aplicatiei
'startVizit - variabila locala care tine data si ora pornirii sesiunii
</SCRIPT>

<SCRIPT LANGUAGE=VBScript RUNAT=Server>

Sub Application_OnStart
  Application("contVizit")=1
  Application("startApp") = Now
End Sub

Sub Session_OnStart
   Session("startVizit") = Now
End Sub
```

```
Sub Session_OnEnd
   If Application("contVizit") > 0 Then
         Application.Lock
         Application("contVizit") = Application("contVizit") + 1
         Application.Unlock
   End If
End Sub
</SCRIPT>
```

Fişierul `aspses1.asp`:

```
<%@ Language=VBScript %>
<% Response.Expires = -1%>
<HTML>
<HEAD>
</HEAD>
<BODY>
<H3>Exemplu de lucru cu obiectul Session</H3>
<P>
<%
Response.write "Valoarea lui SessionID este: " & Session.SessionID & "<BR>"
Response.write "Numarul de vizitatori este: " & Application("contVizit") &
"<BR>"
Response.write "Data si ora de incepere a sesiunii este: " &
Session("startVizit") & "<BR>"
Response.write "iar cea de pornire a aplicatiei este: " &
Application("startApp") & "<BR>"
Session.Abandon
%>
</P>
<P>Sesiunea a fost abandonata.<BR>
La fiecare F5 primele trei valori se vor modifica.<BR>
</BODY>
</HTML>
```

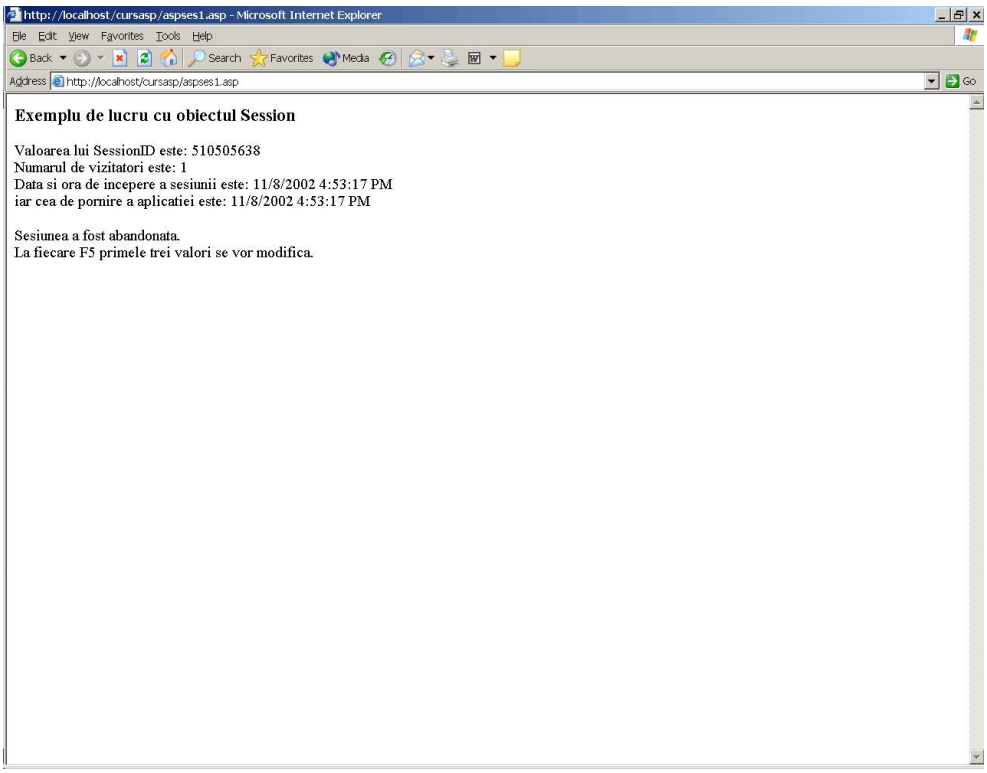



Figure 50 Aplicație cu obiectul Session

Rezultatele sunt afişate în **Figura 50**, la fiecare apăsare a tastei **F5** care duce la reîmprospătarea conținutului paginii. Primele trei valori se vor modifica. Este posibil ca prima valoare să nu se modifice dacă motorul ASP recunoaşte că navigatorul are un `ASPSESSIONID` valid, dar neutilizat, atunci când se reîmprospătează pagina.

Observați că `Session.Abandon` nu duce la abandonarea imediată a sesiunii deoarece textul care urmează după metodă este afişat corect în fereastra navigatorului.

Dacă, de exemplu, în locul contorizării numărului de vizitatori dorim să aflăm numărul sesiunilor active variabila `contVizit` va trebui să participe în `Session_OnStart` şi `Session_OnEnd` după cum urmează:

```
Sub Session_OnStart
   If Application("contVizit") > 0 Then
         Application.Lock
         Application("contVizit") = Application("contVizit") + 1
         Application.Unlock
   End If
End Sub

Sub Session_OnEnd
   Application("contVizit") = Application("contVizit") - 1
End Sub
```


Obiectul Server

Obiectul `Server` permite accesul la server-ul ASP. Prin folosirea metodelor şi proprietăților lui se pot crea obiecte, se pot rula secvențe de cod din alte fişiere ASP, se poate transforma calea virtuală în cea fizică şi redirecta la nivelul server-ului.

Proprietatea Server.ScriptTimeOut

Furnizorii de Internet permit rularea script-urilor pe server-ele lor deoarece acestea pot fi rulate, implicit, numai un timp de 90 de secunde. Motorul ASP are o proprietate prin care se controlează durata de rulare a unui script. Proprietatea poate fi setată separat pentru fiecare pagină prin `Server.ScriptTimeout = durata`, unde `durata` este specificată în secunde. O valoare normală este 20 de secunde, dar dacă server-ul are un trafic mai mare, se poate ca cele 20 de secunde să nu fie suficiente pentru generarea unui răspuns mai complex.

Metoda Server.HTMLEncode

Există secvențe de caractere care au semnificație specială (>, <, " etc.) pentru navigator. Pentru afişarea lor corectă de către navigator se poate folosi metoda `Server.HTMLEncode` după cum se observă în exemplul următor:

```
<%@ Language=VBScript %>
<HTML>
<BODY>
Exemplu de lucru cu "Server.HTMLEncode("<cod>")"<BR>
Exemplu de lucru cu &quot;Server.HTMLEncode(&quot;&lt;cod&gt;&quot;)&quot;<BR>
<%=Server.HTMLEncode("""Server.HTMLEncode(""<cod>"")""")%><BR>
</BODY>
```

În codul de mai sus prima afişare a liniei `"Server.HTMLEncode("<cod>")"` este greşită, urmează apoi două variante corecte. Pentru înțelegerea codului de mai sus trebuie să vă amintiți că `<%=` este scrierea prescurtată a lui `Reponse.Write`.

Metoda Server.URLEncode

Ştiți deja că folosirea metodei `GET` face ca datele să fie incluse în URL, apoi acestea pot fi extrase prin folosirea colecției `Request.QueryString`. Codificarea din URL nu este însă aceeaşi cu cea din HTML. Pentru formarea unui URL corect, în **VBScript**, se poate folosi metoda `Server.URLEncode`. Ea primeşte ca argument un şir **VBScript** normal pe care îl codifică în formatul specific unui URL. Fie URL-ul:

```
http://localhost/CursASP/asprq_qst.asp?Nume=Vasile&Prenume=Anton+Ioan&Telef
on=112233&Tip=ADMINISTRATOR
```

Să presupunem că folosim codul următor ca să afişăm URL-ul de mai sus pe ecranul navigatorului folosind codificarea HTML, apoi pe cea URL.

```
<%@ Language=VBScript %>
<HTML>
http://localhost/CursASP/asprq_qst.asp?Nume=Vasile&Prenume=Anton+Ioan&✂
Telefon=112233&Tip=ADMINISTRATOR<BR>
<%=Server.URLEncode("http://localhost/CursASP/asprq_qst.asp?Nume=Vasile&✂

Prenume=Anton+Ioan&Telefon=112233&Tip=ADMINISTRATOR")%>
</BODY>
```

În cazul codificării HTML şirul afişat este "normal", dar în cazul codificării URL se obține:


```
http%3A%2F%2Flocalhost%2FCursASP%2Fasprq%5Fqst%2Easp%3FNume%3DVasile%26Pren
ume%3DAnton%2BIoan%26Telefon%3D112233%26Tip%3DADMINISTRATOR
```


Metoda `HTMLEncode` se foloseşte atunci când dorim să afişăm în navigator simboluri pe care acesta le foloseşte în alte scopuri. Metoda `URLEncode` se foloseşte atunci când dorim să scriem în navigator date `QueryString`, de exemplu, pentru toate hiperlegăturile (marcajul `HREF`) care conțin caractere diferite de cifre, numere şi simboluri standard `QueryString` (`?`, `&` sau `+`). Deci, un şir care conține spații trebuie obligatoriu codificat.

Metoda Server.CreateObject

Una dintre facilitațile cele mai interesante ale ASP este posibilitatea de extindere. Metoda `CreateObject` permite lansarea în execuție a unui obiect COM. Microsoft furnizează motorul ASP cu mai multe obiecte ce pot fi utilizate în aplicații, dar programatorul poate crea şi propriile lui obiecte în limbajele **VB**, **VBScript** sau **JScript**. Sintaxa metodei este:

```
Server.CreateObject(NumeProiect.NumeClasa)
```

Aici, `NumeProiect.NumeClasa` se va înlocui cu numele corespuzător de proiect şi cel de clasă al obiectului ce se doreşte a fi creat.

Iată un exemplu de lucru cu componenta `Browser Capabilities`. Ea creează un obiect `BrowserType` care permite script-urilor să acceseze posibilitățile navigatorului client. Atunci când navigatorul se leagă la Web server trimite automat şi antetul HTTP al agentului utilizator care conține un şir ASCII de identificare a navigatorului şi a versiunii lui.

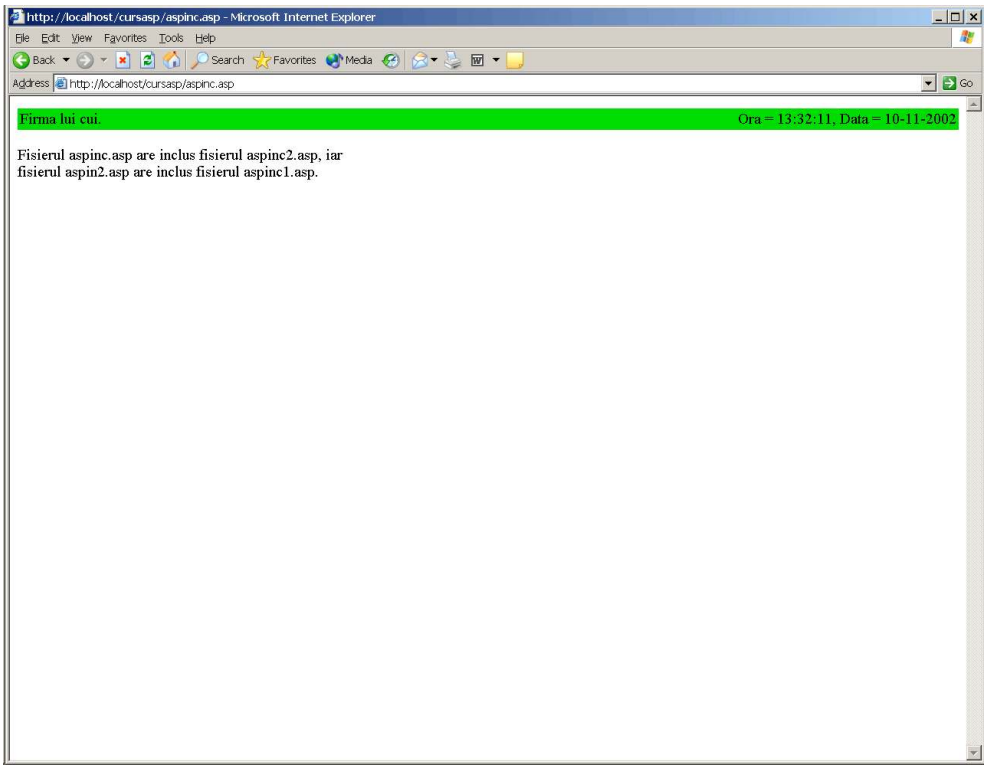
```
<%
htmlHttpUserAgent = Request.ServerVariables("HTTP_USER_AGENT")

Response.Write htmlHttpUserAgent
Response.Write "<BR>"
Response.Write "<BR>"
```

```
Set ServerulMeu= Server.CreateObject("MSWC.BrowserType")

If (InStr(htmlHttpUserAgent, "MSIE 6")) Then
      Session("Navigator") = "IE 6"
ElseIf (InStr(CStr(htmlHttpUserAgent), "Mozilla/3.0")) Then
      Session("Navigator") = "Netscape"
ElseIf (InStr(CStr(htmlHttpUserAgent), "Mozilla/4.0")) Then
      Session("Navigator") = "Netscape"
Else
      Response.Write "Navigator necunoscut<BR>"
      Session("Navigator") = ServerulMeu.Navigator
End If

If (InStr(htmlHttpUserAgent, "6")) Then
      Session("Versiune") = "6"
ElseIf (InStr(CStr(htmlHttpUserAgent), "6.0")) Then
      Session("Navigator") = "6.0"
Else
      Response.Write "Versiune necunoscuta<BR>"
      Session("Navigator") = ServerulMeu.Version
End If


If (InStr(CStr(htmlHttpUserAgent), "Windows 95")) Then
      Session("Platforma") = "Win95"
ElseIf (InStr(htmlHttpUserAgent, "Windows 98")) Then
      Session("Platforma") = "Win98"
ElseIf (InStr(htmlHttpUserAgent, "Win98")) Then
      Session("Platforma") = "Win98"
ElseIf (InStr(htmlHttpUserAgent, "Windows NT")) Then
      Session("Platforma") = "WinNT"
ElseIf (InStr(htmlHttpUserAgent, "Win32")) Then
      Session("Platforma") = "Win32"
ElseIf (InStr(htmlHttpUserAgent, "Windows 3.1")) Then
      Session("Platforma") = "Win16"
ElseIf (InStr(htmlHttpUserAgent, "Mac_PowerPC")) Then
      Session("Platforma") = "MacPPC"
ElseIf (InStr(htmlHttpUserAgent, "Mac_68000")) Then
      Session("Platforma") = "Mac68K"
Else
      Response.Write "Platforma necunoscuta<BR>"
      Session("Platforma") = ServerulMeu.Platform
End If

Response.Write "Navigator = " & Session("Navigator") & "<BR>"
Response.Write "Versiune = " & Session("Versiune") & "<BR>"
Response.Write "Sistem de operare = " & Session("Platforma")

Set ServerulMeu = Nothing
%>
```


Directiva include

Codul stocat în componente este într-un format executabil. Cel puțin în faza de testare este convenabil ca el să fie stocat, în format text ASCII, în fişiere ASP distincte, pentru ca să poată fi modificat cât mai uşor. Codul unui astfel de fişier poate fi inclus într-o aplicație ASP prin folosirea directivei `include`. Fişierul inclus poate conține funcții cu caracter general care vor fi apelate din anumite locuri în aplicație sau secvențe de cod care generează părțile fixe ale unui document ASP (de exemplu, un antet sau un sfârşit de document). Aplicația care urmează este formată din fişierele `aspinc.asp`, `aspinc1.asp`, `aspinc2.asp`, iar rezultatele rulării lui `aspinc.asp` se prezintă în **Figura 51**. Fişierul `aspinc1.asp` generează un antet, iar `aspinc2.asp` conține funcții de uz general pentru aplicație.

Figure 51 Aplicaţie cu directiva include



Fişierul `aspinc.asp`:

```
<%@ LANGUAGE=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>

<!-- #include file="aspinc2.asp" -->

<BR>
<P>
Fisierul aspinc.asp are inclus fisierul aspinc2.asp, iar<BR>
fisierul aspin2.asp are inclus fisierul aspinc1.asp.<BR>
</P
</BODY>
</HTML>
```

Fişierul `aspinc1.asp`:

```
<%
      'Functii de uz general pentru data si ora curenta
      'c - caracterul de separare

      Function Data(c)
            Data = Day(Now()) & c & Month(Now()) & c & Year(Now())
      End Function

      Function Ora(c)
            Ora = Hour(Now()) & c & Minute(Now()) & c & Second(Now())
      End Function
%>
```

Fişierul aspinc2.asp:

```
<!-- #include file="aspinc1.asp" -->

<TABLE ALIGN="CENTER" STYLE="POSITION:RELATIVE;TOP:1" BGCOLOR="#00DD00"✂
WIDTH="100%">
      <TR>
            <TD WIDTH = 50%>
                  Firma lui cui.
            </TD>
            <TD ALIGN="RIGHT">
                  Ora = <%=Ora(":")%>, Data = <%=Data("-")%>
            </TD>
      </TR>
</TABLE>
```

Forma generală a directivei este:

```
<!-- #include TipCale="NumeFisier" -->
```

unde TipCale poate fi FILE sau VIRTUAL. FILE se foloseşte pentru o cale absolută, iar VIRTUAL pentru directoarele virtuale ale server-ului.

Metoda Server.Execute

O variantă prin care directiva include poate fi evitată este Server.Execute. Metoda permite execuția unui cod stocat în fişier ASP după sintaxa:


```
Server.Execute "fisier.asp"
```

Codul din fisier.asp are acces la toate obiectele ASP intrinseci şi la toate obiectele instanțiate în pagina care conține metoda de execuție a lui fisier.asp. În cazul codului anterior se poate folosi în locul lui <!-- #include file="aspinc2.asp" --> linia Server.Execute "aspinc2.asp". În principiu, comparativ cu include, metoda Server.Execute , este mai flexibilă şi lucrează mai bine cu tranzacțiile.

Metoda Server.Transfer

S-a văzut deja că Response.Redirect forțează navigatorul să ceară o pagină diferită de cea cerută de noi, inițial, de la server. În versiunile vechi ale ASP aceasta era singura modalitate de a transfera execuția de la un fişier la altul. Ineficiența acestei metode era aceea că mesajul redirectat trebuia să călătorească de la sever la navigator, apoi înapoi la server. În plus, datorită codului ASP, trebuiau stocate şi eventualele date care țin de formulare sau de URL-uri pentru ca să fie disponibile pentru noul script. Metoda Server.Transfer vine să rezolve această problemă realizând o redirectare, direct de la nivelul server-ului. Această înseamnă că navigatorul cere un fişier, server-ul realizează redirectarea şi navigatorul primeşte ca rezultat conținutul noului fişier. Deoarece navigatorul nu este implicat în redirectare, metoda Server.Transfer menține conținutul formularelor şi datele QueryString, acestea fiind disponibile în script-ul către care s-a făcut redirectarea. Datorită eficienței, Server.Transfer, este de preferat în locul lui Response.Redirect.

Aplicația care urmează este formată din fişierele: asptr.asp, asptr1.asp, PrgNormal.asp, PrgSimbata.asp şi PrgDuminica.asp şi îşi propune să prezinte modul de lucru cu Server.Transfer pentru construirea unei pagini intermediare din care să se realizeze transferul către anumite pagini ale aplicației în funcție de cum au fost completate controalele formularului de selecție. În acest caz, dacă Request.Form("zi") este vidă după transfer utilizatorul ajunge

în acelaşi loc din care a plecat (sistem defensiv ne-selectiv). Altfel este trasferat, conform selecției, către o nouă pagină.

Fişierul `asptr.asp`:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<H3>Site-ul magazinului &quot;aMARE&quot;</H3>

<FORM NAME="frmZi" ACTION="asptr1.asp" METHOD="POST">
<H4>Selectati o zi din lista</H2>
<TABLE ALIGN="LEFT" WIDTH="40%">
   <TR>
      <TD>
         <SELECT NAME="zi">
            <OPTION VALUE="Luni" SELECTED>Luni</OPTION>
            <OPTION VALUE="Marti">Marti</OPTION>
            <OPTION VALUE="Miercuri">Miercuri</OPTION>
            <OPTION VALUE="Joi">Joi</OPTION>
            <OPTION VALUE="Vineri">Vineri</OPTION>
            <OPTION VALUE="Simbata">Simbata</OPTION>
            <OPTION VALUE="Duminica">Duminica</OPTION>
         </SELECT>
      </TD>
      <TD>
       <INPUT TYPE="SUBMIT" VALUE="Trimite">
      </TD>
   </TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```


Fişierul `asptr1.asp`:

```
<%@ Language=VBScript %>
<% Option Explicit %>
<%
Dim ziuaaleasa
If Not isEmpty(Request.Form("zi")) then
   ziuaaleasa= Request.Form("zi")
   Select case ziuaaleasa
      case "Duminica"
            Server.Transfer("PrgDuminica.asp")
      Case "Simbata"
            Server.Transfer("PrgSimbata.asp")
      Case Else
            Server.Transfer("PrgNormal.asp")
   End Select
Else
   Server.Transfer "asptr.asp"
End If
%>
```

Fişierul `PrgNormal.asp`:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<H1>In zilele de luni - vineri</H1>
```

```
<A HREF="asptr.asp"><FONT COLOR="RED" size="2">(inapoi)</FONT></A>
<P>Magazinul are programul:
<PRE>
  8-13: deschis
 13-14: pauza de masa
 14-21: deschis
</PRE>
</P>
</BODY>
</HTML>
```

Fişierul `PrgSimbata.asp`:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<H1>Program de simbata</H1>
<A HREF="asptr.asp"><FONT COLOR="RED" size="2">(inapoi)</FONT></A>
<P>Magazinul este deschis intre orele 10-15.</P>
</BODY>
</HTML>
```

Fişierul `PrgDuminica.asp`:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<H1>Program de duminica</H1>
<A HREF="asptr.asp"><FONT COLOR="RED" size="2">(inapoi)</FONT></A>
<P>Duminica magazinul este inchis!</P>
</BODY>
</HTML>
```


Metoda Server.MapPath

Este bine de evitat scrierea căilor absolute în aplicație deoarece acestea creează dependența de cale. Dacă dorim să rulăm aplicația în directorul curent, o astfel de cale ar putea fi `C:\CursASP\ap1.asp`. Deseori, administratorii de Web server-e preferă ca numai anumite discuri să stocheze aplicații. Într-o astfel de situație discul `C:` ar putea fi interzis aplicațiilor motiv pentru care orice cale care începe cu `C:` va fi incorectă şi aplicația nu va putea rula corect după ce a fost pusă pe server decât dacă se modifică porțiunile de cod sursă care conțin referiri la căile absolute. Cea mai simplă modalitate de evitare a situației este punerea căilor absolute într-un singur fişier, care va fi inclus în porțiunile de cod care trebuie să utilizeze căile respective.

Accesul la un fişier se poate realiza doar dacă se cunoaşte calea fizică spre el. Metoda `Server.Path` realizează transformarea argumentului de intrare, ce poate fi o cale virtuală sau relativă, într-una fizică. Când calea începe cu "`/`" sau cu "`\`", metoda va întoarce o cale virtuală completă. Altfel, metoda întoarce o cale relativă la directorul în care se află fişierul ASP în curs de prelucrare. De exemplu, pentru aflarea directorului fizic corespunzător directorului virtual `unu` se poate folosi codul care urmează. De asemenea, el permite aflarea căii fizice în care este stocat fişierul în curs de execuție prin folosirea variabilei `PATH_INFO`.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<% Dim caleFizica
  caleFizica=Server.MapPath("/unu")
  Response.Write "Calea fizica corespunztoare " & _
```

```
   "directorului virtual <B>unu</B> este: " & caleFizica & "<BR>"
 Response.Write Server.MapPath(Request.ServerVariables("PATH_INFO")) & "<BR>"
  Response.Write Server.MapPath("aspmap.asp") & "<BR>
  Response.Write Server.MapPath("TestMAP/aspmap.asp") & "<BR>"
  Response.Write Server.MapPath("/") & "<BR>"
  Response.Write Server.MapPath("../") & "<BR>"

%>
</BODY>
</HTML>
```

Codul de mai sus lucrează corect doar dacă directorul virtual `unu` există. În lipsa lui nu se va genera o eroare, ci doar un rezultat incorect (o cale fizică care nu există). Soluția nu rezolvă problema, în esență, deoarece dependența de calea fizică este înlocuită cu cea de existență a directorului virtual.

În cazul argumentelor `"aspmap.asp"` ,`"TestMAP/aspmap.asp"`, deoarece nu se foloseşte caracterul / la început, căile generate sunt relative la directorul curent. În cazul argumentului `"/"` este întoarsă calea fizică spre *directorul home* al rădăcinii site-ului de Web, iar în cazul lui `"../"` se întoarce calea fizică, relativă, către pagina care este vizualizată de navigator. Notația `".."` pentru accesarea părinților dintr-o cale poate fi folosită, din motive de securitate, numai dacă proprietatea `ASPEnableParentPath` are valoarea `True`. Prin folosirea ei, un script ar putea accesa directori care sunt deasupra celui virtual până la cel rădăcină.

Metoda Server.GetLastError

Aplicațiile script ASP pot avea erori:

- la preprocesare;
- la compilare;
- în execuție.

Erorile de preprocesare apar atunci când preprocesorul ASP, ca urmare a instrucțiunii include, nu poate găsi fişierul care este referit. Erorile de compilare apar atunci când script-ul conține cuvinte cheie scrise greşit, care nu vor fi recunoscute de compilator, eventual cuvinte cheie care lipsesc. Aceste erori sunt prinse, de obicei, în fazele de dezvoltare şi de testare.

Erorile în execuție sunt mai greu de găsit ca urmare a multitudinii factorilor care pot conduce la ele (erori de logică, resurse blocate, eventual cu versiuni incompatibile sau care lipsesc). Când motorul ASP generează eroarea `500;100`, IIS creează un obiect `ASPErrorObject` care va conține informațiile legate de eroare. Apoi, va folosi metoda `Server.Transfer` pentru a deschide pagina `/iishelp/common/500-100.asp` pentru afişarea erorii. Această pagină este suficientă pentru nevoile simple ale dezvoltării aplicației. Dacă în practică, totuşi, ea nu este suficientă poate fi înlocuită cu o pagină realizată de noi. În cazul lui IIS, se deschide *MMC*, se va selecta *Default Web Site*, vă poziționați pe directorul virtual al aplicație, se face clic pe butonul din dreapta, selectați *Properties*, de aici *Custom Errors*. Aici, selectați intrarea `500;100`, clic pe *Edit Properties*, iar în fereastra apărută înlocuiți URL-ul implicit cu cel ce are pagina nouă pentru tratarea erorilor. Clic pe **OK**, închideți fereastra de dialog *Error Mapping Properties*, clic pe **Apply** din fereastra de dialog *Custom Errors*, apoi clic pe **OK.** În această pagină vor putea fi afişate următoarele proprietăți ale obiectului `ASPErrorObject`:

Proprietate	Descriere
`ASPCode`	Codul de eroare intern al IIS.

Proprietate	Descriere
`Number`	Numărul erorii.
`Source`	Linia codului de script în care apare eroarea.
`File`	Numele fişierului cu eroare.
`Line`	Numărul liniei de cod în care apare eroarea.
`Description`	Text succint care descrie eroarea.

NOTĂ

Deşi conform documentaţiei IIS, toate erorile în execuţie au codul `500;100`, se pare că nu este exact aşa. Unele erori în execuţie generează mesaje implicite fără folosirea paginii de erori. Un astfel de exemplu este împărţirea cu zero.

Obiectul Scripting Dictionary

În afară de obiectele ASP intrinseci, ASP vine cu o mulțime de obiecte care pot fi utile în anumite situații. Unul dintre acestea este obiectul `Dictionary`, el fiind parte din **Microsoft Scripting Runtime**, care este un model de obiecte conținut de o bibliotecă ce se instalează cu **Microsoft Office**. Un *model de obiecte* definește structura unui program organizat pe obiecte. Prin definirea relațiilor între obiectele programului, modelul de obiecte pune obiectele într-o ordine cât mai natural accesibilă programatorului.

Obiectul `Dictionary` permite stocarea unor perechi `cheie=articol` (denumirea tehnică este aceea de memorie asociativă). Articolele pot fi orice date, iar fiecare articol trebuie asociat unei chei unice. Cheia se foloseşte pentru extragerea unui articol individual şi este, de obicei, un întreg sau un şir (poate fi orice tip cu excepția unui tablou). Fiind un obiect COM el are următoarele caracteristici:

- are proprietăți şi metode publice expuse printr-o interfață care pot fi utilizate în coduri care ştiu să trateze modelul COM;
- fiind un obiect COM, interfața trebuie să rămână neschimbată.

Proprietăţile obiectului Dictionary

Proprietate	Descriere
`Count`	Întoarce numărul de asocieri din obiectul `Dictionary`.
`Item`	Întoarce sau setează valoarea asociată cu şirul cheie sau indexul întreg.
`Key`	Modifică şirul cheie.

Metodele obiectului Dictionary

Metodă	Descriere
`Add(cheie,valoare)`	Adaugă un nou şir `cheie` lui `Dictionary` împreună cu `valoarea` specificată. Dacă `cheia` există, atunci apare o eroare.
`CompareMode(ModComparatie)`	Controlează modul în care obiectul `Dictionary` compară cheile. Dacă `ModComparatie` ia valoarea `vbBinaryCompare` (valoarea 0) nu se ține cont de scrierea cu litere mari şi mici, dacă valoarea este `vbTextCompare` (valoarea 1) se ține cont de scrierea de literele mari şi mici; metoda nu este documentată, dar poate fi folosită fără a primi eroare.
`Exists(cheie)`	Întoarce valoarea `True` în cazul în care cheia există sau `False` dacă nu există.
`Items`	Întoarce un tablou `Variant` cu toate valorile curente stocate în `Dictionary`.
`Keys`	Întoarce un tablou `Variant` cu toate cheile curente stocate în `Dictionary`.
`Remove(cheie)`	Şterge pe `cheie`, dacă aceasta există.

Metodă	Descriere
RemoveAll	Şterge toate cheile.

În continuare se prezintă o aplicație cu acest obiect, rezultatele afişate de aceasta fiind prezentate în **Figura 52**.

```
<%@ Language=VBScript %>
<% Option Explicit %>
<%
      Function spatii(n)
            Dim i
            Dim s
            s = ""
            For i=0 to n
                  s=s+"    "
            Next
            spatii = s
      End Function

      Function dictionarNou()
         Dim D
         Set D = Server.CreateObject("Scripting.Dictionary")
         D.CompareMode=vbTextCompare
         Set dictionarNou=D
      End Function

      Function tiparireTablou(tabl,poz)
         Dim V
         For Each V in tabl
            If isArray(V) Then
               Call tiparireTablou(V, poz+1)
            Elseif isObject(V) Then
               If Typename(V) = "Dictionary" then
                 Call tiparireDictionar(V, poz+1)
               Else
                 Response.Write "{Obiect: " & Typename(V) & "}<BR>"
               End If
            Else
               Response.Write spatii(poz) & V & "<BR>"
            End If
         Next
      End Function

      Function tiparireDictionar(D,poz)
         For Each V in D.Keys()
            If isObject(D(V)) Then
               If Typename(D(V)) = "Dictionary" Then
                Response.Write spatii(poz) & "Dictionar <B>" & V & "</B><BR>"
                Call tiparireDictionar(D(V), poz+1)
               End If
            Elseif isArray(D(V)) Then
                Response.Write spatii(poz) & "Tablou <B>" & V & "</B><BR>"
               Call tiparireTablou(D(V), poz+1)
            Else
               Response.Write spatii(poz) & V & "=" & D(V) & "<BR>"
            End If
         Next
      End Function

      Dim dictRad 'Variabila dictionar principala
      Dim D     'Variabila dictionar secundara
      Dim i
      Dim j
      Dim tabl  'Variabila tablou cu elemente sir de caractere
```

```
      Dim V     'Variabila Variant

      Set dictRad = dictionarNou()
      For i = 1 to 5
         Set D = dictionarNou()
         For j = 1 to 3
            D.Add "Cheie " & cstr(j), j
            if (i=3) and (j=2) Then
            D.Add "Cheie - Tablou", Array ("Un", "tablou", "de", "siruri")
            End If
         Next
         dictRad.Add "Cheie Radacina " & cstr(i), D
      Next
      tabl = Array("Inca", "un", "tablou", "de", "siruri")
      dictRad.Add "Cheie Racina - Tablou", tabl

      Call tiparireDictionar(dictRad,0)
%>
```

Funcția `tiparireDictionar(D,poz)` este recursivă. Apelul recursiv este determinat de găsirea unui articol care, la rândul lui, este un obiect `Dictionary`. Este posibil ca un articol al dicționarului să fie tablou. Acesta va fi afişat folosind funcția `tiparireTablou(tabl,poz)`. Parametrul `poz` al celor două funcții permite afişarea articolelor de dicționar aliniate în funcție de nivelul de imbricare.

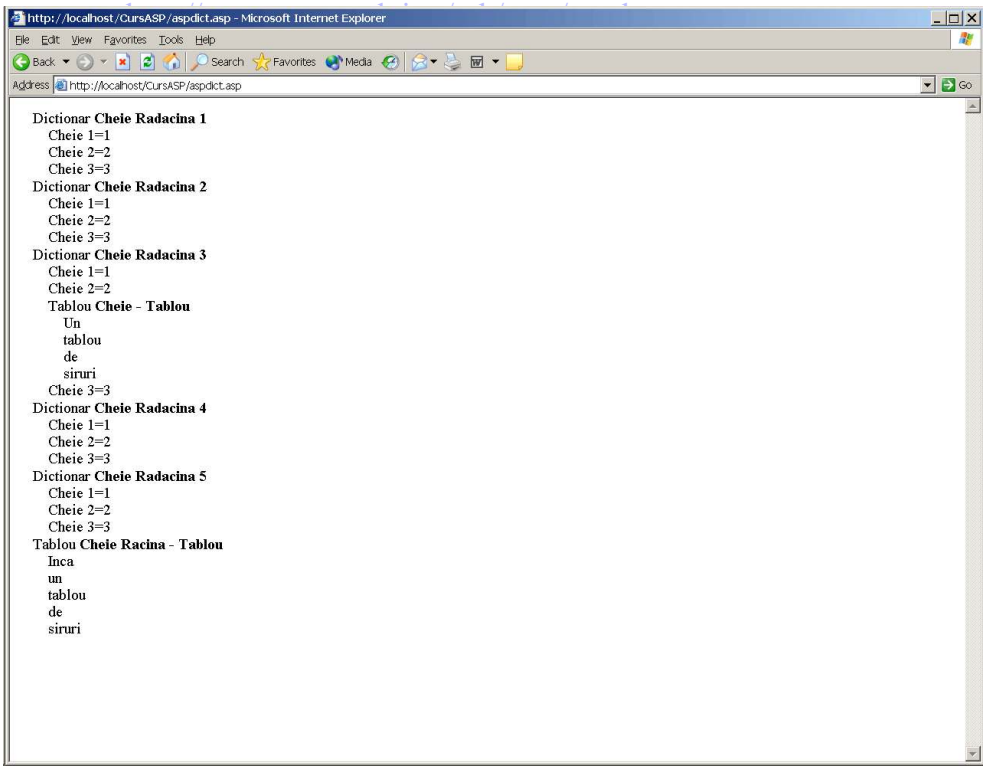


Figure 52 Lucrul cu obiectul Scripting.Dictionary

Obiectul FSO

Modelul de obiecte `FileSystemObject` (`FSO`) permite folosirea unor obiecte cu proprietăți, metode şi evenimente pentru accesul la directoare şi pentru realizarea de prelucrări ale fişierelor pe server. FSO este conținut în biblioteca cu numele `scrrun.dll` şi trebuie să fie în directorul `system32` al calculatorului pe care se rulează serevrul de Web. El are următoarele obiecte şi colecții:

Obiect	Descriere
`FileSystemObject`	Obiectul de bază al modelului FSO. Deseori el trebuie creat pentru a putea accesa alte obiecte ale modelului FSO. Are metode care le dublează pe cele ale altor obiecte din FSO în scopul scăderii nevoii de creare a mai multor obiecte pentru îndeplinirea obiectivelor.
`Drive`	Permite accesul la informațiile legate de discuri (fizice: HDD, CD-ROM etc. şi logice: discuri de rețea).
`File`	Permite crearea, mutarea, ştergerea de fişiere şi extragerea informațiilor legate de nume, cale etc.
`Folder`	Permite crearea, mutarea, ştergerea de directori şi extragerea informațiilor legate de nume, cale etc.
`TextStream`	Permite citirea şi scrierea de fişiere text.

Colecție	Descriere
`Drives`	Lista discurilor fizice şi logice legate la sistem. Dacă unitatea este CD-ROM ea apare în listă chiar dacă nu are CD inserat în ea.
`Files`	Lista tuturor fişierelor unui director.
`Folders`	Lista tuturor subdirectorilor unui director.

În ASP, accesul la fişiere se realizează automat de către server. Atunci când primeşte o cerere de la un client, în funcție de acțiunea de realizat, server-ul deschide fişierul, îi citeşte conținutul, îl rulează, apoi îl trimite clientului. Aplicațiile Web pot citi şi scrie date direct în fişiere cu condiția să ştie calea fizică spre acestea (nu se poate lucra cu directori şi fişiere virtuale). În acest scop colecția `Request.ServerVariables` se poate folosi pentru determinarea căii fizice către script-ul rulat din aplicația curentă, iar `Server.MapPath` se va folosi pentru transformarea căii virtuale în una fizică. Deci, limitarea datorată obligativității folosirii căilor fizice poate fi uşor evitată, iar accesul la fişierele fizice se poate face simplu şi elegant. Este bine ca referințele la surse externe de date, după cum am mai spus deja, să fie stocate într-un singur fişier pentru a creşte portabilitatea codului.

NOTĂ

`FSO` **are un comportament ciudat dacă sunt instalate, pe Web server, soft-uri de tipul antivirus care au opțiunea "Script Blocking" activată. În cazul lui NAV 2001 și 2002, dacă FSO încearcă să scrie sau să citească fișiere, acțiunea pare să fie identificată ca o activitate a unui virus, motiv pentru care server-ul, deci și navigatorul, vor fi blocate.**

În cazul lui NAV, după dezactivarea opțiunii "Script Blocking", sistemul trebuie repornit.

Afişarea fişierelor dintr-un director

Afişarea numelor fişierelor unui director presupune cunoaşterea căii fizice (eventual relative) la directorul în cauză. În exemplul următor calea este `"C:/Inetpub/poze"`. Metoda `GetFolder` întoarce un obiect `Folder` la calea primită prin care vor putea fi accesate toate proprietățile directorului. Obiectul `Folder` întoarce, prin proprietatea `File`, o colecție de obiecte `File` stocate în director, iar prin proprietatea `SubFolders` colecția de obiecte `Folders` din respectivul director. Codul aplicației ce realizează afişarea numelor fişierelor unui director este prezentat în `liastafisiere.asp`, iar rezultatele în **Figura 53**. O aplicație mai practică apare în `poze.asp`. Aici, aplicația de afişare a numelor fişierelor este extinsă pentru a realiza afişarea automată a unor poze. Pentru aceasta se verifică extensia fişierelor, iar în situația în care aceasta este ".JPG" se face afişare (se generează marcaj IMG), rezultatele sunt prezentate în **Figura 54**. Codul de afişare este stocat în directorul `cod`, iar pozele trebuie să fie stocate în directorul părinte a lui `cod`. Aplicația `listafisieredirectori.asp` este şi ea o extensie a celei de afişare a numelor fişierelor, permițând afiarea tururor numelor de fişiere şi de subdirectori din directorul de pornire. Rezultatele se prezintă în **Figura 55**. Pentru a simplifica utilizarea modelului `FSO` voi crea un grup de funcții care vor fi stocate într-un fişier cu numele `bibFSO.inc`. Acestea vor trebui incluse cu `include` în aplicația ce doreşte să le folosească.

Fişierul `bibFSO.inc`:


```
<%
Function creeazaFSO()
    Set creeazaFSO = Server.CreateObject("Scripting.FileSystemObject")
End Function

Function existaFisier(specFisier)
    existaFisier = creeazaFSO.FileExists(specFisier)
End Function

%>
```

Fişierul `liastafisiere.asp`:

```
<%@ LANGUAGE = VBScript %>
<% Option Explicit %>
<!-- #include file = "bibFSO.inc" -->
<%
Function AfisareFisiere(cale)
   Dim fso, dir, listadir, listafis,f,s


   Set fso = CreateObject("Scripting.FileSystemObject")
   Set dir = fso.GetFolder(cale)
   Set listafis = dir.Files

   For Each f in listafis
      s = s & f.name
      s = s & "<BR>"
   Next

   AfisareFisiere= s
End Function
%>

<HTML>
    <HEAD>
```

```
        <TITLE>Afisarea fisierelor cu FSO</TITLE>
    </HEAD>
    <BODY>
      <%
            Dim f
            f="C:/Inetpub/poze"
            response.write AfisareFisiere(f)
      %>
    </BODY>
</HTML>
```

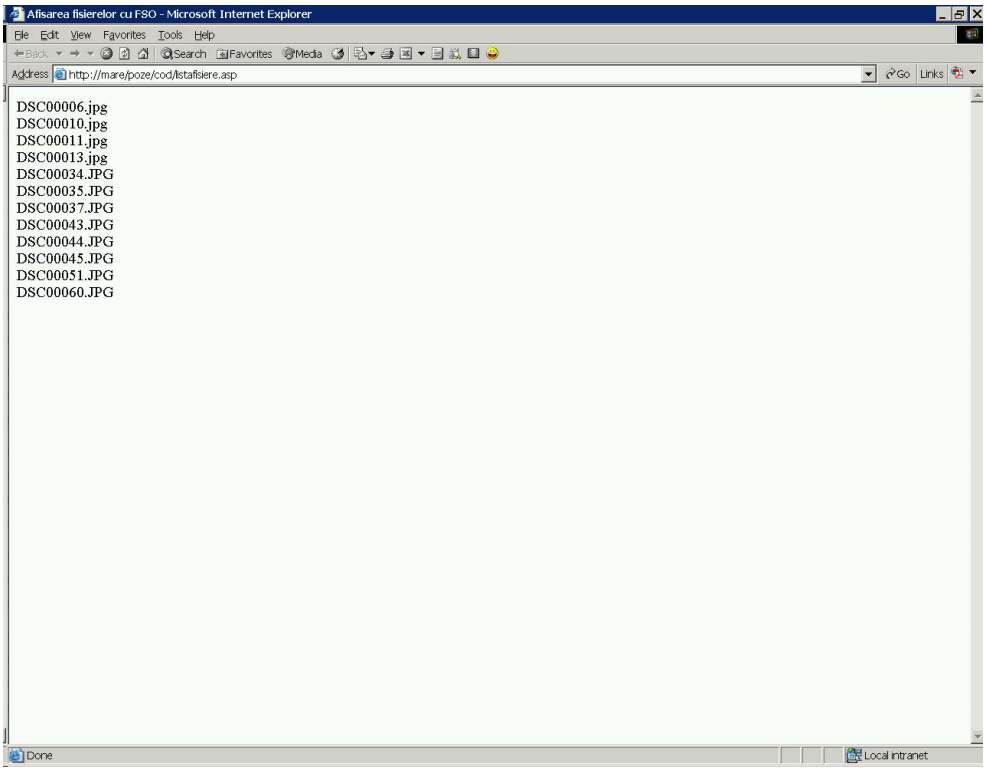


Figure 53 Afişarea fişierelor cu FSO

Fişierul `poze.asp`:

```
<%@ LANGUAGE = VBScript %>
<% Option Explicit %>
<!-- #include file ="bibFSO.inc" -->
<%
Function AfisareFisiere(cale)
   Dim dir, listadir, listafis, f, s, c, n

   Set dir = creeazaFSO.GetFolder(cale)
   Set listafis = dir.Files
   c = 1

   For Each f in listafis
      if (UCase(Right(f,3)) = "JPG") Then
            n = "'../" & f.name & "'"
            If ((c mod 4) <> 0) Then
             s = s & "<IMG SRC =" & n & "ALT =" & n & " WIDTH=300 ALIGN=LEFT>"
            Else
            s = s & "<IMG SRC =" & n & "ALT =" & n & " WIDTH=300>"
            End If
      end if
        c=c+1
      s = s & "<P>"
   Next
```

```
   AfisareFisiere = s
End Function
%>

<HTML>
    <HEAD>
        <TITLE>Afisare imagini</TITLE>
    </HEAD>
    <BODY>
      <%
            Dim f
            f="C:/Inetpub/poze"
            response.write AfisareFisiere(f)
      %>
    </BODY>
</HTML>
```

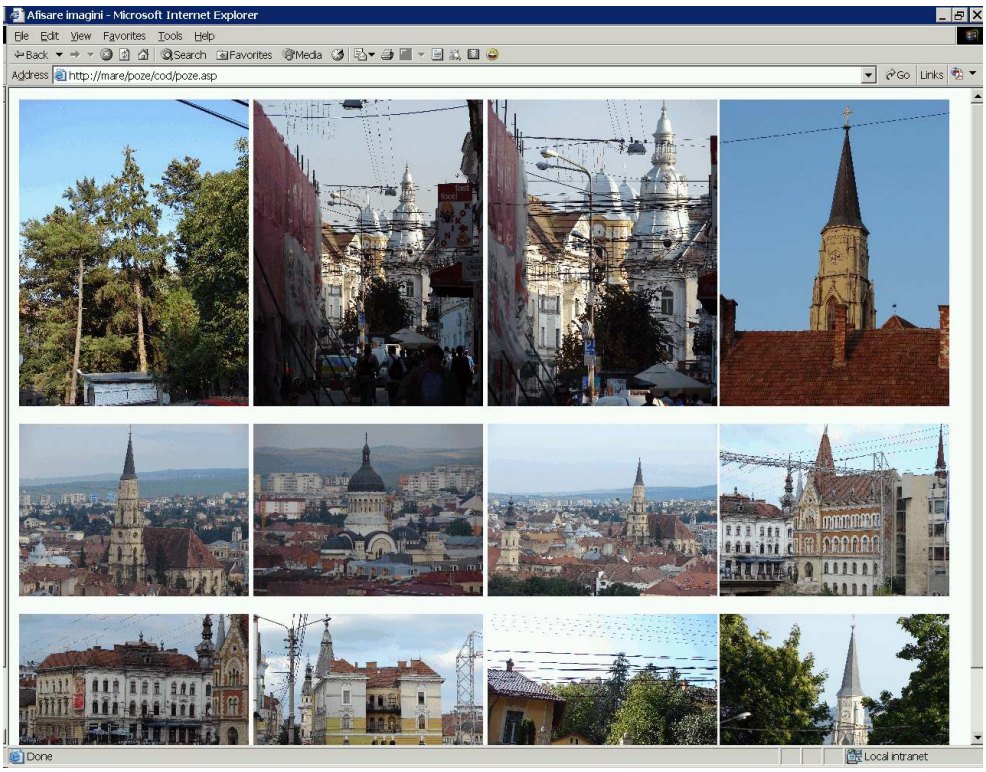


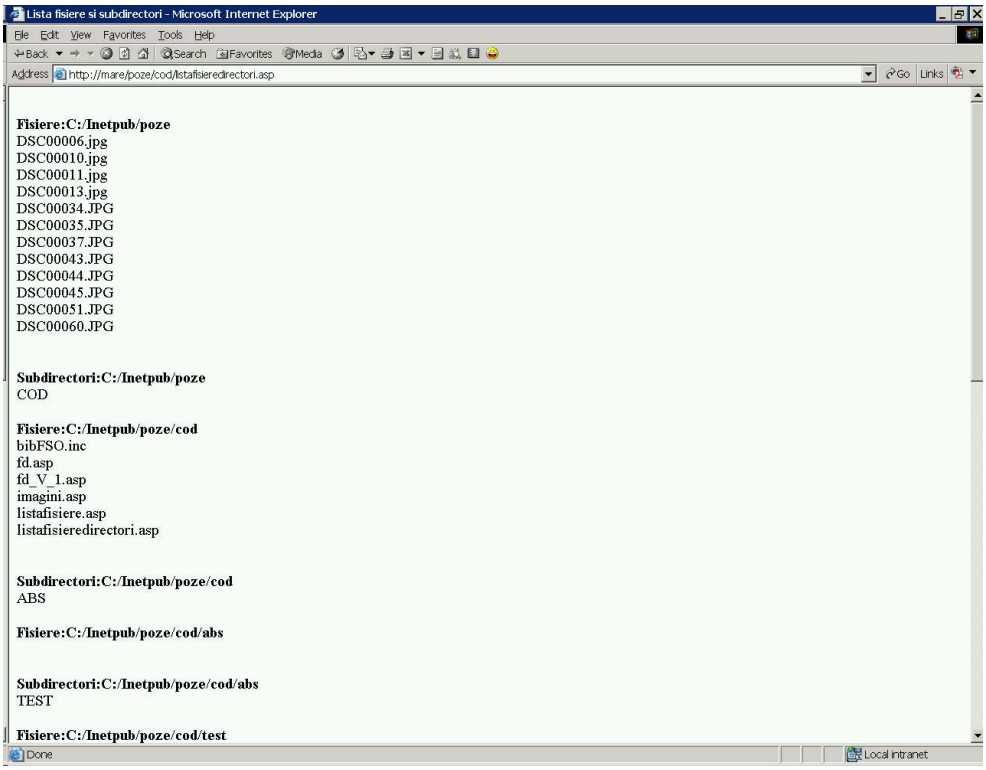
Figure 54 Vizualizarea pozelor unui director cu FSO

Fişierul `listafisieredirectori.asp`:

```
<%@ LANGUAGE = VBScript %>
<% Option Explicit %>
<!-- #include file = "bibFSO.inc" -->
<%
Function FisiereDirectori(cale)
   Dim dir, subdir, f, fis, d
   Dim s
   Set dir = creeazaFSO.GetFolder(cale)
   Set subdir = dir.SubFolders
   Set fis = dir.Files

   s= "<BR><B>Fisiere:" + cale + "</B><BR>"
   For Each f in fis
      s = s & f.name
      s = s & "<BR>"
   Next

   s = s+ "<BR><BR><B>Subdirectori:" + cale +"</B><BR>"
   For Each d in subdir
```

```
      s = s & UCase(d.name)
      s = s & "<BR>"
      s= s+FisiereDirectori(cale & "/" & d.name)
   Next

   FisiereDirectori = s
 End Function
%>

<HTML>
    <HEAD>
        <TITLE>Lista fisiere si subdirectori</TITLE>
    </HEAD>
    <BODY>
      <%
            Response.Write FisiereDirectori("C:/Inetpub/poze")
      %>
    </BODY>
</HTML>
```

Figure 55
Lista fişierelor şi a subdirectorilor cu FSO



Verificarea existenţei unui fişier

Fişierele pot fi folosite la stocarea unor date legate de starea aplicației de către utilizatorii care au vizitat pagina sau de cei care au dreptul să o viziteze. Acestea ar fi motivele pentru care este esențial să nu scriem peste conținutul unui fişier ce stochează date esențiale pentru funcționarea aplicației. Pentru verificarea existenței unui fişier se va folosi metoda `FileExist` a lui `Scripting.FileSystemObject`. Pentru a verifica existența fişierelor `aspexistaFSO.asp`, `unfis.txt` se va folosi aplicația următoare care va fi stocată în fişierul `aspexistaFSO.asp`.

```
<%@ Language=VBScript %>
<% Option Explicit %>
<!-- #include file="bibFSO.inc" -->

<%
      Dim numeFis
```

```
        numeFis ="C:\CursASP\aspexistaFSO.asp" 'numele fisierului
        numeFis = numeFis & "-><B>" & existaFisier(numeFis) & "</B><BR>"
        Response.write numeFis

        numeFis ="C:\CursASP\unfis.txt" 'numele fisierului
        numeFis = numeFis & "-><B>" & existaFisier(numeFis) & "</B><BR>"
        Response.write numeFis

%>
```

Fişierul `bibFSO.inc` trebuie stocat în acelaşi director cu fişierul `aspexistaFSO.asp`. Deoarece primul fişier există, funcția `existaFisier` întoarce valoarea `True`. Deoarece cel de al doilea nu există, funcția `existaFisier` va întoarce valoarea `False`.

Deschiderea unui fişier

Pentru deschiderea unui fişier se va folosi metoda `OpenTextFile` care facilitează accesul secvențial la fişiere text după cum urmează:

```
Set TxtStr = FileSystemObject.OpenTextFile(numeFis, modIE, Creare, Format)
```

Ultimele trei argumente sunt opționale şi au semnificațiile prezentate în tabelele care urmează.

Argumentul `modIE`

Valoare	Semnificație
`1`	Deschide fişierul numai pentru citire. Se generează eroare dacă se încearcă scrierea în acest mod.
`2`	Deschide fişierul pentru scriere. Dacă fişierul există deja, acest mod duce la distrugerea conținutului anterior.
`8`	Deschide fişierul în modul de scriere cu adăugare la sfârşitul fişierului.

Argumentul `Creare`

Valoare	Semnificație
`True`	Se va crea un nou fişier dacă acesta nu există.
`False`	Nu se va crea un fişier nou (valoare implicită).

Argumentul nu opreşte suprascrierea unui fişier dacă acesta există deja. Dacă se încearcă deschiderea unui fişier care nu există cu valoarea `False` apare eroare.

Argumentul `Format`

Valoare	Semnificație
`-1`	Manipulează fişierul după codificarea Unicode.
`0`	Manipulează fişierul după codificarea ASCII.
`-2`	Manipulează fişierul în funcție de setările implicite ale sistemului de operare.

În aplicația care urmează metoda `OpenTextFile` a obiectului `FileSystemObject` se foloseşte pentru a crea un obiect de tipul flux de text (`TextStream`) şi pentru a deschide un fişier cu numele

`Text.htm` în rădăcina discului `D:`. Scrierea în fişier se face cu metodă `WriteLine`, iar fluxul se închide cu metoda `Close`. Apoi, un nou flux va fi creat şi conţinutul fişierului va fi citit cu metoda `ReadAll` după care va fi afişat în fereastra navigatorului.

```
<!-- #include file="bibFSO.inc" -->
<HTML>
<HEAD>
<BODY>
<%
      Dim numeFis
      Dim FS
      Dim TS

      numeFis ="D:\Text.htm" 'numele fisierului
      Set FS = creeazaFSO()
      ' Adaugare, True, Unicode
      Set TS = FS.OpenTextFile(numeFis, 8 ,True, -1)
      TS.WriteLine "<H3>Un fisier</H3>"
      TS.WriteLine "<P>format din 2 linii"
      TS.Close

      If existaFisier(numeFis) Then
            Response.write "Fisierul " & numeFis & " are continutul: <BR>"
            ' Citire, False, Unicode
            Set TS = FS.OpenTextFile(numeFis, 1 ,False, -1)
            Response.write TS.ReadAll
            TS.Close
            Set TS= Nothing
      Else
            Response.Write "Fisierul " & numeFis & " nu exista!<BR>"
      End If
      Set FS= Nothing
%>
</BODY>
</HTML>
```


Obiectul TextStream

Obiectul `TextStream` are metode şi proprietăţi ce permit accesul secvenţial la datele unui fişier. Termenul de flux se foloseşte pentru a descrie transferul de date între un generator de date şi un consumator. În cazul fluxului text, în momentul deschiderii lui, i se asociază şi fişierul care va fi manipulat (citit sau scris) cu ajutorul metodelor specifice acestui obiect. După terminarea activităţilor cu fluxul, acesta trebuie închis. Metodele şi proprietăţile obiectului `TextStream` se prezintă în tabelele următoare.

Metodă	Descriere
`Close`	Închide fişierul asociat obiectului `TextStream`.
`Read`	Citeşte un număr specificat de caractere din fişier şi întoarce textul citit.
`ReadAll`	Citeşte şi întoarce întregul conţinut al fişierului.
`ReadLine`	Citeşte şi întoarce următoarea linie din fişier; linia se termină la întâlnirea caracterului `newline` (`vbLF`). Caracterul `newline` nu este întors.
`Skip`	Sare peste un număr specificat de caractere ale fişierului. Nu se poate "sări" înapoi, ci doar înainte, adică numărul de caractere de sărit trebuie să fie un întreg pozitiv.

Metodă	Descriere
`SkipLine`	Sare la linia următoare. Se poate folosi numai la citirea fişierului.
`Write`	Scrie un şir în fişier din poziția curentă a pointer-ului de fişier.
`WriteBlankLines`	Scrie o linie vidă în fişier. Această linie va conține doar caracterul `newline` (`vbLF`).
`WriteLine`	Scrie şir în fişier şi adaugă la sfârşitul lui caracterul `newline`.

Proprietate	Descriere
`AtEndOfLine`	`True` dacă pointer-ul de fişier este la sfârşitul liniei (următorul caracter este `newline`).
`AtEndOfStream`	`True` dacă pointer-ul de fişier este la sfârşitul fişierului.
`Column`	Permite citirea numărului de coloană curent al pointer-ului de fişier. Valoarea pentru primul caracter este 1.
`Line`	Permite citirea numărului de linie din fişierul curent. La deschiderea acestuia numărul de linie este 1.

Pentru a lucra mai simplu cu fişierele se prezintă în continuare o variantă extinsă a "fişierului bibliotecă" `bibFSO.inc` . Aici se implementează câte o procedură pentru scrierea, adăugarea şi citirea unui fişier. Apoi, se exemplifică modul de folosire a noilor funcții în fişierul `asptextSCAFSO.asp`.

Fişierul `bibFSO.inc`:

```
<%
Function creeazaFSO()
    Set creeazaFSO = Server.CreateObject("Scripting.FileSystemObject")
End Function

Function existaFisier(specFisier)
    existaFisier = creeazaFSO.FileExists(specFisier)
End Function

Function dirSpecial(director)
    dirSpecial = creeazaFSO.getSpecialFolder(director)
End Function

Function dirWindows()
    dirWindows = dirSpecial(0)
End Function

Function dirSystem()
    dirSystem = dirSpecial(1)
End Function

Function dirTemp()
    dirTemp = dirSpecial(2)
End Function

Sub scrieFisText(numeFis, Tip, Linie)
    Dim FS
    Dim TS
    Set FS = creeazaFSO()
```

```
    Set TS = FS.OpenTextFile(numeFis, 2, Tip, -1)
    TS.WriteLine Linie
    Ts.Close
    Set TS = Nothing
    Set TS = Nothing
End Sub

Sub adaugaFisText(numeFis, Tip, Linie)
    Dim FS
    Dim TS
    Set FS = creeazaFSO()
    Set TS = FS.OpenTextFile(numeFis, 8, Tip, -1)
    TS.writeLine Linie
    TS.Close
    Set TS = Nothing
    Set FS = Nothing
End Sub

Function citesteFisText(numeFis)
    Dim s
    Dim FS
    Dim TS
    Dim F
    Set FS = creeazaFSO()
    If Not existaFisier(numeFis) Then
      Err.Clear
      Err.Raise 50000, "bibFSO.inc -> citesteFisText()","Fisierul"&
 & numeFis & " nu exista!."
      Exit Function
    End if
    Set F = FS.GetFile(numeFis)
    If F.Size > 0 Then
      Set TS = FS.openTextFile(numeFis, 1, False, -1)
      citesteFisText = TS.ReadAll
      TS.Close
    End If
    Set TS = Nothing
    Set FS = Nothing
End Function

Sub stergeFisText(numeFis)
   Set FS = creeazaFSO()
   FS.DeleteFile(numeFis)
End Sub
%>
```


Fişierul `asptextSCAFSO.asp`:

```
<!-- #include file="bibFSO.inc" -->
<HTML>
<HEAD>
<BODY>
<%
      Dim numeFis
      Dim textFis

      numeFis ="D:\Text1.htm" 'numele fisierului
      Call scrieFisText(numeFis, True, "<H1>Prima linie</H1>")
      Call adaugaFisText(numeFis, True, "<H2>A doua linie</H2>")
      Call adaugaFisText(numeFis, True, "<H3>A treia linie</H3>")

      textFis = citesteFisText(numeFis)
      Response.Write textFis

      Call stergeFisText(numeFis)

%>
```

```
</BODY>
</HTML>
```

Scrierea într-un fişier

Scrierea într-un fişier secvenţial se poate face doar la sfârşitul lui. Dacă totuşi dorim să inserăm date în el va trebui să rescriem întreg fişierul. În cazul unor fişiere mari acţiunea va consuma mult timp. Atunci când se lucrează cu fişiere secvenţiale, cea mai rapidă variantă este cea de adăugare la sfârşitul lui a noilor date. Procedura `scrieFisText` creează şi scrie într-un fişier o linie, iar procedura `adaugaFisText` adaugă o linie la un fişier text deja existent.

Citirea dintr-un fişier

Pentru citirea unui fişier acesta trebuie să existe. Funcţia `citesteFisText` realizează citirea întregului conţinut al unui fişier. Dacă fişierul este foarte mare metoda de citire duce la un cosum mare de resurse. Pentru evitarea acestora se poate realiza citirea linie cu linie.

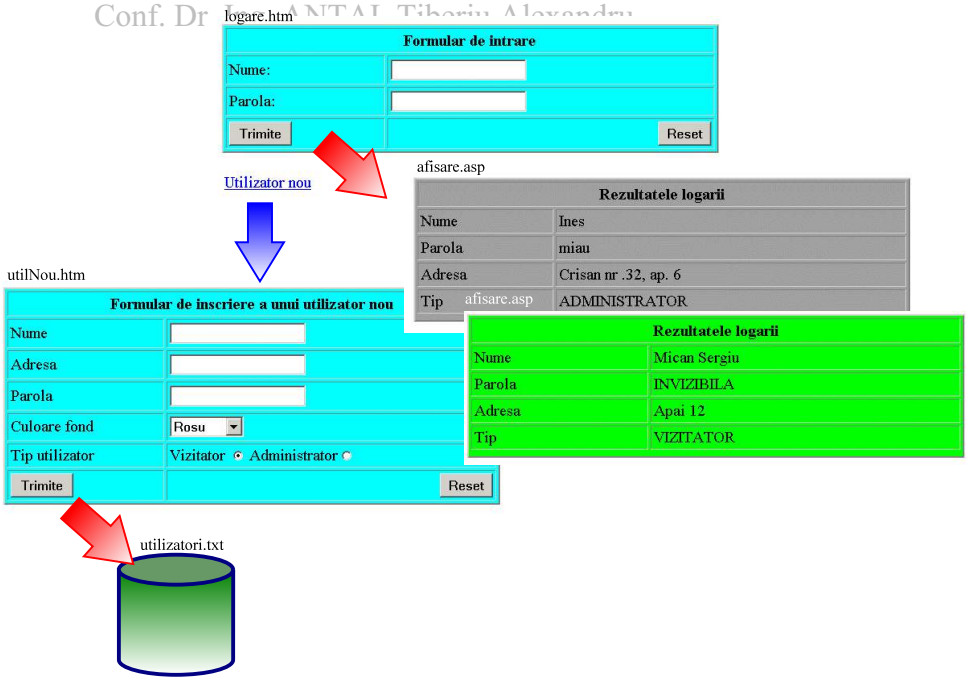
Ştergerea unui fişier

Un fişier poate fi şters numai dacă acesta există. Încercarea de ştergere a unui fişier inexistent va genera eroare. Procedura `stergeFisText` implementează cea mai simplă variantă a unei secvenţe de cod pentru ştergerea unor fişiere.

Realizarea unei aplicaţii cu securitate la legare

Aplicaţia care urmează îşi propune să asigure legarea unui utilizator la o pagină personalizată. Datele se scriu pe server-ul de Web şi se citesc de acolo cu ajutorul modelului `FSO`. Formularele şi rezultatele aplicaţiei se prezintă în **Figura 56**.

Figure 56 Formularele aplicaţiei de legare



Fişierele aplicaţiei trebuie stocate în directorul cu numele *logare*, iar fişierele care o alcătuiesc sunt următoarele:

- ☐ `logare.htm`: formularul de legare necesită un nume de utilizator şi o parolă. Dacă aceastea sunt introduse corect, după apăsarea butonului **Trimite**, se va lansa `verifica.asp`. Dacă sunt câmpuri vide se generează mesaje de eroare, local, cu

ajutorul unor ferestre de alertă. Un mesaj de eroare apare şi dacă numele utilizatorului sau parola lui sunt greşite, dar acest mesaj se generează deja cu ajutorul server-ului din `verifica.asp`.

- ☐ `utilNou.htm`: formular deschis atunci când se face clic pe hiperlegătura Utilizator nou al formularului `logare.htm`. După completarea câmpurilor, la apăsarea butonului **Trimite,** dacă un utilizator cu acelaşi nume şi aceeaşi parolă nu mai există, noul utilizator este înscris în fişierul text `utilizatori.txt` cu utilizatori.
- ☐ `inscriere.asp`: cod lansat în execuție atunci când se apasă butonul **Trimite** al formularului `utilNou.htm`:
- ☐ `verifica.asp`: cod care verifică existenta unui utilizator în fişierul text `utilizatori.txt`
- ☐ `afisare.asp`: cod care afişează datele utilizatorului care s-a legat la site. Tabelul este afişat în culorile specifice utilizatorului, iar parola lui este invizibilă dacă acesta este un utilizator de tipul VIZITATOR.
- ☐ `bibUTIL.inc`: bibliotecă pentru verificarea existenței unui utilizator;
- ☐ `bibFSO.inc`: biblioteca procedurilor de lucru cu modelul FSO

Fişierul `logare.htm`:

```
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM NAME="Legare" ONSUBMIT="return Test()" METHOD="POST" ACTION=
 ="http://localhost/CursASP/logare/verifica.asp">
<TABLE BGCOLOR="AQUA"cellSpacing=3 cellPadding=3 width="50%" align=center>
 border=2>
  <THEAD>
      <TH COLSPAN=2>Formular de intrare</TH>
  </THEAD>
  <TR>
    <TD>Nume:</TD>
    <TD><INPUT TYPE=TEXT NAME=Nume></TD>
  </TR>
  <TR>
    <TD>Parola:</TD>
    <TD><INPUT TYPE=TEXT NAME=Parola></TD>
  </TR>
  <TR>
    <TD><INPUT type=submit value=Trimite></TD>
    <TD ALIGN=RIGHT><INPUT type=reset value=Reset></TD>
  </TR>
  </TABLE>
  </P>
</FORM>
<TABLE width="50%" align=center>
<TR>
      <TD>
            <A HREF="Utilnou.htm">Utilizator nou</A>
      </TD>
</TR>
</TABLE>

<SCRIPT LANGUAGE="VBScript">
      Function Test()
            Dim Frm
            Dim rasp
            rasp=True
            Set Frm = document.Legare
            If Len(Trim(Frm.Nume.Value))= 0 Then
                  alert("Nume nu poate fi vid")
```

```
                  rasp=False
            ElseIf Len(Trim(Frm.Parola.Value))= 0 Then
                  alert("Parola nu poate fi vida")
                  rasp=False
            End If
            Test=rasp
      End Function
</SCRIPT>
</BODY>
</HTML>
```

Fişierul `utilNou.htm`:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM ACTION="http://localhost/CursASP/logare/inscrie.asp" METHOD="POST"✂
 NAME="FrmNou">
<TABLE WIDTH="50%" BGCOLOR="AQUA" ALIGN="CENTER" BORDER=2 CELLSPACING=3✂
 CELLPADDING=3>
   <THEAD>
  <TR>
    <TH COLSPAN=2>Formular de inscriere a unui utilizator nou</TH>
      <TR>
            <TD>Nume</TD>
            <TD><INPUT TYPE="TEXT" NAME="NUME"></TD>
      </TR>
      <TR>
            <TD>Adresa</TD>
            <TD><INPUT TYPE="TEXT" NAME="ADRESA" ></TD>
      </TR>
  <TR>
    <TD>Parola</TD>
    <TD><INPUT TYPE="TEXT" NAME="PAROLA"></TD></TR>
  <TR>
    <TD>Culoare fond</TD>
    <TD>
            <SELECT NAME="CULOARE">
                  <OPTION VALUE="#FF0000">Rosu<BR>
                  <OPTION VALUE="#00FF00">Verde<BR>
                  <OPTION VALUE="#0000FF">Albastru<BR>
                  <OPTION VALUE="#FFFFFF">Alb<BR>
            </SELECT>
      </TD></TR>
  <TR>
    <TD>Tip utilizator</TD>
    <TD>
            Vizitator <INPUT TYPE="RADIO" NAME="TIP" VALUE="VIZITATOR"✂
 CHECKED>
            Administrator<INPUT TYPE="RADIO"  NAME="TIP"✂
 VALUE="ADMINISTRATOR">
      </TD>
    </TR>
      <TR>
            <TD><INPUT TYPE="SUBMIT" VALUE="Trimite"></TD>
            <TD ALIGN="RIGHT"><INPUT TYPE="RESET" VALUE="Reset"></TD>
      </TR>
</TABLE>

</FORM>

</BODY>
</HTML>
```


Fişierul `inscriere.asp`:

```
<%@ Language=VBScript %>
<!-- #include file="../bibFSO.inc" -->
<!-- #include file="bibUTIL.inc" -->
<HTML>
<HEAD>
<BODY>
<%
      Dim numeFis
      Dim textFis
      Dim Nume
      Dim Parola
      Dim Adresa
      Dim Tip
      Dim Culoare
      Dim Linie

      numeFis ="C:\CursASP\logare\utilizatori.txt"
      Nume=Trim(Request.Form("NUME"))
      Parola=Trim(Request.Form("PAROLA"))
      Adresa=Trim(Request.Form("ADRESA"))
      Culoare=Trim(Request.Form("CULOARE"))
      Tip=Trim(Request.Form("TIP"))
      Linie=Nume&";"&Parola&";"&Adresa&";"&Culoare&";"&Tip

      If Not existaFisier(numeFis) Then
            Call scrieFisText(numeFis, True, Linie)
            Response.Write "Fiserul " &numeFis & "a fost creat cu succes."
            Response.Write "si utilizatorul a fost adaugat cu succes."
      Else
            If Not isArray(existaUtil(numeFis,Nume,Parola)) Then
                  Call adaugaFisText(numeFis, True, Linie)
                  Response.Write "Utilizatorul a fost adaugat cu succes."
            Else
                  Response.Write "Eroare, acest utilizator exista deja!"
            End If
      End If
%>

</BODY>
</HTML>
```


Fişierul `verifica.asp`:

```
<%@ Language=VBScript %>
<!-- #include file="../bibFSO.inc" -->
<!-- #include file="bibUTIL.inc" -->

<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim numeFis
      Dim textFis
      Dim Nume
      Dim Parola
      Dim Adresa
      Dim Tip
      Dim Culoare
      Dim tabUtil
      Dim URL

      numeFis ="C:\CursASP\logare\utilizatori.txt"
      Nume=Trim(Request.Form("NUME"))
      Parola=Trim(Request.Form("PAROLA"))
      tabUtil=existaUtil(numeFis,Nume,Parola)
      If isArray(tabUtil) Then
```

```
                  Adresa=tabUtil(2)
                  Tip=tabUtil(4)
                  Culoare=tabUtil(3)
                  URL="http://localhost/CursASP/logare/afisare.asp?"
                  URL = URL & "ADRESA=" & Adresa & "&"
                  URL = URL & "PAROLA=" & Parola & "&"
                  URL = URL & "NUME=" & Nume & "&"
                  URL = URL & "TIP=" & Tip & "&"
                  URL = URL & "CULOARE=" & "'" &Culoare &"'"
                  Response.Redirect(URL)
      Else
                  Response.Write "Eroare, utilizatorul NU exista!"
      End If
%>
</BODY>
</HTML>
```

Fişierul `afisare.asp`:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<%
      Dim Nume,Parola, Adresa, Culoare, Tip
      Nume = Request.QueryString("NUME")
      Parola = Request.QueryString("PAROLA")
      Adresa = Request.QueryString("ADRESA")
      Tip = Request.QueryString("TIP")
      Culoare = Request.QueryString("CULOARE")
%>
<P>
<TABLE CELLSPACING=3 CELLPADDING=3 WIDTH="50%" ALIGN="CENTER"
 BGCOLOR=<%=Culoare%> BORDER=2>
  <THEAD>
    <TH COLSPAN=2>Rezultatele logarii</TH>
  </THEAD>
  <TR>
    <TD>Nume</TD>
    <TD><%=Nume%></TD>
  </TR>
  <TR>
    <TD>Parola</TD>
    <TD>
            <%
              If Tip = "ADMINISTRATOR" Then
                  Response.write Parola
              Else
                Response.write "INVIZIBILA"
              End If
            %>
      </TD>
  </TR>
  <TR>
    <TD>Adresa</TD>
    <TD><%=Adresa%></TD></TR>
  <TR>
    <TD>Tip</TD>
    <TD><%=Tip%></TD></TR>
  <TR>
    <TD></TD>
    <TD></TD></TR></TABLE></P>

</BODY>
</HTML>
```

Fişierul `bibUTIL.inc`:

```
<%
Function existaUtil(numeFis,Nume,Parola)
      Dim textFis
      Dim TabLiniiFis
      Dim oLinie
      Dim TabOlinie
      Dim i
      textFis = citesteFisText(numeFis)
      TabLiniiFis=Split(textFis,vbCrLf,-1,1)
      For i = 0 to UBound(TabLiniiFis)
            oLinie=TabLiniiFis(i)
            If Len(oLinie) > 0 Then
                  TabOlinie=Split(oLinie,";",-1,1)
                  If (StrComp(TabOlinie(0),Nume, 1) = 0) And✂
 (StrComp(TabOlinie(1),Parola, 1) = 0) Then
                        existaUtil = TabOlinie
                        Exit Function
                  End If
            End If
      Next
      existaUtil = ""
End Function
%>
```

Pentru ca aplicația să poată funcționa corect fișierul, `bibFSO.inc` trebuie să fie stocat în directorul fizic `"C:/CursASP"`, iar celelalte fișiere ale aplicație vor fi stocate în directorul fizic `"C:/CursASP/logare"`. `CursASP` trebuie să fie numele directorului virtual corespunzător lui `"C:/CursASP"`.

Instrucţiuni pentru tratarea erorilor

Aplicațiile ASP sunt rulate cu ajutorul motorului ASP. Acesta nu permite ca aplicația, datorită unor erori, să o ia razna. Implicit, în cazul unei erori, motorul ASP opreşte aplicația. La nivelul utilizatorului aceasta înseamnă că programul devine inutil, deoarece nu-şi mai poate urma acțiunile de realizat. **VBScript** are mai multe instrucțiuni care permit continuarea execuției paginii, în cazul unei erori, cu scopul terminării aplicației.

Instrucţiunea On Error Resume Next

Prima variantă de "tratare a unei erori" este aceea de ignorare a ei. Instrucțiunea `On Error Resume Next` nu opreşte aplicația, ci face ca programul **VBScript** să fie continuat din linia imediat următoare celei în care a apărut eroarea. Datorită modului de acțiune instrucțiunea permite ascunderea erorilor de scriere greşită a cuvintelor cheie (sintaxă) şi a celor de logică. Ea trebuie folosită dacă se anticipează apariția unor erori dincolo de posibilitățile de control ale programatorului. De exemplu, se încearcă ştergerea unui fişier, dar fişierul poate că este folosit de o altă persoană sau nu mai există deoarece a fost şters de altcineva. Dincolo de anticiparea unor erori `On Error Resume Next` mai poate fi folosită pentru prinderea rapidă a unor erori. De exemplu, dacă se doreşte adăugarea unor valori într-un fişier, trebuie instanțiată o componentă ActiveX pentru verificarea drepturilor de acces la fişiere. Apoi, se va citi fişierul drepturilor de acces înainte de începerea acțiunii de adăugare. Procesul este complicat şi prea lung. Este mult mai simplu să încercăm scrierea în fişier având activată prinderea erorilor. Dacă scrierea este interzisă se va genera o eroare, altfel aceasta va reuşi.

În lipsa folosirii lui `On Error Resume Next` în cod, la apariția unei erori în timpul execuției aplicației se va afişa un mesaj de eroare după care aplicația va fi oprită.

Instrucţiunea On Error Goto 0

Folosirea lui `On Error Resume Next` activează prinderea erorilor până la întâlnirea instrucțiunii `On Error Goto 0`. Aceasta are rolul dezactivării "ignorării erorilor" datorită lui `On Error Resume Next`. Fie aplicația următoare:

```
<!-- #include file="bibFSO.inc" -->
<HTML>
<HEAD>
<BODY>
<%
      Dim numeFis

      numeFis ="D:\Text12.htm" 'numele fisierului

      On Error Resume Next 'prinderea erorilor s-a activat
      Call stergeFisText(numeFis)
      'On Error GoTo 0
      If Err.Numebr <> 0 Then
            Response.Write "Eroare: " & Err.Number & "<BR>"
            Response.Write "Sursa: " & Err.Source & "<BR>"
            Response.Write "Descriere: " & Err.Description & "<BR>"
            Response.Write "FISIERUL " & numeFis  & " NU POATE FI STERS!<BR>"
      End If
      On Error GoTo 0 'prinderea erorilor s-a dezactivat

%>
</BODY>
</HTML>
```

Mesajul de eroare afişat este:

```
Eroare: 438
Sursa: Microsoft VBScript runtime error
Descriere: Object doesn't support this property or method
FISIERUL D:\Text12.htm NU POATE FI STERS!
```

Privit fugar, şi mai ales dacă nu suntem vorbitori de engleză, totul pare să fie în ordine. Aplicaţia a încercat să şteargă fişierul `D:\Text12.htm`. Deoarece acesta nu există, s-a generat un mesaj de eroare. Însă adevărul stă departe de cele afirmate mai sus. Am scris că `On Error Resume Next` ascunde erorile de sintaxă, iar linia lui `If` conţine o astfel de eroare. Mesajul de eroare afişat ţine de eroarea de sintaxă şi nu de imposibilitatea ştergerii fişierului. Dacă eroarea sintactică este eliminată, adică în locul lui `Err.Numebr` se va scrie `Err.Number`, mesajul care este afişat devine:

```
Eroare: 53
Sursa: Microsoft VBScript runtime error
Descriere: File not found
FISIERUL D:\Text12.htm NU POATE FI STERS!
```

Acesta este mesajul de eroare corect în care suntem anunţaţi că nu se poate şterge fişierul dorit. El este generat de la nivelul funcţiei `Call stergeFisText(numeFis)`. Dacă imediat după aceasta ar fi scrisă instrucţiunea `On Error GoTo 0`, pe lângă dezactivarea prinderii erorilor, se vor reseta şi valorile curente ale erorilor. În acest caz acestea nu se vor mai afişa.

Obiectul Err

VBScript foloseşte obiectul `Err` pentru stocarea erorilor care apar în timpul rulării aplicaţiei. Proprietăţile obiectului pot fi setate de către limbajul **VBScript**, un obiect Automation sau de către programatorul în **VBScript**. Metodele şi proprietăţile lui se prezintă în continuare.

Metodă	Descriere
`Clear`	Şterge toate valorile stocate în proprietăţile lui `Err`.
`Raise`	Permite ca utilizatorul să seteze proprietăţile obiectului `Err` pentru ca să poată genera apariţia unei erori.

Proprietate	Descriere
`Description`	Un şir ce conţine o descriere scurtă a ultimei erori apărute.
`HelpContext`	În **VBScript** proprietatea nu are semnificaţie pe server. Ea poate fi folosită însă pentru stocarea unei valori în vederea utilizării ulterioare.
`HelpFile`	La nivelul clientului, proprietatea trebuie să conţină numele fişierului Windows de help care este asociat erorii ce se va afişa. La nivelul server-ului, proprietatea poate fi folosită pentru stocarea unui nume de fişier HTM sau ASP care va putea fi trimis clientului în vederea afişării erorii.
`Number`	Codul numeric al erorii.
`Source`	Şir care conţine sursa erorii. În ASP acesta este deseori numele fişierului ASP generator al erorii.

În continuare se va prezenta o metodă de completare a mesajelor stocate în obiect `Err` pentru o

mai bună descriere a erorii apărute. Secvență de cod corespunzătoare aplicației `asptextSCAFSO.asp` a fost completată pentru prinderea erorilor şi afişarea lor cât mai detaliată după cum urmează (`asptextSCAFSO1.asp`):

```
<!-- #include file="bibFSO.inc" -->
<HTML>
<HEAD>
<BODY>

<%
      Sub Afisare()
            With Response
                  .Write "<TABLE ALIGN=CENTER BORDER=4 BGCOLOR=#66CCFF>"
                  .Write "<TR BGCOLOR=#77BBEE><TD COLSPAN=2✄
 ALIGN=CENTER><B>Eroare in EXECUTIE</B></TD></TR>"
                  .Write "<TR><TD>Cod</TD><TD>" & Err.Number & "</TD>"
                  .Write "<TR><TD>Descriere</TD><TD>" & Err.Description &✄
 "</TD>"
                  .Write "<TR><TD>Sursa</TD><TD>" & Err.Source & "</TD>"
                  .Write "<TABLE>"
            End With
      End Sub

      Dim numeFis
      Dim textFis

      numeFis ="D:\Text1.htm" 'numele fisierului
      On Error Resume Next

      Err.Clear
      Call scrieFisText(numeFis, True, "<H1>Prima linie</H1>")
      If Err.Number <> 0 Then
            Err.Raise Err.Number, Err.Source & " -> Call scrieFisText()", _
            Err.Description & " -> Fisierul" & numeFis & " nu poate fi ✄
 deschis."
            Call Afisare()
            Response.End 'Aceasta eroare este critica
      End If

      Err.Clear
      Call adaugaFisText(numeFis, True, "<H2>A doua linie</H2>")
      If Err.Number <> 0 Then
            Err.Raise Err.Number, Err.Source & " -> Call adaugaFisText()", _
            Err.Description & " -> Nu se pot adauga date la fisierul " &✄
 numeFis
            Call Afisare()
      End If
      Call adaugaFisText(numeFis, True, "<H3>A treia linie</H3>")

      Err.Clear
      textFis = citesteFisText(numeFis)
      If Err.Number <> 0 Then
            Err.Raise Err.Number, Err.Source & " -> citesteFisText()", _
            Err.Description & " -> Nu se pot citi date din fisierul " &✄
 numeFis
            Call Afisare()
      End If

      If Len(Trim(textFis)) = 0 Then
            Err.Raise Err.Number, Err.Source & " -> citesteFisText()", _
            Err.Description & " -> Motivul erorii este necunoscut!"
            Call Afisare()
      End If

      Response.Write textFis
```

```
      numeFis ="D:\Text123.htm" ' stergere fisier inexistent
      Call stergeFisText(numeFis)
      If Err.Number <> 0 Then
            Err.Raise Err.Number, Err.Source & " -> Call stergeFisText()", _
            Err.Description & " -> Fisierul " & numeFis & " NU poate fi✂
 sters."
            Call Afisare()
      End If
      On Error GoTo 0
%>
</BODY>
</HTML>
```

La începutul aplicației se activează "trecerea peste erori" cu `On Error Resume Next`. Înainte de apelarea funcțiilor de manipulare a fişierelor erorile sunt şterse cu `Err.Clear`. Dacă după un astfel de apel apare o eroare, secvența `If Err.Number <> 0` o va prinde. Din acest moment avem următoarele variante:

- dacă eroarea apărută este critică, atunci se afişează un mesaj de eroare, după care aplicația se opreşte. În exemplul prezentat o astfel de eroare este dată de imposibilitatea creării fişierului în care dorim să scriem mai departe date;
- dacă eroarea nu este critică se afişează un mesaj de eroare (în aplicația prezentată prin apelul procedurii `Afisare()`), apoi se continuă execuția codului ca urmare a folosirii lui `On Error Resume Next`;
- ultima variantă este aceea de ignorare a erorii, însă această modalitate de reacție este permisă doar dacă suntem siguri de apariția ei şi de modalitatea de rezolvare a problemei.

Dincolo de erorile legate de manipularea fişierelor, o problemă ar putea fi un fişier care nu conține date. Cazul este ciudat deoarece noi suntem cei care am scris datele în el. Totuşi, o astfel de eroare ar putea fi teoretic posibilă şi o putem prinde verificând lungimea şirului care stochează datele citite din fişier.

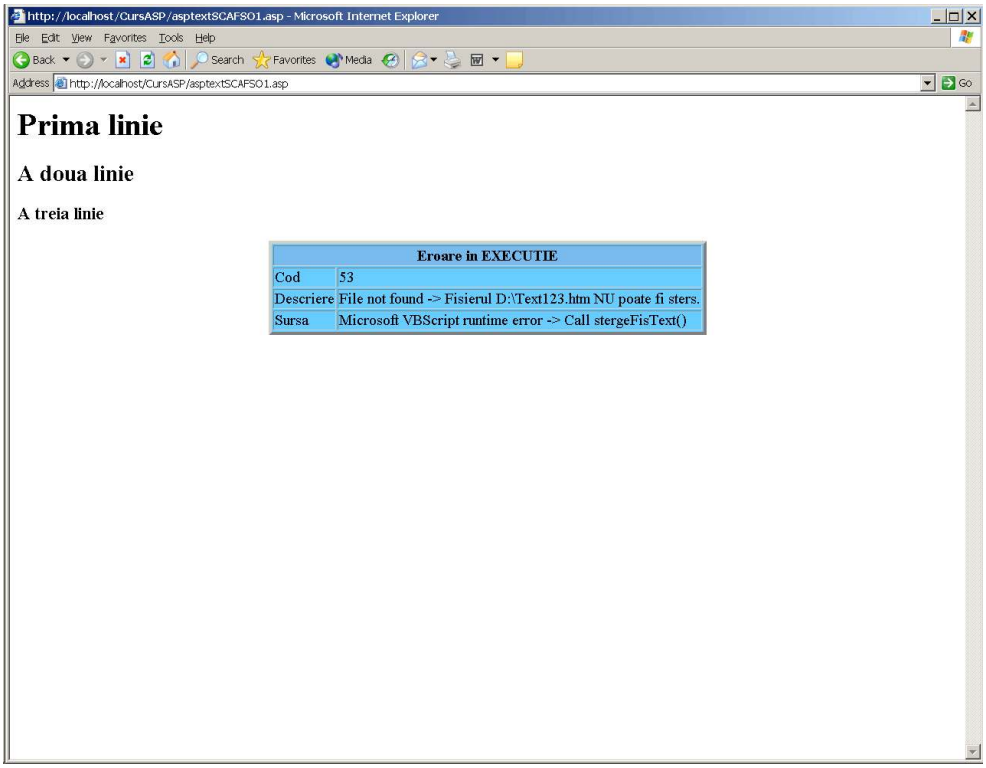


Figure 57 Folosirea lui Err.Raise la extinderea mesajelor de eroare afişate

Orice eroare poate fi tratată imediat în secvența de cod în care a apărut. Pentru că nu am dorit să mai modific codul lui `bibFSO.inc` tratarea erorii de ştergere a unui fişier inexistent am făcut-o în secvența de cod care apelează funcția de ştergere în linia `Call stergeFisText(numeFis).` Aici se încearcă ştergerea unui fişier cu nume diferit de cel care a fost creat la începutul aplicației. Eroarea este afişată după cum se observă în **Figura 57**.

Stocarea erorilor în fişiere

Dacă dorim să ținem un istoric al erorilor apărute în funcționarea aplicației o variantă este stocarea lor într-un fişier text pe discul server-ului de Web. Pentru aceasta utilizatorul anonim al contului de Internet (`IUSR_MACHINENAME`) trebuie să aibă dreptul de scriere în directorul care va stoca fişierul erorilor. Chiar şi în lipsa acestor drepturi erorile pot fi înscrise în fişiere de tipul Windows IIS log. Pentru a înscrie o eroare în IIS log se foloseşte metoda `Response.AppendToLog` care are ca şi unic argument un şir ce nu poate conține caracterul virgulă(",").

Metoda funcționează doar dacă sunt făcute următoarele setări (sub Windows XP Professional şi cu IIS 5.1):

- Lansați în execuție aplicația *Computer Management* (vezi **Figura 58**) din *Programs➡Administrative Tools*;
- Clic pe *Default Web Site*, clic pe butonul din dreapta al mouse-ului, apoi din lista selectați *Properties* pentru a deschide fereastra de dialog a proprietăților (vezi **Figura 59**);
- Selectați butonul ***Web Site***, iar în această fereastră activați check box-ul *Enable Logging*, apoi verificați ca la *Activie log format* să fie selectată opțiunea *W3C Extended Log File Format*.
- Clic pe butonul **Properties** din regiunea *Enable Logging*, pe ecran apare fereastra din **Figura 60**;
- Notați locul în care se va stoca fişierul de log. În condiții normale, fişiere de log IIS sunt stocate în directorul `C:\WINDOWS\System32\LogFiles`. Datorită posibilităților de configurare locația implicită ar putea fi schimbată;
- Clic pe butonul ***Extended Properties***, aici verificați să fie activate check box-urile *URI Stem* şi *URI Query* ca în **Figura 61**. Metoda `Response.AppendToLog` lucrează corect doar dacă aceste opțiuni sunt activate.
- Selectați butoanele **Apply** apoi, închideți toate ferestrele.

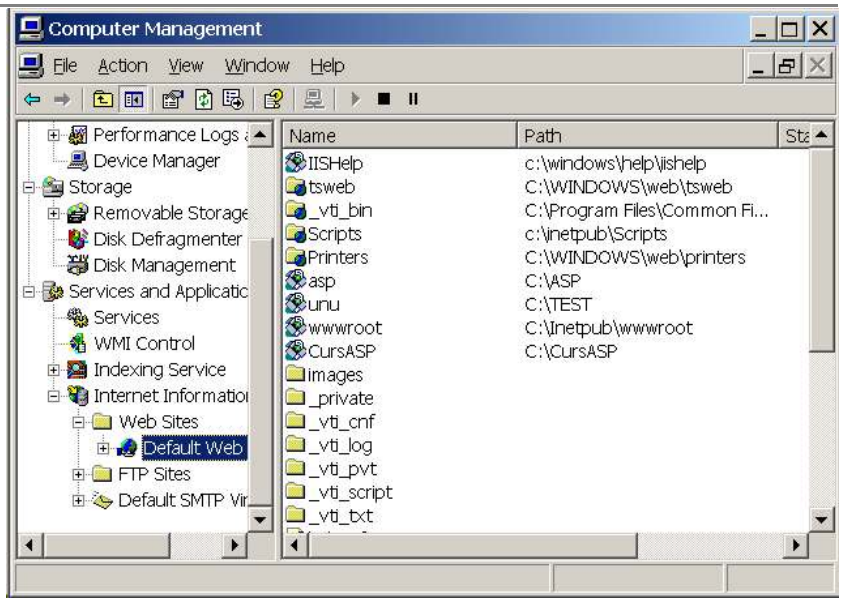


Figure 58
Fereastra principală a lui *Computer Management*

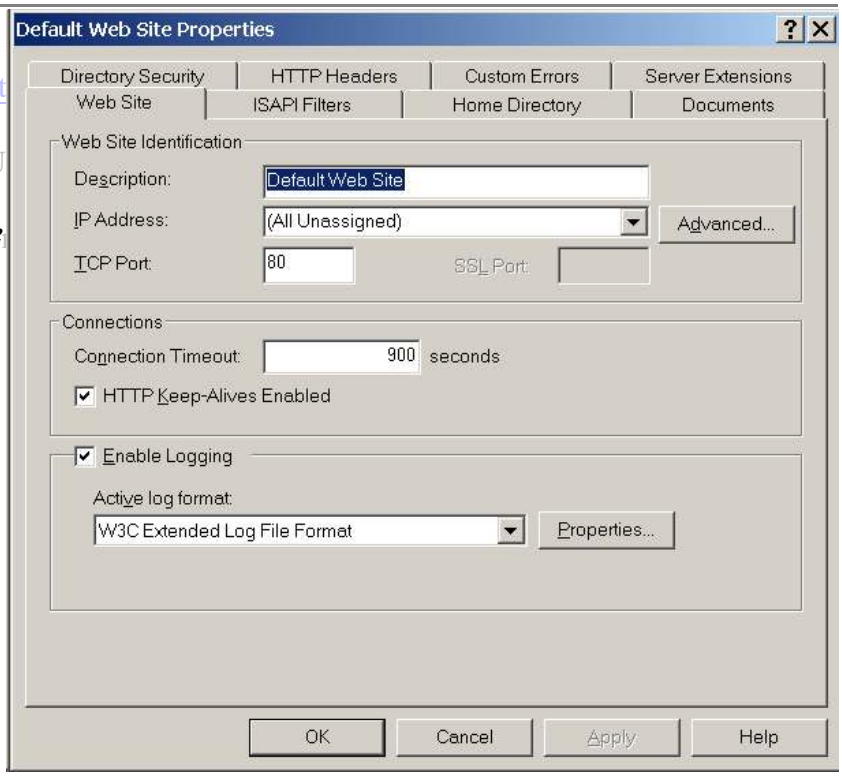


Figure 59
Fereastra de dialog *Default Web Site Properties* cu butonul ***Web Site*** selectat

Figure 60 Fereastra de dialog *Extended Logging Properties* cu butonul ***General Properties*** selectat

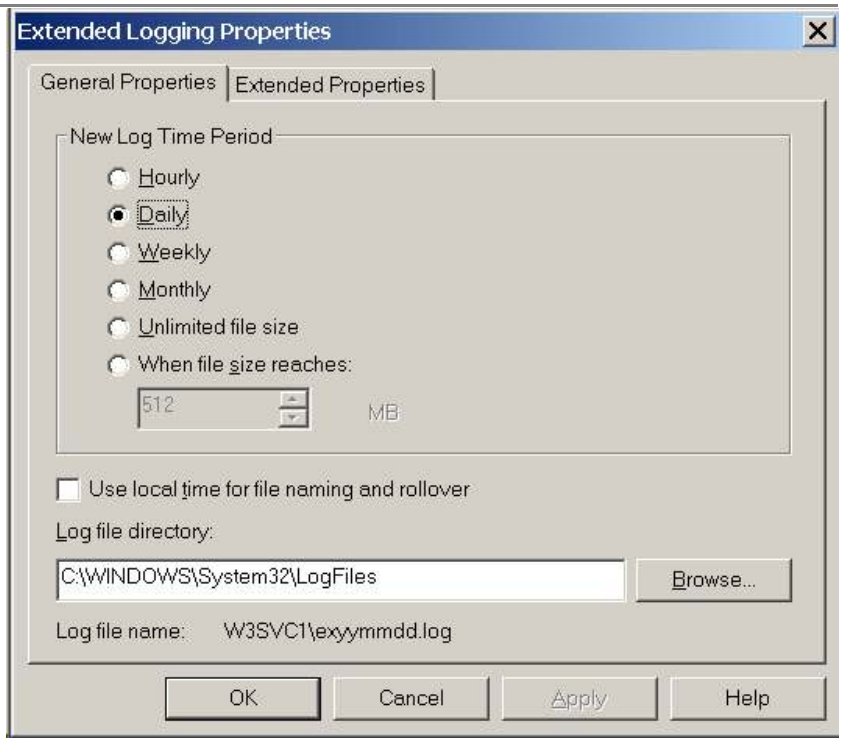
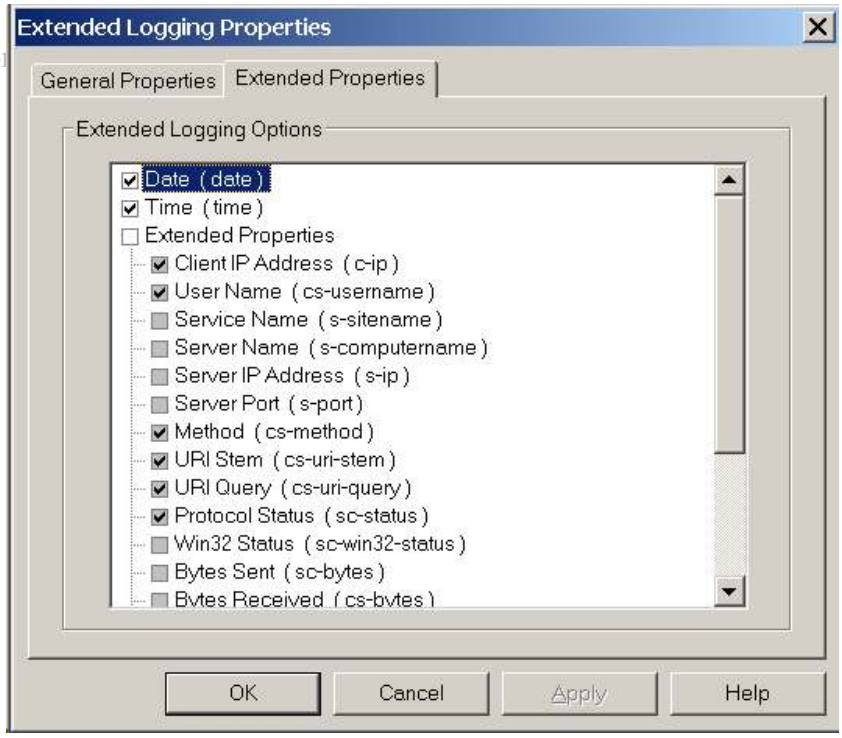


Figure 61 Fereastra de dialog Extended Logging Properties cu butonul Extended Properties selectat



Ca urmare a setărilor efectuate, în directorul `C:\WINDOWS\System32\LogFiles\W3SVC1`, apare un fişier de log pentru stocarea erorilor. Un nume al acestuia ar putea fi `ex021208.log`. Conţinutul lui este text, iar pentru aplicaţia care urmează, un apel al procedurii `IISlog` generează o linie de eroare ce are forma:

```
2002-12-08    13:45:50    127.0.0.1    -    GET    /CursASP/asptextSCAFSO2.asp
```

```
Fisier:+asptextSCAFSO2.aspCod:+53Sursa:+Microsoft+VBScript+runtime+error+->
+Call+stergeFisText()Desriere:+File+not+found+->+Fisierul+D:\Text123.htm+NU
+poate+fi+sters.Ora/Data:+12/8/2002+3:45:50+PM 200
```

Se poate observa mesajul de eroare având spații codificate în caractere +.

Fişierul `asptextSCAFSO2.asp`:

```
<!-- #include file="bibFSO.inc" -->
<HTML>
<HEAD>
<BODY>
<%
      Sub IISlog(Fisier, CodErr, SursaErr, DescriereErr)
            Dim e
            e="Fisier: " & Fisier
            e=e + "Cod: " & CodErr
            e=e + "Sursa: " & SursaErr
            e=e + "Desriere: " & DescriereErr
            e=e + "Ora/Data: " & Now
            Call Response.AppendToLog(e)
      End Sub

      Dim numeFis

      On Error Resume Next
      numeFis ="D:\Text123.htm" ' stergere fisier inexistent
      Call stergeFisText(numeFis)
      If Err.Number <> 0 Then
            Err.Raise Err.Number, Err.Source & " -> Call stergeFisText()", _
            Err.Description & " -> Fisierul " & numeFis & " NU poate fi✂
 sters."
            Call IISLog("asptextSCAFSO2.asp", Err.Number, Err.Source✂
 ,Err.Description)
      End If
      On Error GoTo 0
%>
</BODY>
</HTML>
```

E-mail

Terminologie

- *E-mail (electronic mail)* - este un mesaj transferat de la un utilizator al unui calculator, la un altul, deseori prin rețele de calculatoare sau modem, via linia telefonică. Mesajul are antet şi corp. Există mai multe standarde de transfer, cel mai utilizat fiind MIME (Multipurpose Internet Mail Extensions) deoarece acesta permite ataşarea la corpul mesajului a unor blocuri de tipul diferit de cel al unui text ASCII. Convențional, corpul se poate termina cu o semnătură. Antetul conține adresa de e-mail a destinatarului şi pe cea a expeditorului împreună cu ora, data la care s-a trimis mesajul şi un subiect. Mesajul este compus de expeditor, deseori cu un program special numit "Mail User Agent" (MUA). Acesta este interfața între utilizator şi programul numit "Mail Transfer Agent" (MTA) responsabil pentru transferul mesajelor e-mail la nivel local sau către un alt MTA aflat, probabil pe un alt calculator. MTA-urile unor calculatoare din rețele distincte comunică prim SMTP. Mesajul ajuns la destinatar se depune într-o căsută poştală, de obicei un fişier pe calculatorul destinatarului, de unde acesta poate fi vizualizat cu ajutorul unui program de citire al e-mail-urilor (care poate să fie acelaşi sau nu cu MUA folosit de expeditor).
- *SMTP (Simple Mail Transfer Protocol)* - un protocol folosit pentru transferul de e-mail-uri între calculatoare. Este un protocol de server (alte protocoale sunt folosite pentru accesarea mesajelor). Dialogul SMTP se derulează, de regulă, în spatele altor programe, sub controlul sistemului de transport al mesajelor.

În cele ce urmează se prezintă modul de folosire al Microsoft Collaboratory Data Objects pe NT (CDONTS) server pentru trimiterea unor e-mail-uri cu ajutorul script-urilor ASP. Pentru execuția codurilor trebuie să aveți acces la un server SMTP. Cea mai simplă variantă este SMTP-ul care vine cu IIS. O alternativă mai avantajoasă, datorită posibilităților de administrare comode, este cea a SMTP-ului din Microsoft Exchange Server. Microsoft permite programarea CDONTS prin COM. Modelul de obiecte corespunzător este prezentat în **Figura 62**.

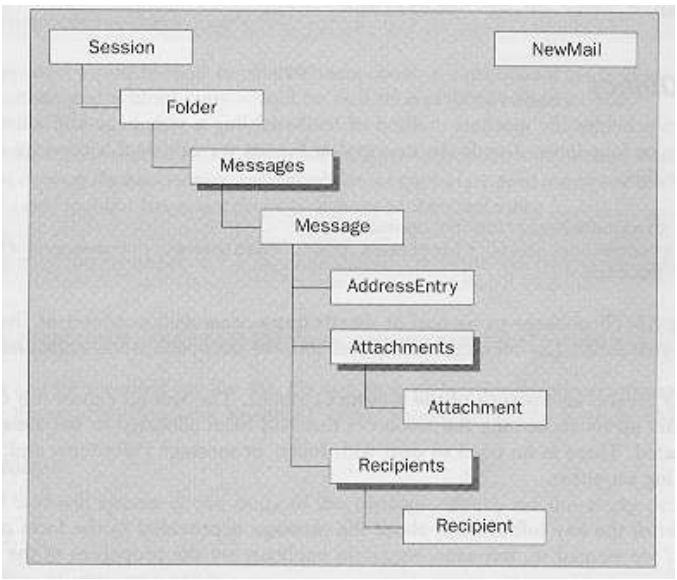


Figure 62 Modelul de obiecte CDONTS

În vârful ierarhiei CDONTS stă obiectul `Session`. Toate celelalte obiecte sunt subordonate lui cu excepția obiectului `NewMail`. Acesta din urmă permite trimiterea unui e-mail fără prea mult efort, în timp ce ierarhia obiectelor `Session` oferă posibilitatea unui control mai detaliat al

transferului de e-mail-uri.

Trimiterea unui e-mail cu obiectul NewMail

Codul care urmează este un exemplu simplu pentru a trimite un e-mail de la `webmaster@utcluj.ro` la `ramona@pcnet.ro`. Obiectul `NewMail` foloseşte proprietatea `To` pentru a stoca adresa de e-mail a destinatarului, proprietatea `From` pentru a stoca adresa de e-mail a expeditorului, iar proprietăţile `Subject` şi `Body` se folosesc pentru stocarea subiectului şi a corpului e-mail-ului de trimis.

```
<%@ Language=VBScript %>
<%
      Dim EMail
      Set EMail = Server.CreateObject("CDONTS.NewMail")
      EMail.To = "ramona@pcnet.ro"
      EMail.From = "webmaster@utcluj.ro"
      EMail.Subject = "Sarbatori fericite!"
      strBody = "Ramona" & "," & vbCrLf & vbCrLf & vbTab
      strBody = strBody & "Sarbatori Pline de Bucurii si"
      strBody = strBody & vbCrLf & vbCrLf & "1 An Nou Fericit."
      strBody = strBody & vbCrLf & vbCrLf & "http://www.utcluj.ro"
      EMail.Body = strBody
      EMail.Send
      Set EMail = Nothing

%>
```


`NewMail` permite şi ataşarea de fişiere unui mesaj prin metodele `AttachFile` sau `AttachURL`. `AttachFile` realizează ataşarea unui fişier, iar `AttachURL` include un URL care poate fi folosit de către destinatar pentru obţinerea datelor ataşate.

De exemplu, dacă dorim să trimitem ataşat la un e-mail fişierul, `C:\CursASP\salut.txt`, atunci se va folosi codul:

```
<%@ Language=VBScript %>
<%
      Dim EMail
      Set EMail = Server.CreateObject("CDONTS.NewMail")
      EMail.To = "ramona@pcnet.ro"
      EMail.From = "webmaster@utcluj.ro"
      EMail.Subject = "Ai 1 fisier text atasat."
      strBody = "e-mail trimis cu CDONTS"
      EMail.Body = strBody
      EMail.AttachFile = "C:\CursASP\salut.txt"
      EMail.Send
      Set EMail = Nothing
%>
```

O variantă de cod mai complicată este aceea în care se doreşte trimiterea unui e-mail la mai multe persoane. Pentru acest caz codul este:

```
<%@ Language=VBScript %>
<%
      Dim EMail
      Dim ListaAdrese
      Dim Adresa
      ListaAdrese= Array("ion@k.ro","vasile@pcnet.ro","tibi@dntcj.ro")
      For Each Adresa In ListaAdrese
            Set EMail = Server.CreateObject("CDONTS.NewMail")
            EMail.To = Adresa
            EMail.From = "webmaster@utcluj.ro"
```

```
            EMail.Subject = "Salutare"
            strBody = "e-mail trimis cu CDONTS"
            EMail.Body = strBody
            EMail.Send
            Set EMail = Nothing
      Next
%>
```

Observaţi că pentru fiecare adresă de e-mail din `ListaAdrese` se creează un nou obiect `NewMail`. Aceasta este singura modalitate de lucru deoarece după trimiterea unui e-mail obiectul `NewMail` nu mai poate fi folosit, deci el trebuie distrus şi apoi recreat pentru trimiterea unui nou e-mail.

Folosirea componentei w3 JMail pentru trimiterea unui e-mail

Pentru manipularea mesajelor de e-mail nu suntem limitaţi la CDONTS. Există un număr mare de producători de componente care asigură facilităţi mai multe şi au o mai bună flexibilitate decât cea a instrumentelor Microsoft. Câteva dintre cele mai populare sunt: JMail, ASPEMail, ASPFreeMail, SA-SmtpMail. Majoritatea acestora pot fi găsite prin site-ul **http://www.15seconds.com**. În cele ce urmează am ales să vă prezint un exemplu de lucru cu o componentă numită **JMail**. Ea trebuie descărcată de pe site-ul **http://www.dimac.net**. În momentul scrierii acestei cărţi varianta gratuită a componentei se prezintă sub forma unui fişier executabil cu numele **w3JMail4Free.exe**. După descărcarea pe server-ul de Web, programul executabil se va lansa în execuţie, el instalând automat componenta **JMail**. Instalarea se poate face şi manual, în acest caz fişierul `jmail.dll` trebuind copiat pe server-ul de Web. Apoi componenta se înregistrează prin comanda `regsvr32 jmail.dll`. Toate versiunile anterioare trebuie şterse înainte de această acţiune. Pentru rularea programului de instalare trebuie să aveţi drepturile de administrator pe Web server. Implicit, instalarea se va face în `C:\Program Files\Dimac\w3JMail4` împreună cu manualele de utilizare şi unele exemple de lucru simple. Instalarea va înregistra fişierul `jmail.dll` ca obiect COM.

Secvenţa de cod care urmează este un exemplu pentru trimiterea unui e-mail de la persoana cu adresa de e-mail `nume@domeniu.ro` către persoana cu adresa de e-mail `antaljr@mail.dntcj.ro`. Observaţi că adresa server-ului de mail este "`smtp.pcnet.ro`". E-mail-ul este trimis cu succes numai dacă persoana care se va lega la server are un cont valid pe acel server.În cazul particular al exemplului meu, eu am un cont DialUp făcut pe server-ul lui PCNET.

```
<%@ Language=VBScript %>
<%
      Dim JMail
      Dim Server
      Dim Destinatar, Subiect, Corp

' Informatii legate de Server si contul de e-mail al utilizatorului
      Server = "smtp.pcnet.ro"

' Adresa de e-mail a destinatarului
      Destinatar = "antaljr@mail.dntcj.ro"

      Subiect = "Aici se va scrie subiectul mesajului."
      Corp = "Aici vine corpul mesajului."

' Crearea obiectului JMail
      Set JMail = Server.CreateObject( "JMail.Message" )

' Silent se pune pe True daca doriti sa tratati erorile Dvs.
' And set logging to true to ease any potential debugging
      JMail.Silent = True
      JMail.Logging = True
```

```
' Majoritatea server-elor cer 1 adresa de e-mail valida pentru expeditor
      JMail.From = "nume@domeniu.ro"
      JMail.FromName = "Numele expeditorului"

' Adaugarea destinatarilor se face cu metoda addRecipients
' Ea poate fi folosita de mai multe ori.
      JMail.AddRecipient Destinatar, "Numele destinatarului este opţional"

' Daca este cazul, pot fi adaugati mai multi destinatari
'JMail.AddRecipient "Destinatar2@domeniu.ro"

' Formarea subiectului
      JMail.Subject = Subiect

' Urmeaza corpul mesajului
      JMail.Body = Corp

' Corpul mesajului se poate scrie si citi. Pentru adaugarea unui text se va
' folosi sintaxa:
'     JMail.Body = JMail.Body & " Salutare!"

' Mesajul se trimite cu metoda Send()
' Aceasta are ca parametru adresa server-ului de e-mail
' unde aveti Dumneavoastra contul


      If Not objJMail.Send(Server) then
          Response.write JMail.log
      Else
          Response.write "Mesajul a fost trimis cu succes!"
      End If
%>
```

Accesul la baze de date relaţionale

Terminologie

- *Relaţie (relation)* - în **matematică**, relaţia `R,` este o submulţime a produsului cartezian a două mulţimi `(A, B)`, `R : A x B`. Dacă `(a, b)` este un element din `R` notaţia `a R b` are semnificaţia de "`a` este în relaţia `R` cu `b`" (o notaţie echivalentă folosită, mai des, în teoria bazelor de date relaţionale este `R(a,b)`). O relaţie poate fi *reflexivă* `(a R a)`, *simetrică* `( a R b => b R a)`, *tranzitivă* `(a R b & b R c => a R c)`, *antisimetrică* `(a R b & b R a => a = b)` sau *totală* `(a R b sau b R a)`. În **bazele de date**, relaţia, este *un tabel* dintr-o bază de date relaţională.
- *Bază de date (database)* - una sau mai multe mulţimi de date persistente asociate cu o aplicaţie pentru actualizarea şi interogarea datelor;
- *Persistenţă* - proprietate a unei aplicaţii de a crea obiecte şi date care continuă să existe şi să-şi păstreze valorile între două execuţii ale aplicaţiei.
- *Model de date (data model)* - formalism matematic cu notaţii pentru descrierea structurilor de date şi a mulţimii operatorilor folosiţi pentru manipularea şi validarea datelor;
- *Model de date relaţional (relational model)* - un model de date introdus de E. F. Codd în 1970. În acest model datele sunt organizate în tabele. Mulţimea numelor coloanelor se numeşte schema tabelului. Datele pot fi manipulate folosind algebra relaţională. Limbajul SQL este implementarea unei astfel de algebre.
- *SQL (Structured Query Language)* - un limbaj neprocedural, de interfaţă, între utilizator şi bazele de date relaţionale dezvoltat de IBM în anii '70. Diferenţa esenţială între SQL şi alte limbaje este aceea că în SQL instrucţiunile specifică ce operaţii se vor face cu datele şi nu modul în care se realizează acestea.
- *Bază de date relaţională (relational database, relational database management system)* - o bază de date ce utilizează modelul de date dezvoltat de E. F. Codd. O bază de date relaţională permite definirea structurilor de date, operaţiile de stocare, de extragere a datelor şi a constrângerilor de integritate. Într-o bază de date relaţională (BDR), datele şi toate relaţiile între date sunt organizate în tabele. Un tabel este o colecţie de înregistrări (rânduri) unde fiecare înregistrare a tabelului conţine aceleaşi câmpuri. Anumite câmpuri pot fi desemnate chei. În acest caz viteza de căutare a unor valori specifice acelui câmp va creşte datorită folosirii de indexuri. Între înregistrările unor tabele diferite se pot realiza asocieri pe baza valorilor comune ale unui anumit câmp particular din fiecare tabel.
- *Cheie primară* - numărul minim de câmpuri care au fost alese pentru a identifica unic fiecare rând dintr-un tabel.
- *Constrângere de integritate* - o regulă care trebuie să rămână adevărată pentru ca baza de date să-şi păstreze integritatea. De exemplu, într-o bază de date genealogică fiecare individ trebuie să fie copilul unor părinţi. De asemenea, individul nu poate avea decât doi părinţi naturali.
- *Index* - o secvenţă de perechi (cheie, pointer) în care fiecare pointer poantează la o înregistrare din baza de date care conţine valoarea cheii într-un câmp particular. Indexul este sortat pe baza valorilor de chei şi permite căutarea rapidă a unei valori particulare de cheie.
- *Tabel* - un obiect al bazei de date care stochează datele. Este format din mai multe coloane. Fiecare coloană are asociat un tip de dată. Tipul coloanei se foloseşte pentru manipularea corectă a conţinutului.
- *Normalizare* - o relaţie dintr-o bază de date relaţională se zice că este într-o formă normală dacă satisface un anumit grup de constrângeri. Lucrarea iniţială a lui Codd definea trei forme normale.

- *Tuplă* - un obiect de date ce conţine una sau mai multe componente. Componentele unei tuple pot fi de tipuri diferite. Câteva exemple de tuple sunt: (1,2), ("Tuple", Da), (a, (x, y), c).
- *Tranzacție* - o unitate logică de interacțiune cu o BDR. Aceasta trebuie tratată într-o manieră coerentă şi sigură, independent de alte tranzacții. Pentru o tranzacție se garantează, fie realizarea ei completă, fie nerealizarea ei. Dacă o eroare conduce la realizarea parțială a tranzacției, tranzacția poate fi "derulată înapoi" pentru a opri lăsarea bazei de date într-o stare de inconsistență.
- *Interogare (query)* - acțiune de extragere a unor informații din baza de date.

Conceptul de dată

Pentru cei care au deja experiență în domeniul calculatoarelor, noțiunea de dată este asociată cu numere, şiruri de caractere, imagini, sunete ce pot fi stocate şi prelucrate pe calculator, eventual transmise mai departe prin rețea. Privind însă conceptul de dată general, data, reprezintă o reflectare, parțială, a realității. Deseori, cuvântul de dată apare în contextul conceptului de model. Necesitatea modelului apare datorită complexității lumii pe care încercăm să o surprindem. Abundența detaliilor ne depăşeşte. Din acest motiv, realitatea va fi simplificată de om, prin crearea unor modele. Modelul se obține prin eliminarea detaliilor şi păstrarea unor caracterstici minimale din realitate, fără a pierde însă veridicitatea modelului (care dintre carateristici sunt păstrare, deoarece se consideră a fi semnificative, respectiv de ce şi care anume se elimină asta este o altă carte). Datele reprezintă o măsură cantitativă a procesului de evoluție a caracteristicilor considerate semnificative pentru model (aceste caracteristici descriu modelul). Generalizând, datele reprezintă o colecție de observații pe marginea unor evenimente pe care le considerăm importante.

Datele în sine nu au nici o semnificație (ele nu au un înțeles). Ele pot însă deveni informație în urma asocierii directe acestora cu anumite caracteristici. Datele vor fi astfel încărcate cu semnificație ca urmare a interpretării lor într-un context particular. Această interpretare se poate face de către om sau, automat, de către un sistem de calcul. Daca considerăm numărul întreg “1100”, el este dată dar nu este informație. În contextul “Salariul meu este de 1100 RON”, data “1100” are deja un înțeles particular (este valoarea remunerării mele), devenind informație. În contextul “Salariul meu de 1100 RON este acceptabil.”, data “1100” permite trecerea la cunoaştere. Deoarece eu cunosc salariile unei mari majorități, sunt în stare să calculez un salar mediu şi să mă plasez undeva, în raport cu acesta. Procesul de plasare la nivelul de “acceptabil” are la bază un grup de prelucrări iar rezultatul de “acceptabil” are la bază un proces deductiv. Foarte pe scurt, datele reprezintă materialul primar, informația reprezintă date + înțeles, iar cunoaşterea este informație + prelucrare.

Scopul colectării datelor este obținerea de informații sau de cunoaştere ca urmare a unui proces de deducție bazat pe experientă sau pe reguli ce poate fi implementate la nivelul unui sistem de calcul. În bazele de date, data, va fi organizată sub forma unui singur articol de informație (o singură valoare). Deoarece în procesul de colectare a datelor pot să apară erori, termenul de integritate a datelor definește mecanismele de prevenire a erorilor.

Modelul de date relaţional

Se numeşte **model de date**, un formalism matematic cu notații pentru **descrierea structurilor de date** şi a mulțimii **operatorilor** folosiți pentru manipularea şi validarea datelor. Modelul de date relațional este introdus de E.F. Codd în 1970. În acest model datele sunt structurate (organizate) în relații care au nume. Mulțimea numelor coloanelor se numeşte schema relației. Structura de date logică numită **relație** este bidimensională, fiind cunoscută sub numele de **tabel**,

în baza de date fizică. Cele două dimensiuni ale structurii sunt denumite rânduri şi coloane. Datele modelului se pot modifica folosind operatorii algebrei relaţionale (limbajul SQL este implementarea unei astfel de algebre), ei asigurând selectarea, inserarea, modificarea şi ştergerea datelor din tabele, iar **restricțiile de integritate** asigură păstrarea consistenței datelor.

Conceptul de redundanţă

Redundanța datelor apare atunci când o anumită dată este stocată în mai multe tabele ale bazei de date. Acest surplus de date necesită o atenție specială atunci când se aduc modificări asupra conținutului valorilor stocate în tabele. Dacă valorea unei date redundante este modificată, atunci va trebui să ne asigurăm că toate copiile respectivei date au fost aduse la nouă valoare. În această situație se zice că valoarea respectivă este consistentă în baza de date. Dacă nu toate copiile au fost actualizate la noua valoare se zice că data este inconsistentă. Se zice că o bază da date este în stare consistentă dacă toate datele acesteia sunt consistente, altfel este în stare inconsistentă.

Terminologii alternative ale modelului relaţional

Termenii folosiți în modelul de date relaționali sunt, deseori, producători de confuzie. Deoarece modelul are ca sursă algebra relațională, matematicienii preferă să folosească anumite denumiri. Proiectanții bazelor de date au şi ei termeni specifici, dar echivalenți cu cei din algebra. În final, administratorii de sistem folosesc şi ei propriul lor jargon atunci când realizează operații de întreținere a bazelor de date. Tabelul care urmează îşi propune să prezinte termenii alternativi folosiți de cele trei categorii de persoane care folosesc acest model.

Algebra relațională	Proiectant BDR	Administrator de sistem
Relație	Tabel	Fişier
Tuplă	Rând	Înregistrare
Atribut	Coloană	Câmp

Atributele reprezintă date elementare care sunt nume ale coloanelor unei relații. De exemplu, relația `chirie(IDClient, NumeClient)` conține atributele `IDClient`, `NumeClient`. Valorile actuale pentru aceste atribute ale relației se stochează în **tuple** sau rânduri ale tabelului. Atributele se grupează pe baza dependenței de valorile din cheia primară. O **cheie primară** (CP) este un atribut, sau un grup de atribute, care indentifică unic un rând din tabel. Un tabel poate avea o singură cheie primară. Deoarece valorile din CP se folosesc pentru identificare, ele nu pot fi vide (nule). Asocierea între două relații se face, tipic, printr-un atribut comun celor două relații. Atributul comun este, de obicei, CP într-un tabel şi **cheie străină** (CS) în celălalt. Regulile de **integritate referențială** dictează valorile CS dintr-o relație în raport cu valorile CP din cealaltă relație.

Proiectarea unei baze de date relaționale se face pe baza **regulilor de normalizare** care dictează apartenența atributelor la relații. Aceste reguli vor fi descrise, pe scurt, în continuare. Folosirea modelului de date relațional normalizat asigură minimizarea redundanței datelor împreună cu protecția contra anomaliilor de inserare, ştergere şi actualizare care apar ca urmare a definirii unor relații incorecte.

Fie relația, cu numele **ProiecteMembri** şi atributele `IDPr`, `NumePR`, `Suma`, `IDMem`, `Membru`, `Tip Adresa` şi `Telefon` având şapte tuple, prezentată în continuare:

Tabelul ProiecteMembri

IDPr	NumePr	Suma	IDMem	Membru	Calitate	Catedra	Telefon
PR1	Proiect1	1000000	ME1	Ion	Director	Mecanica	111111
PR1	Proiect1	1000000	ME2	Vasile	Membru	Mecanica	222222
PR1	Proiect1	1000000	ME3	Ada	Membru	Programare	333333
PR2	Proiect2	2000000	ME4	Nelu	Director	Programare	444444
PR2	Proiect2	2000000	ME3	Ada	Membru	Programare	333333
PR3	Proiect3	3000000	ME2	Vasile	Membru	Mecanica	222222
Pr3	Proiect3	3000000	ME1	Ion	Director	Mecanica	111111

Pentru ilustrarea problemelor legate de redundanța datelor se va compara prima relație prezentată în tabelul de mai sus, cu cele două relații din tabelele următoare:

Tabelul Proiecte

IDPr	NumePr	Suma
PR1	Proiect1	1000000
PR2	Proiect2	2000000
PR3	Proiect3	3000000

Tabelul Membri

IDPr	IDMem	Membru	Calitate	Catedra	Telefon
PR1	ME1	Ion	Director	Mecanica	111111
PR1	ME2	Vasile	Membru	Mecanica	222222
PR1	ME3	Ada	Membru	Programare	333333
PR2	ME4	Nelu	Director	Programare	444444
PR2	ME3	Ada	Membru	Programare	333333
PR3	ME2	Vasile	Membru	Mecanica	222222
PR3	ME1	Ion	Director	Mecanica	111111

Observați că, în cazul primei relații, în tabel avem stocate 8 x 7 = 56 de celule de date. În cazul următoarelor două tabele, pentru stocarea aceloraşi informații, s-au folosit numai 3 x 3 + 7 x 6 = 51 de celule de date. Putem spune că datorită descompunerii relației inițiale în două noi relații, spațiul necesar pentru stocarea informațiilor a scăzut ca urmare a eliminării unor repetări de date, adică redundanța datelor s-a micşorat.

Conceptul dependenţei funcţionale

Dependența funcțională descrie legătura dintre atributele unei relații sau vorbind în jargonul proiectanților de baze de date descrie legătura între coloanele unui tabel. Fiind dat un tabel T şi două nume de coloane A şi B, se zice că A determină funcțional pe B în raport cu T şi scriem A ➜ B, dacă şi numai dacă pentru orice două rânduri r_i şi r_j de fiecare dată când avem $\mathbf{r_i(A) = r_j(A)}$ atunci avem şi $\mathbf{r_i(B) = r_j(B)}$. Dacă A determină funcțional pe B, de fiecare dată când avem două valori indentice în A, valorile corespunzătoare din B trebuie sa fie şi ele indentice. Pentru tabelul următor dorim să verificăm următoarele dependențe funcționale: A ➜ B, A ➜ C, C ➜ A, AB ➜

C, AC ➜ B.

Tabelul T

	A	B	C
r1	1	4	2
r2	3	5	6
r3	3	4	6
r4	7	3	8
r5	9	1	0

Dependența funcțională A ➜ B *nu se verifică* deoarece $\mathbf{r_2(A) = r_3(A)}$, dar $\mathbf{r_2(B) \neq r_3(B)}$. Dependența funcțională A ➜ C *se verifică* deoarece pentru cele două rânduri unde avem valori egale se verifică $\mathbf{r_2(A) = r_3(A)}$, dar $\mathbf{r_2(C) = r_3(C)}$. Dependența funcțională C ➜ A se verifică şi ea. Dependența funcțională AB ➜ C *se verifică* deoarece nu există rânduri care să aibă valori identice pentru coloanele A şi B, iar în lipsa unor astfel de valori dependența funcțională nu poate fi infirmată. Dependența funcțională AC ➜ B *nu se verifică* deoarece $\mathbf{r_2(AC) = r_3(AC)}$, dar $\mathbf{r_2(B) \neq r_3(B)}$.

În cazul relației **ProiecteMembri** se poate stabili dependența funcțională `IDPr` ➜ `NumePr`, atributul `IDPr` mai poartă denumirea de *variabilă independentă* sau *determinant*, iar atributul `NumePR` se numeşte *variabilă dependentă*.

Tipuri de dependențe funcționale

Fiind dat un tabel T şi coloanele A, B pentru care avem A ➜ B, se zice că B este **dependent funcțional complet** de A, dacă şi numai dacă nu există o submulțime W a valorilor din coloana A astfel încât W ➜ B. Dacă mulțimea W există se zice că B este **dependent funcțional parțial** de A. Un exemplu de coloane între care există dependență funcțională completă ar putea fi NrComanda,Produs ➜ Cantitate. Există situații în care între coloanele unui tabel pot fi scrise simultan, dependențele funcționale: A ➜ B, B ↛ A, B ➜ C, C ↛ A şi A ➜ C. În aceste condiții se zice că atributul C este **dependent funcțional tranzitiv** de atributul A. În practică o dependența funcțională tranzitivă ar putea lua forma NrFactura ➜ IDClient ➜ NumeClient.

Conceptul de normalizare

În bazele de date relaționale, normalizarea defineşte un proces de descompunere reversibil, care se aplică asupra unei mulțimi de relații pentru a se obține o nouă mulțime de relații mai simple şi mai uniforme ca structură. Analiza relațiilor inițiale se face la nivel de structură, utilizând atribute pentru care se vor căuta dependențe funcționale. În practică, analiza se face prin aplicarea unor teste relațiilor pentru a determina dacă acestea satisfac sau nu cerințele unei anumite forme normale. Inițial, E. F. Codd, a definit trei forme normale cunoscute sub numele de prima formă normală (se prescurtează 1NF), a două formă normală (se prescurtează 2NF) şi a treia formă normală (se prescurtează 3NF), ulterior, acestora sau mai adăugat şi altele. În cele ce urmează voi discuta numai primele trei forme normale. O relație care trece testul 3NF, trece şi testele 2NF şi 1NF. O relație care trece 2NF, trece şi testul 1NF.

Etapele procesului de normalizare

Normalizarea este o tehnică de analiză a relațiilor bazată pe cheile primare şi dependența funcțională. Ea descompune o relație nenormalizată, progresiv, în mai multe relații cu număr de atribute mai mici şi tuple mai puține, până la atingerea unui nivel optim al descompunerii. Câteva

dintre rezultatele pozitive ale normalizării sunt:

- spaţiul folosit pentru stocarea datelor este mai mic;
- actualizarea tabelelor se poate face mai eficient;
- descrierea bazei de date devine mai clară.

Forma nenormală

Se numeşte formă nenormală (*unnormalized form* - UNF) un tabel care conţine unul sau mai multe grupuri de date care se repetă. Acest tabel nu este încă o relaţie, deoarece, uneori, la intersecţia dintre un rând şi o coloană avem mai mult de o dată (sunt mai multe valori distincte).

Exemplul următor îşi propune să normalizeze tabelul iniţial **ProiecteMembri** care stochează date legate de proiecte de cercetare şi membrii acestor proiecte. Relaţia în UNF este:

ProiecteMembri(<u>IDPr</u>, NumePr, Suma, **Membri**(IDPr, IDMem, Membru , ...))

Tabelul ProiecteMembri

IDPr	NumePr	Suma	IDMem	Membru	Calitate	Catedra	Telefon
PR1	Proiect1	1000000	ME1 ME2 ME3	Ion Vasile Ada	Director Membru Membru	Mecanica Mecanica Programare	111111 222222 333333
PR2	Proiect2	2000000	ME4 ME3	Nelu Ada	Director Membru	Programare Programare	444444 333333
PR3	Proiect3	3000000	ME2 ME1	Vasile Ion	Membru Director	Mecanica Mecanica	222222 111111

Pentru ca acest tabel să devină o relaţie acesta va trebui normalizat.

Prima formă normală

Un tabel este adus la prima formă normală (*First Normal Form - 1NF*) dacă:

- este o relaţie validă (nu mai există grupuri de valori care se repetă la nivelul unei celule);
- o cheie unică (Cheia Primară - CP) a fost găsită pentru identificarea fiecărui rând (în exemplele următoare, CP va fi subliniată);
- toate atributele sunt dependente de toate atributele cheii sau de o parte dintre ele.

Prima etapă constă în reorganizarea tabelului pe linii şi coloane, astfel încât, la intersecţia între o linie şi o coloană să avem cel mult o singură dată.

Tabelul ProiecteMembri

IDPr	NumePr	Suma	IDMem	Membru	Calitate	Catedra	Telefon
PR1	Proiect1	1000000	ME1	Ion	Director	Mecanica	111111
			ME2	Vasile	Membru	Mecanica	222222
			ME3	Ada	Membru	Programare	333333
PR2	Proiect2	2000000	ME4	Nelu	Director	Programare	444444
			ME3	Ada	Membru	Programare	333333
PR3	Proiect3	3000000	ME2	Vasile	Membru	Mecanica	222222
			ME1	Ion	Director	Mecanica	111111

Cea de a două etapă constă în completarea datelor lipsă în noul tabel. Pentru aceasta se vor copia valorile care până acum nu se repetau, pentru fiecare rând pentru care se repeta. Noul tabel ce se obține este:

Tabelul ProiecteMembri

IDPr	NumePr	Suma	IDMem	Membru	Calitate	Catedra	Telefon
PR1	Proiect1	1000000	ME1	Ion	Director	Mecanica	111111
PR1	Proiect1	1000000	ME2	Vasile	Membru	Mecanica	222222
PR1	Proiect1	1000000	ME3	Ada	Membru	Programare	333333
PR2	Proiect2	2000000	ME4	Nelu	Director	Programare	444444
PR2	Proiect2	2000000	ME3	Ada	Membru	Programare	333333
PR3	Proiect3	3000000	ME2	Vasile	Membru	Mecanica	222222
PR3	Proiect3	3000000	ME1	Ion	Director	Mecanica	111111

Noul tabel obținut nu este încă o relație deoarece nu are CP. Observați că în acest moment, alegerea lui `IDPr` pe post de CP nu mai asigură identificarea unică a unui rând.

Cea de a treia etapă a trecerii la UNF la 1NF se face prin identificarea grupurilor care se repetă în noul tabel. Acestea sunt hașurate cu linii înclinate în reprezentarea următoare.

Tabelul ProiecteMembri

IDPr	NumePr	Suma	IDMem	Membru	Calitate	Catedra	Telefon
PR1	Proiect1	1000000	ME1	Ion	Director	Mecanica	111111
PR1	Proiect1	1000000	ME2	Vasile	Membru	Mecanica	222222
PR1	Proiect1	1000000	ME3	Ada	Membru	Programare	333333
PR2	Proiect2	2000000	ME4	Nelu	Director	Programare	444444
PR2	Proiect2	2000000	ME3	Ada	Membru	Programare	333333
PR3	Proiect3	3000000	ME2	Vasile	Membru	Mecanica	222222
PR3	Proiect3	3000000	ME1	Ion	Director	Mecanica	111111

Dacă ele există, se extrag împreună cu CP a relației principale, într-un nou tabel (în exemplul prezentat el se numește **Proiecte**). Celelalte coloane cu valorile corepunzătoare vor fi stocate şi ele într-un nou tabel (în exemplul prezentat el se numeşte **Membri**). Pentru ca descompunerea relației inițiale să se facă fără pierdere (descompunerea realizează o separare a atributelor relației inițiale, care trebuie să poată fi refăcute din noile relații) este nevoie să găsim un atribut (sau grup de atribute) care să permită refacerea tabelului inițial. În acest scop, CP a noii relații, care era şi CP a "relației inițiale" trebuie extrasă şi în cel de al doilea tabel. Deoarece aceasta este străină de noua relație, fiind extrasă numai pentru ca descompunerea să se facă fără pierdere, ea mai poartă şi denumirea de cheie străină (CS). Pentru noile relații formate se determină CP. Conform celor scrise 1NF este:

```
Proiecte(IDPr, NumePr, Suma)
Membri(IDPr, IDMem, Membru, Calitate, ...)
```

Anomaliile relaţiilor 1NF

Redundanțele relațiilor 1NF conduc la posibilitatea apariției unor anomalii la manipularea datelor.

Există două tipuri de anomalii: de actualizare şi de inserare/ştergere. În practică, deoarece la nivelul limbajului SQL există implementați operatorii de actualizare (UPDATE), de inserare (INSERT) şi de ştergere (DELETE) se preferă tratarea a trei cazuri distincte. Ultimele două cazuri însă pot fi considerate o singură categorie deoarece, o relație care are anomalia la ştergere, o va avea automat şi pe cea la inserare.

Tabelul Membri

IDPr	IDMem	Membru	Calitate	Catedra	Telefon
PR1	ME1	Ion	Director	Mecanica	111111
PR1	ME2	Vasile	Membru	Mecanica	222222
PR1	ME3	Ada	Membru	Programare	333333
PR2	ME4	Nelu	Director	Programare	444444
PR2	ME3	Ada	Membru	Programare	333333
PR3	ME2	Vasile	Membru	Mecanica	222222
PR3	ME1	Ion	Director	Mecanica	111111

Anomalia la inserare apare atunci când dorim să inserăm în tabel un nou membru care însă încă nu s-a hotărât la care proiect ar dori să lucreze. Deoarece CP a acestei relații este una compusă din atributele (IDPr, IDMem) ar însemna să lăsăm vidă o parte din CP integritatea datelor fiind compromisă.

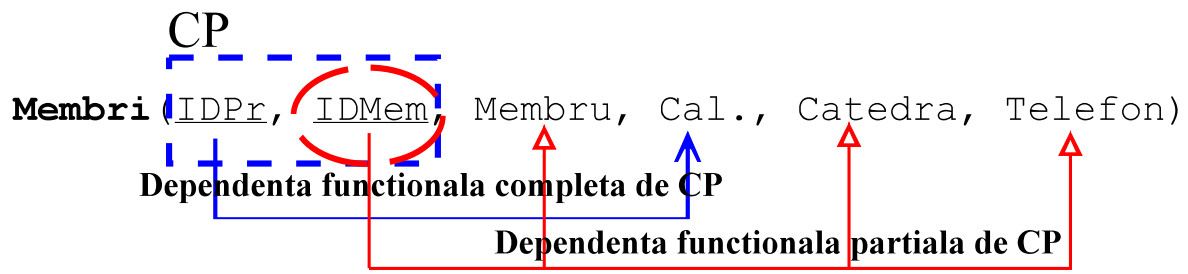
Anomalia la ştergere apare atunci când ştergem şi ultimul rând din tabel corespunzător unui proiect particular (de exemplu, se şterg toate rândurile cu IDPr = PR2). Menționez că această ştergere se face din punctul de vedere al proiectului, dar va duce şi la pierderea membrilor (care ar putea fi utilizați în alte proiecte). În acel moment pe lângă faptul că dispare asocierea între respectivul proiect şi membrii lui, se pot pierde şi informațiile legate de catedrele la care lucrau respectivii membri. În cazul nostru, Ada mai apare în tabel, dar toate informațiile legate de Nelu vor fi pierdute.

Anomalia la actualizare apare ca urmare a redundanței datelor, de exemplu, la nivelul atributului Catedra. Deoarece unele nume de catedre se repetă, în situația în care cineva se mută de la o catedră la o alta, trebuie să verificăm toate aparițiile respectivului membru şi să le modificăm corespuzător. Dacă, din anumite motive, nu se fac toate modificările necesare, atunci, baza de date ajunge în starea de inconsistență.

A două formă normală

O relație este în a două formă normală (*Second Normal Form - 2NF*) dacă este deja în 1NF şi toate atributele ne-cheie sunt dependente complet de CP (nu există dependențe parțiale). Trecerea de la 1NF la 2NF se face prin extragerea dependențelor parțiale într-o nouă relație.

Membri(IDPr, IDMem, Membru, Tip, Catedra, Telefon)



Proiecte(IDPr, NumePr, Suma) este deja în 2NF pentru că are un singur atribut în cheia primară (CP). În urma eliminării dependențelor parțiale pentru cea de a două relație din 1NF

Membri(IDPr, IDMem, Membru, Calitate, Catedra, Telefon)

devine

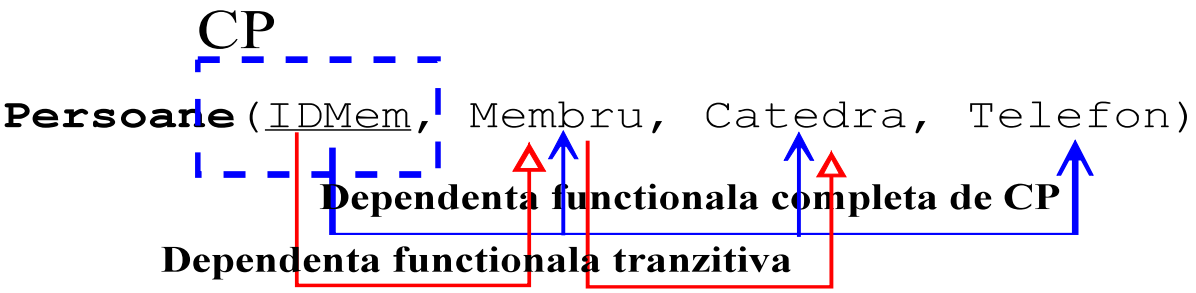
Membri(IDPr, IDMem, Calitate)
Persoane(IDMem, Membru, Catedra, Telefon)

Şi în cazul relațiilor aduse la 2NF sunt posibile anomalii. De exemplu, anomalia la inserare apare atunci când dorim să inserăm un rând în tabelul **Membri** (adică avem nevoie de un **IDMem** valid în CP), dar nu avem încă inserate rânduri în tabelul **Persoane**. Anomalia la ştegere apare la nivelul relației **Persoane** atunci când ştergem un membru care este unicul participant dintr-o catedră în tabelul **Persoane**, situație în care se pierd toate datele legate de catedra respectivă. Anomalia la actualizare ar putea să apară deoarece pot exista mai mulți membri din aceeaşi catedră (deci, numele respectivei catedre se va repeta).

A treia formă normală

O relație este în a treia formă normală (*Third Normal Form - 3NF*) dacă este în 2NF şi toate dependențele tranzitive au fost extrase într-o altă relație. Se verifică dependența atributelor ne-cheie de alte atribute ne-cheie. **Membri**(IDPr, IDMem, Calitate) este deja în 3NF deoarece nu sunt dependențe tranzitive. În urma extragerii dependențelor tranzitive relația din 2NF

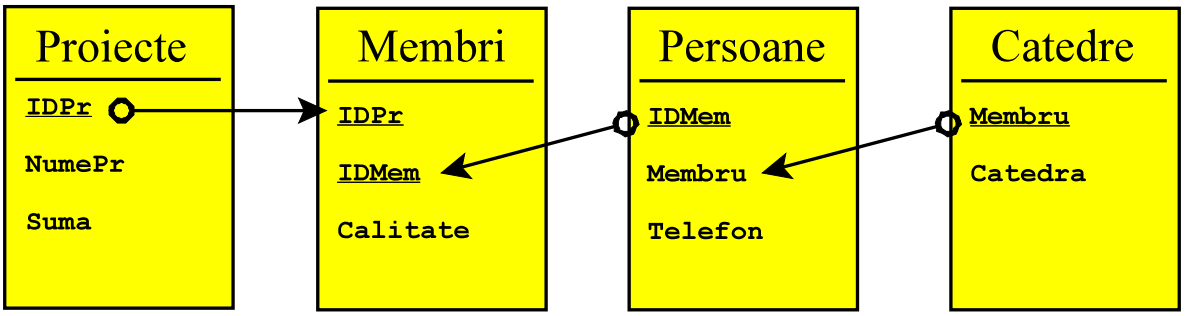
Persoane(IDMem, Membru, Catedra, Telefon)



devine

Persoane(IDMem, Membru, Telefon)
Catedre(Membru, Catedra)

În final, se obține următoarea structură logică normalizată pentru stocarea persoanelor în baza de date relațională.



Această nouă organizare a bazei de date conține tabelele, atributele, CP şi asocierile formate între noile tabele pe baza valorilor comune stocate în CP şi în CS. Asocierile sunt marcate în figură prin săgeți care pleacă de la CP către CS.

Introducere în SQL

SQL este un limbaj particular născut şi dezvoltat în baza răspândirii modelului relațional. În ultimii ani a devenit limbajul standard pentru BDR. Obiectivele limbajului sunt:

- crearea bazelor de date şi a relațiilor;
- realizarea operațiilor de întreținere a BDR;
- realizarea interogărilor simple şi complexe.

Ca urmare a standardizării ISO limbajul SQL are două componente majore:

- limbajul pentru definiția datelor (*Data Definition Language - DDL*): pentru definirea structurii bazei de date şi controlul accesului la date;
- limbajul pentru manipularea datelor (*Data Manipulation Language - DML*): pentru extragerea şi actualizarea datelor.

O *instrucțiune SQL* este formată din *cuvinte rezervate* şi *cuvinte definite de utilizator*. Cuvintele rezervate (numite şi cuvinte cheie) sunt partea fixată a limbajului deoarece au o semnificație predefinită. Ele trebuie scrise fără a fi despărțite pe mai multe linii. Cuvintele definite de utilizator sunt "inventate" de el şi reprezintă nume ale obiectelor bazei de date cum sunt tabelele, coloanele etc. Cuvintele unei instrucțiuni SQL se scriu conform mulțimii regulilor de sintaxă specifice limbajului SQL. Multe dintre dialectele SQL folosesc caracterul "punct şi virgulă" ";" pe post de terminator de instrucțiune SQL, însă conform standardului, acestea sunt opționale.

Pentru exemplele care vor fi dezvoltate în continuare este suficientă prezentarea a numai patru dintre instruncțiunile DML din SQL:

- `SELECT` - pentru interogarea bazei de date;
- `INSERT` - pentru inserarea datelor într-un tabel;
- `UPDATE` - pentru actualizarea datelor unui tabel;
- `DELETE` - pentru ştergerea datelor unui tabel.

Instrucţiunea SELECT

Permite extragerea şi afişarea datelor din unul sau mai multe tabele.

SELECT cu un singur tabel

Pentru extragerea tuturor numelor de proiecte stocate în câmpul `NumePr` din tabelul **Proiecte** scriem:

```
SELECT NumePr FROM Proiecte;
```

Pentru extragerea numelor de proiecte şi a sumei plătite pe proiect, stocată în câmpul `Suma`, din acelaşi tabel se scrie:

```
SELECT NumePr, Suma FROM Proiecte;
```

Deoarece multe interogări SQL folosesc toate câmpurile unui tabel. Folosirea unui asterisc "*" este o formă prescurtată de scriere pentru "toate câmpurile".

```
SELECT * FROM Proiecte;
```

Atunci când dorim să impunem condiții cu privire la rândurile extrase se va folosi clauza `WHERE`. De exemplu, pentru extragerea proiectelor din tabelul **Proiecte** care au o suma mai mare de 500000 în câmpul `Suma` se va scrie:

```
SELECT * FROM Proiecte WHERE Suma > 500000;
```


Dacă dorim ca rândurile sortate să fie în ordinea crescătoare alfabetică a proiectelor vom folosi clauza `ORDER BY` astfel


```
SELECT * FROM Proiecte ORDER BY NumePr;
```

Pentru sortarea în ordinea descrescătoare a numelor de proiecte scriem:

```
SELECT * FROM Proiecte ORDER BY NumePr DESC;
```

SELECT cu mai multe tabele

Dacă datele de extras sunt stocate, ca urmare a normalizării, în mai multe tabele, una dintre metodele pentru conectarea tabelelor pe baza unui câmp comun foloseşte instrucțiunea `JOIN`. Să presupunem că dorim să extragem toate proiectele şi tipurile membrilor pe baza câmpurilor `IDPr` din **Proiecte** şi `IDPr` din **Membri**. `IDPr` este CP în **Proiecte** şi CS în **Membri**. Între proiecte şi membri este o *relație unu-la-mulți*, adică unui rând din tabelul **Proiecte** pot să-i corespundă mai multe rânduri ale tabelului **Membri**. Instrucțiunea `INNER JOIN` face ca din cele două tabele să fie extrase doar rândurile care au valori comune în câmpurile de conectare.

```
SELECT NumePr.Proiecte, Membri.Calitate
FROM Proiecte INNER JOIN Membri ON Proiecte.IDPr = Membri.IDPr;
```

Instrucţiunea INSERT

Instrucțiunea `INSERT` permite adăugarea unui rând într-un tabel. Una dintre formele ei este:

```
INSERT INTO NumeTable (lista de câmpuri)
VALUES (lista de valori)
```

Dacă dorim să inserăm un nou proiect în tabelul **Proiecte** scriem:

```
INSERT INTO Proiecte (IDPr, NumeProiect, Suma1)
```

```
VALUES ('PR4', Proiect nou', 100000);
```

Instrucţiunea UPDATE

Instrucțiunea `UPDATE` permite modificarea conținutului unui rând existent dintr-un tabel. Forma ei este:

```
UPDATE NumeTabel
SET NumeColoana1=Valoare1 [,NumeColoana2=Valoare2 ...]
[WHERE Conditie]
```

Pentru a creşte cu 20% valoarea din câmpul `Suma` ale tuturor proiectelor din tabelul **Proiecte** scriem:

```
UPDATE Proiecte SET Suma=Suma*1.20;
```

Pentru a creşte valoarea din câmpul `Suma` cu 20% a proiectului cu numele "`Proiect1`" din tabelul **Proiecte** scriem:

```
UPDATE Proiecte
SET Suma=Suma*1.20
WHERE NumePr="Proiect1";
```

Instrucţiunea DELETE

Instrucțiunea permite ştergerea unor rânduri ale unui tabel. Forma ei este:

```
DELETE FROM NumeTabel
[WHERE Conditie]
```

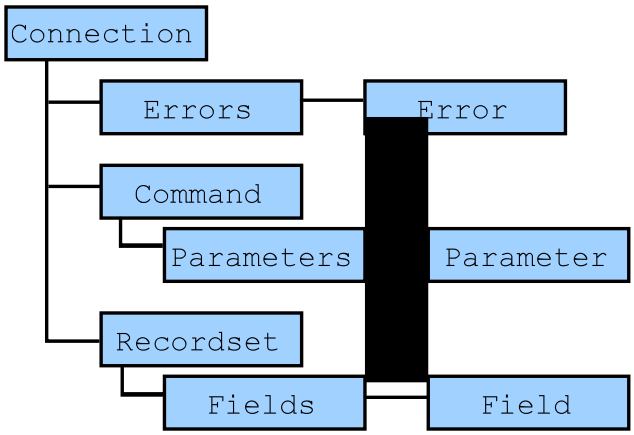

Pentru a şterge rândul corespunzător proiectului cu numele "`Proiect1`" din tabelul **Proiecte** scriem:

```
DELETE FROM Proiecte
WHERE NumePr="Proiect1";
```

Introducere în ADO

Interfața între ASP şi bazele de date relaționale se face prin biblioteca de obiecte *ActiveX Data Objects* (ADO). ADO permite aplicațiilor client să acceseze şi să manipuleze date prin orice furnizor de servicii OLE DB (OLE DB este o componentă a unui SGBDR care implementează accesul eficient prin rețea locală sau Internet la date stocate conform modelului relațional, e-mail, foi de calcul etc.). ADO poate accesa date stocate conform mai multor modele de date, însă în cele ce urmează se va prezenta manipularea datelor din bazele de date Microsoft Access. ADO conține obiecte pentru conectarea la o sursă de date şi manipularea acestora. În modelul de obiecte ADO (**Figura 63**), obiectul `Connection` definește o sesiune de lucru pentru un utilizator al unei surse de date particulare. Obiectul ADO `Command` este folosit pentru executarea instrucțiunilor SQL legate de o anumită sursă de date, iar obiectul `Recordset` pentru vizualizarea conținutului unui tabel sau a rezultatelor execuției unei instrucțiuni SQL.

Figure 63
Modelul de obiecte ADO



Setări specifice pentru deschiderea conexiunii

Obiectul `Connection` reprezintă conexiunea fizică la sursa de date. După inițializarea şi stabilirea conexiunii, pot fi utilizate metode specifice pentru accesarea sursei de date la care ne-am conectat (în cazul nostru, baza de date). Principalele metode şi proprietăți ale lui `Connection` se prezintă în tabelele următoare:

Metodă	Descriere
`Open`	Deschide o conexiune nouă cu sursa de date.
`Close`	Închide conexiunea împreună cu obiectele dependente de aceasta.
`Execute`	Execută o interogare; tipic, aceasta este o instrucțiune SQL.
`BeginTrans`	Începe o tranzacție nouă.
`CommitTrans`	Salvează modificările realizate în timpul tranzacției.
`RollBackTrans`	Anulează toate modificările făcute în timpul tranzacției.

Proprietate	Descriere
`ConnectionString`	Conține informațiile necesare stabilirii conexiunii.
`State`	Indică starea conexiunii.
`CursorLocation`	Setează sau întoarce locul cursorului.
`Mode`	Setează modul de citire, scriere sau citire/scriere al conexiunii.

Starea proprietăților poate fi setată cu ajutorul unor constante care sunt stocate în fişierul text *adovbs.inc*. Fişierul se instalează, implicit, cu calea *C:\Program Files\Common Files\System\ado* odată cu ASP-ul. Pentru folosirea acestor constante, fiecare aplicație ASP care foloseşte ADO va trebui să includă acest fişier în ea prin linia:

```
<!-- #INCLUDE FILE="calerelativălaaplicație\adovbs.inc" -->
```

Dacă şirul care reprezintă calea este prea lung cel mai simplu este să copiați *adovbs.inc* în directorul virtual care conține aplicațiile ASP, caz în care includerea se va face cu linia:

```
<!-- #INCLUDE FILE="adovbs.inc" -->
```

Obiectele ADO pentru accesul la date sunt stocate de firma Microsoft într-un singur proiect cu numele *ActiveX Data Objects Database* (ADODB). Întrucât sunt deja disponibile mai multe versiuni, o referință la proiectul ADODB folosit pentru accesul la date poate fi setată în fişierul `global.asa` prin linia următoare:

```
<!--METADATA TYPE="TypeLib" NAME="Microsoft ActiveX Data Objects 2.6 Library"
UUID="{00000206-0000-0010-8000-00AA006D2EA4}" VERSION="2.6"-->
```

Linia de mai sus este corectă numai pentru ADODB versiunea 2.6. Ea va trebuie modificată în cazul versiunilor mai vechi sau mai noi. De exemplu, pentru versiunea 2.1 ea este:

```
<!--METADATA TYPE="TypeLib" NAME="Microsoft ActiveX Data Objects 2.1 Library"
UUID="{00000201-0000-0010-8000-00AA006D2EA4}" VERSION="2.1"-->
```

Pentru a putea experimenta câteva dintre acțiunile elementare cu bazele de date va trebui să aveți instalat sistemul de gestiune a bazelor de date Microsoft Access (acesta vine cu produsul Microsoft Office Professional) împreună cu fişierele sale de exemple. În cazul în care Access a fost instalat în directorul implicit, copiați fişierul

C:\Program Files\Microsoft Office\Office10\Samples\Northwind.mdb

în directorul fizic corespunzător directorului virtual al aplicației ASP din care se face accesarea bazei de date Access cu numele *Northwind.mdb*. În cazul meu voi copia fişierul în directorul *C:\CursASP*.

Deschiderea unei conexiuni noi se realizează prin secvența de cod:

```
Dim cnx
Dim sirCnx
Set cnx = Server.CreateObject("ADODB.Connection")
cnx.Mode = adModeRead
cnx.CursorLocation = adUseClient
sirCnx = "Provider=Microsoft.Jet.OLEDB.4.0;" &
         "Data Source=C:\cursasp\Northwind.mdb;"
cnx.Open sirConex
```

Alegerea valorii proprietății Mode

În ADO se foloseşte termenul de **cursor** pentru a descrie comportamentul referinței la înregistrarea curentă dintr-o mulțime de înregistrări. Un cursor este obiectul ce permite deplasarea printre înregistrările unei mulțimi de înregistrări. ADO permite definirea modului de lucru al cursorului unei conexiuni prin proprietatea `Mode`, acesta poate lua valori constante ce sunt prezentate în tabelul următor:

Nume constantă	Valoare	Semnificație
`adModeUnknown`	`0`	Modul de lucru prin conexiune este nespecificat.
`adModeRead`	`1`	Permite citirea datelor prin conexiune (implicit).

Nume constantă	Valoare	Semnificație
adModeWrite	2	Permite scrierea datelor prin conexiune.
adModeReadWrite	3	Permite citirea şi scrierea datelor prin conexiune.
adModeShareDenyRead	4	Opreşte citirea datelor printr-o altă conexiune.
adModeShareDenyWrite	8	Opreşte scrierea datelor printr-o altă conexiune.
adModeShareExclusive	&Hc	Opreşte deschiderea bazei de date printr-o altă conexiune.
adModeShareDenyNone	&H10	Permite ca toate celelalte conexiuni să se ataşeze bazei de date în orice mod.
adModeRecursive	&H400000	Se foloseşte împreună cu toate modurile ce au Share în nume, mai puțin adModeShareDenyNone. Asigură propagarea setărilor subînregistrărilor corespunzătoare înregistrării curente. De exemplu, pentru a opri citirea subînregistrărilor se va folosi expresia de constante adModeShareDenyRead + adModeRecursive.

Setarea proprietăţii ConnectionString

Proprietatea ConnectionString are mai multe părți, dintre care unele sunt opționale, în funcție de tipul şirului de conexiune folosit. Tipic, în cazul unei aplicații client-server care foloseşte SQL Server sau MSDE, el trebuie să conțină următoarele părți:

Parte	Exemplu
Numele furnizorului	Provider=SQLOLEDB;
Numele server-ului de baze de date	Data Source=(local);
Numele bazei de date la care ne conectăm	Database=Northwind;
Indentificatorul utilizatorului (UID)	UID=NumeUtilizator;
Parola utilizatorului (PWD)	PWD=ParolăUtilizator;

Şirul de conectare pentru acest caz se poate forma astfel:

```
sirCnx = "Provider=SQLOLDEB;" & _
         "Data Source =(local);" & _
         "Database = Nortwind;" & _
         "UID = NumeUtilizator;"  & _
         "PWD = ParolăUtilizator;"
```

În cazul în care dorim să ne conectăm la o bază de date Microsoft Access, în locul driver-ului SQLOLEDB se va folosi driver-ul Microsoft Access după cum urmează:

```
sirCnx = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
         "Data Source=C:\cursasp\Northwind.mdb;"
```

Alegerea proprietăţii CursorLocation

ADO suportă lucrul cu două tipuri de cursoare: **cel client** (client-side) şi **cel server** (server-side). Un cursor client se creează pe calculatorul clientului (adică cu utilizatorul bazei de date), iar cel server se creează pe aceeaşi maşină cu server-ul (care poate fi chiar calculatorul pe care lucrează utilizatorul sau poate fi un server oarecare). Pentru cursorul server majoritatea datelor rezultate dintr-o interogare vor fi păstrate pe server. Pe măsură ce utilizatorul se deplasează prin mulțimea de înregistrări noile înregistrări se trimit, rând pe rând, clientului. Cu cât clientul se deplasează mai mult prin mulțimea de înregistrări cu atât creşte şi traficul generat de el în rețea. Avantajul acestui cursor este acela de a putea începe imediat deplasarea prin mulțimea de înregistrări, deşi numai o parte mică dintre datele ei au fost, momentan, generate. Dezavantajul cursorului este acela că numai furnizorul SQLOLEDB suportă acest tip de cursor. Cursorul client se inițializează mai încet deoarece toate datele trebuie transferate de pe server pe client, dar are avantajul de a putea avea mai mulți clienți deschişi simultan şi de a putea deconecta obiectul mulțime de înregistrări de la o anumită conexiune. Acest avantaj este important în aplicațiile Web deoarece conexiunile consumă multe resurse ale sistemului care ar putea fi eliberate ca urmare a deconectării. Două constante sunt definite pentru alegerea cursorului, `adUseServer` (implicită) pentru cursorul server şi `adUseClient` pentru cursorul client.

Deschiderea conexiunii

După setările făcute, deschiderea conexiunii se face cu metoda `Open`. Dacă aceasta se execută fără erori se obține o conexiune la baza de date. După ce conexiunea a fost deschisă se va putea extrage o mulțime de înregistrări prin ea. Metoda `Open`, a obiectului `Recordset`, se foloseşte pentru stabilirea conexiunii fizice cu sursa de date şi pentru deschiderea unui obiect mulțime de înregistrări care reprezintă rândurile unui tabel sau rezultatul unei interogări.

Gestiunea tranzacţiilor cu obiectul Connection

Tranzacțiile reprezintă o *unitate de lucru logică* utilă atunci când dorim să aducem mai multe modificări datelor din bază "dintr-o singură bucată". În lumea calculatoarelor acronimul ACID se foloseşte pentru descrierea caracteristicilor unei tranzacții:

- *atomicitate* - deşi se poate ca mai multe înregistrări să fie deja modificate, dacă apare o eroare undeva, întreaga tranzacție este anulată, iar sistemul este readus la starea avută înaintea începerii tranzacției;
- *consistență* - o tranzacție nu lasă baza de date într-o stare de inconsistență. Dacă s-au făcut o parte dintre modificările specifice tranzacției acestea poate fi anulate, în cazul în care tranzacția eşuază ulterior într-un punct oarecare;
- *izolare* - o tranzacție se comportă ca şi când ar fi izolată de alte tranzacții din baza de date;
- *durabilitate*- după ce modificările tranzacției sunt finalizate, acestea persistă dincolo de orice defecte hard care pot surveni în sistemul de calcul.

Când se folosesc comenzi pentru modificarea datelor stocate în baza de date (`UPDATE`, `INSERT` sau `DELETE`) fiecare comandă este tratată ca o tranzacție. Atunci când comanda, dintr-un oarecare motiv, nu poate fi executată, toate efectele corespunzătoare ei sunt anulate. Există însă cazuri în care trebuie să executăm un grup de comenzi ca o singură tranzacție. Pentru ca grupul de comenzi să fie tratat ca o singură tranzacție, înainte de prima comandă a grupului se va folosi metoda `BeginTrans`. Aceasta marchează începutul tranzacției. În continuare se vor scrie comenzile care modifică baza de date, apoi se foloseşte metoda `CommitTrans` pentru finalizarea tuturor modificările corespunzătoare comenzilor tranzacției. Înainte de această acțiune se va folosi proprietatea `Count` a obiectului `Error` din conexiune pentru a verifica lipsa erorilor. O secvență de cod, tipică, este:

```
cnn.BeginTrans

'Aici se scriu comenzile care modifică baza de date

If cnn.Errors.Count = 0 Then
      cnn.CommitTrans
Else
      cnn.RollbackTrans
End If
```

Dacă nu sunt erori, atunci modificările se vor finaliza în baza de date. Dacă sunt erori, metoda `RollbackTrans` se foloseşte pentru a termina tranzacția curentă şi pentru a readuce baza de date la starea în care era înainte de începerea tranzacției.

Unul dintre cazurile clasice de folosire a tranzacțiilor este acela în care trebuie să scriem date în două sau mai multe tabele, fie cu certitudinea realizării tuturor înscrierilor, fie cu cea a lipsei modificărilor aduse tuturor tabelelor.

Obiectul mulţime de înregistrări (Recordset)

Obiectul `Recordset` este un tabel virtual de valori organizat pe linii şi pe coloane. El extinde posibilitățile de stocare şi de parcurgere a datelor la cele de căutare, sortare, actualizare şi ştergere. În plus, valorile stocate în acest obiect pot să vină din mai multe tabele, unele valori ale coloanelor pot fi calculate pe baza unor expresii, iar mulțimea de înregistrări poate fi transformată, simplu, într-un tabel sau un şir.

Deschiderea unei mulţimi de înregistrări

Obiectul `Recordset` are, şi el, o metodă `Open` care necesită mai multe argumente. Sintaxa este:

```
Recordset.Open TextComandă, Conexiune/ŞirConexiune, TipCursor, TipBlocare,
Opţiuni
```

Argumentul `TextComandă` conține un şir SQL pentru selecția unor înregistrări.

Argumentul `Conexiune/ŞirConexiune` conține o referință la un obiect `Connection` deschis sau un şir conexiune valid. Dacă în codul unei pagini se face o singură referire la baza de date specificarea lui `ŞirConexiune` face ca ADO să creeze automat obiectul `Connection`. În cazul în care se vor face mai multe referiri la baza de date dorim să avem un control mai strict asupra obiectului `Connection` şi îl deschidem cu metoda `Open`, iar apoi îl închidem cu metoda `Close`.

Argumentul `TipCursor` poate lua valori constante prezentate în tabelul următor:

Constantă	Descriere	Avantaje	Dezavantaje
`adOpenDynamic`	Mulțime de înregistrări dintr-un tabel sau dintr-o altă sursă de date.	Afişează toate modificările făcute de alți utilizatori.	Permite folosirea marcajelor numai dacă furnizorul le suportă. Nu poate lucra cu jet OLEDB în acest mod.

adOpenKeyset	Mulțime de pointer-i (bookmarks) care referă datele unui tabel sau unei interogări din baza de date.	Afişează toate modificările făcute de alți utilizatori. Suportă lucrul cu marcaje întotdeauna.	Nu afişează noile înregistrări adăugate de alți utilizatori. Împiedică accesul la înregistrările şterse de alți utilizatori.
adOpenStatic	Copie a unei mulțimi de înregistrări aşa cum era ea la momentul creării ei.	Întotdeuna suportă marcaje. Singurul tip de mulțime de înregistrare este client-side cursor.	Nu reflectă modificările aduse datelor în regimul de lucru cu mai mulți utilizatori.
adOpenForwardOnly (implicit)	Copie a unei mulțimi de înregistrări aşa cum era ea la momentul creării ei.	Mai rapidă decât cursorul adOpenStatic.	Permite navigarea în mulțime numai în față.

Argumentul TipBlocare poate lua valori constante prezentate în tabelul următor.

Constantă	Descriere
adLockReadOnly	Mulțimea de înregistrări nu poate fi editată.
adLockPessimistic	Blocarea mulțimii de înregistrări este pesimistă.
adLockOptimistic	Blocarea mulțimii de înregistrări este optimistă.

Dacă un utilizator blochează baza de date, atunci, un alt utilizator nu va putea insera date sau actualiza valori decât după ce utilizatorul care a blocat baza a închis mulțimea de înregistrări corepunzătoare blocării. Blocarea pesimistă face datele blocate invizibile altor utilizatori ai aplicației, închiderea mulțimii de înregistrări face ca blocarea pesimistă să dispară. Blocarea optimistă blochează înregistrările cu puțin timp înainte de actualizare sau inserare şi le deblochează imediat după aceasta. Alți utilizatori vor putea vedea datele blocate, dar dacă mai mulți utilizatori, simultan, încearcă să actualizeze aceleaşi date apar probleme de acces la ele.

Argumentul Opțiuni spune lui ADO cum să evalueze parametrul TextComandă. În cazul aplicațiilor prezentate, deoarece parametrul TextComandă este întotdeauna un şir SQL valoarea constanta folosită va fi adCmdText.

Parcurgerea unei mulţimi de înregistrări

Parcurgerea unei mulțimi de înregistrări, în vederea afişării înregistrărilor, se face cu ajutorul metodei MoveNext. Aceasta realizează avansul, de la înregistrarea curentă, la cea următoare. Metoda EOF (End-Of-File) se foloseşte pentru a determina dacă s-a trecut dincolo de ultima înregistrare din mulțime. EOF va întoarce în acest caz valoarea True, altfel, valoarea întoarsă este False. Secvența de cod, tipică, pentru afişarea tuturor înregistrărilor din mulțime este:

```
Do While Not rst.EOF
      'Afisează înregistrare curentă
      rst.MoveNext 'Treci la înregistrarea următoare
```

```
Loop
```

Aplicația din fișierul `aspado1.asp` se folosește de obiectele `Connection` și `Recordset` pentru accesarea bazei de date *Northwind.mdb*. Din tabelul **Employees** se vor afișa, pentru fiecare angajat, numele (câmpurile `LastName` și `FirstName`), adresa (câmpul `Address`) și orașul (câmpul `City`).

Fișierul `aspado1.asp`:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ADO si ASP</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<!-- #INCLUDE FILE ="..\Program Files\Common Files\System\ado\adovbs.inc" -->

<H1>Baza de date: Northwind.mdb, Tabel: Employees</H1>
<%
      Dim cnn
      Dim rst
      Dim sirSQL
      Dim sirConex

      Set cnn = Server.CreateObject("ADODB.Connection")
      Set rst = Server.CreateObject("ADODB.Recordset")
      sirConex = "Provider=Microsoft.Jet.OLEDB.4.0;" &
      "Data Source=C:\cursasp\Northwind.mdb;"

      cnn.Mode = adModeRead
      cnn.CursorLocation = adUseClient
      cnn.Open sirConex

      sirSQL = "SELECT LastName, FirstName, Address, City FROM Employees"
      rst.Open sirSQL, cnn, adOpenStatic, adLockOptimistic, adCmdText
%>
<P>
<TABLE BORDER = 2 BGCOLOR=YELLOW>
      <THEAD>
            <TH>  Nume angajat   </TH>
            <TH>   Adresa   </TH>
            <TH>   Oras   </TH>
      </THEAD>
<%    Do While Not rst.EOF %>
      <TR>
            <TD><%=rst("LastName")%>   <%=rst("FirstName")%></TD>
            <TD><%=rst("Address")%></TD>
            <TD><%=rst("City")%></TD>
      </TR>
<%
      rst.MoveNext
      Loop

      rst.Close
      cnn.Close
%>

</TABLE>

<%
      Set rst = Nothing
      Set cnn = Nothing
%>
</BODY>
```

```
</HTML>
```

Rezultatele aplicației se prezintă în **Figura 64**.

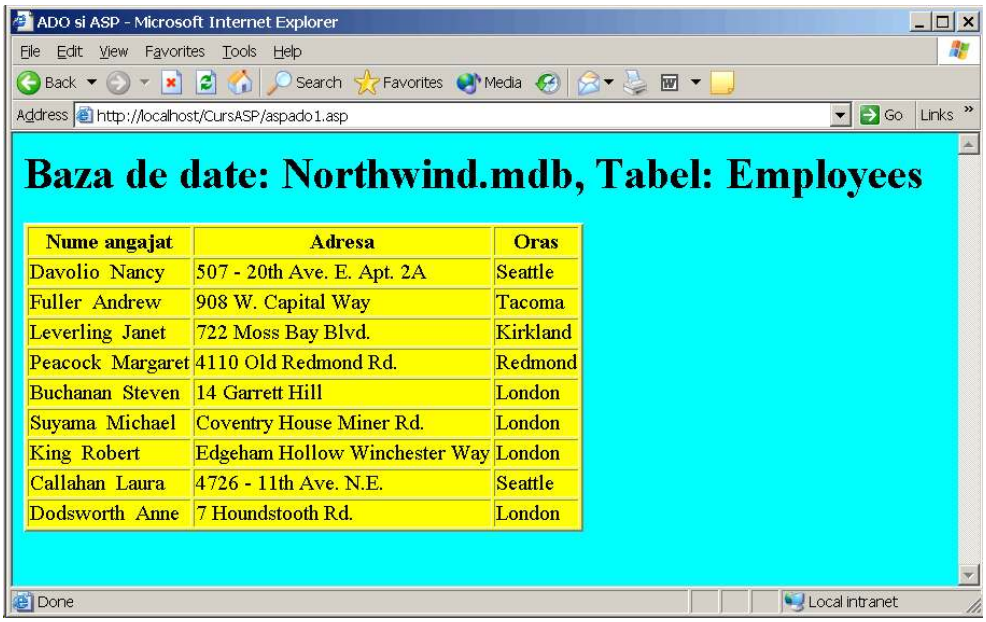


Figure 64 Afișarea cu ADO a unui tabel din baza de date Northwind.mdb

Metoda Connection.Execute

Cel mai simplu mod de interacțiune cu baza de date, prin ADO, este metoda `Execute` a obiectului `Connection`. Ea permite transmiterea unei comenzi, deseori SQL, unei surse de date. Dacă instrucțiunea SQL întoarce înregistrări, atunci se creează un obiect `Recordset`. Această mulțime de înregistrări poate fi stocată prin linia:

```
Set rst = cnn.Execute(sirSQL, , adCmdText)
```

Metoda `Execute` are trei argumente, primul este o instrucțiune SQL, un nume de tabel sau de interogare; al doilea este un nume de variabilă care va stoca numărul înregistrărilor afectate după terminarea lui `Execute`; al treilea este o constantă care descrie tipul instrucțiunii. Pentru cele SQL valoarea constantei este `adCmdText` și asigură întoarcerea obiectului `Recordset`. Pentru cazul în care nu se dorește întoarcerea lui, constanta se formează prin suma `adCmdText + adExecuteNoRecords`.

Exemplul din `aspado2.asp` folosește conexiunea la baza de date *Northwind.mdb* pentru inserarea cu `Execute` a unei înregistări noi (`Nume=Ion, Prenume=Vasile, Adresa=Crinului, nr. 12, Oras=Cluj`) în tabelul **Employees**, și pentru afișarea noii mulțimi de înregistrări. Efectele aplicației se prezintă în **Figura 65** .

NOTĂ

Unul dintre mesajele de eroare comune a lui ADO cu ASP este:

Microsoft JET Database Engine (0x80004005)
Operation must use an updateable query.
/CursASP/aspado2.asp, line 27

Motivul cel mai comun este acela că utilizatorul contului Internet Guest (IUSR_MACHINE), care este implicit în grupul "Everyone", nu are dreptul "Write" asupra fișierului care este baza de date. Pentru rezolvarea problemei mergeți în Explorer și folosiți fereastra Security pentru a corecta dreptul de acces a lui Internet Guest la baza de date.

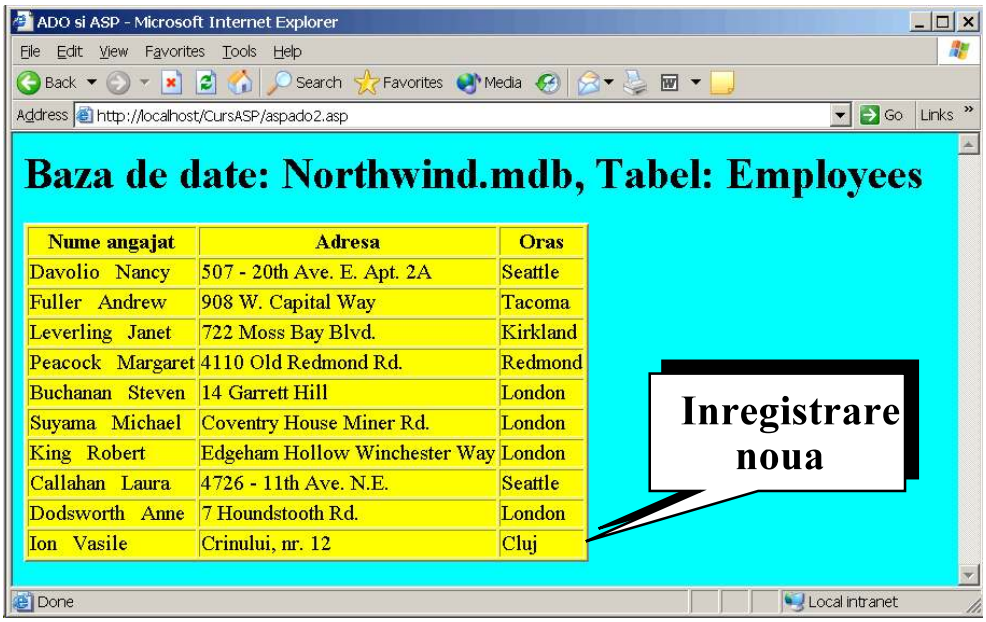


Figure 65 Inserarea unei înregistrări în tabelul Employees din baza de date Northwind.mdb

Fişierul `aspado2.asp`:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ADO si ASP</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<!-- #INCLUDE FILE ="..\Program Files\Common Files\System\ado\adovbs.inc" -->

<H1>Baza de date: Northwind.mdb, Tabel: Employees</H1>
<%
      Dim cnn
      Dim rst
      Dim sirSQL
      Dim sirConex
      Set cnn = Server.CreateObject("ADODB.Connection")
      sirConex = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=C:\cursasp\Northwind.mdb;"

      cnn.Mode = adModeReadWrite
      cnn.Open sirConex
      cnn.CursorLocation = adUseClient

      sirSQL="INSERT INTO Employees" & _
      "(LastName, FirstName, Address, City)" & _
      "VALUES ('Ion', 'Vasile', 'Crinului, nr. 12', 'Cluj')"
      cnn.Execute sirSQL, , adCmdText + adExecuteNoRecords

      sirSQL = "SELECT LastName, FirstName, Address, City FROM Employees"
      Set rst = cnn.Execute(sirSQL, , adCmdText)
%>
<P>
```

```
<TABLE BORDER = 2 BGCOLOR=YELLOW>
      <THEAD>
            <TH>  Nume angajat   </TH>
            <TH>   Adresa   </TH>
            <TH>   Oras   </TH>
      </THEAD>
<%    Do While Not rst.EOF %>
      <TR>
            <TD><%=rst("LastName")%>   <%=rst("FirstName")%></TD>
            <TD><%=rst("Address")%></TD>
            <TD><%=rst("City")%></TD>
      </TR>
<%
      rst.MoveNext
      Loop
%>

</TABLE>

<%
      rst.Close
      cnn.Close

      Set rst = Nothing
      Set cnn = Nothing
%>
</BODY>
</HTML>
```


Obiectul Command

Ierarhia ADO asigură o flexibilitate mare în reutilizarea obiectelor dincolo de unele limitări contextuale. Adică, în ADO se poate crea un singur obiect `Command`, dar acesta se poate folosi cu mai multe obiecte `Connection`. O facilitate a lui `Command` este colecția de obiecte `Parameters`. Aceasta se foloseşte pentru stocarea parametrilor specifici unei comenzi. Utilizarea principală a obiectului `Command` este aceea de execuție a interogărilor. Proprietățile şi metodele obiectului se prezintă în cele două tabele care urmează.

Proprietate	Descriere
`ActiveConnection`	Dă valorea obiectului `Connection` sau şirului de conexiune care va fi folosit de `Command`.
`CommandText`	Şir SQL sau numele procedurii stocate, a tabelului, sau a vederii care va fi sursa interogării.
`CommandType`	Descrie tipul comenzii din `CommandText` care urmează să se execute. Câteva dinte valorile folosite mai des sunt: • `adCmdText`, pentru şir SQL; • `adCmdTable`, pentru tabel al bazei de date; • `adCmdStoredProc`, pentru procedură stocată sau interogare de acțiune; • `adCmdUnknown`, (implicit) pentru tip necunoscut.
`CommandTimeout`	Numărul secundelor care se aşteaptă înainte de abandonarea execuției interogării (implicit, are valoarea 30).
`Prepared`	Dacă este `True`, valoarea din `CommandText` se va executa ca şi o comandă pregătită (compilată şi optimizată).

Proprietate	Descriere
State	Întoarce informații legate de starea închisă (adStateClose) sau deschisă (adStateOpen) a obiectului Command.

Metodă	Descriere
Cancel	Termină execuția asincronă a unei interogări executate prin Command.
CreateParameter	Creează un parametru pentru o interogare de acțiune parametrizată sau o procedură stocată.
Execute	Execută interogarea obiectului Command.

Codul din fişierul aspado3.asp creează o mulțime de înregistări (Recordset) cu ajutorul obiectului Command. Efectul lui este acelaşi cu cel al codului din fişierul aspado1.asp.

Fişierul aspado3.asp:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ADO si ASP</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<!-- #INCLUDE FILE =".\Program Files\Common Files\System\ado\adovbs.inc" -->

<H1>Baza de date: Northwind.mdb, Tabel: Employees</H1>
<%
      Dim cnn
      Dim cmd
      Dim rst
      Dim sirConex
      Set cnn = Server.CreateObject("ADODB.Connection")
      sirConex = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=C:\cursasp\Northwind.mdb;"

      cnn.Mode = adModeReadWrite
      cnn.Open sirConex
      cnn.CursorLocation = adUseClient

      Set cmd = Server.CreateObject("ADODB.Command")
      Set cmd.ActiveConnection = cnn
      cmd.CommandType = adCmdText
      cmd.CommandText = "SELECT * FROM Employees"
      Set rst = cmd.Execute

      Set cmd = Nothing
%>
<P>
<TABLE BORDER = 2 BGCOLOR=YELLOW>
      <THEAD>
            <TH>  Nume angajat   </TH>
            <TH>   Adresa   </TH>
            <TH>   Oras   </TH>
      </THEAD>
<%    Do While Not rst.EOF %>
      <TR>
            <TD><%=rst("LastName")%>   <%=rst("FirstName")%></TD>
```

```
            <TD><%=rst("Address")%></TD>
            <TD><%=rst("City")%></TD>
      </TR>
<%
      rst.MoveNext
      Loop
%>

</TABLE>

<%
      rst.Close
      cnn.Close

      Set rst = Nothing
      Set cnn = Nothing
%>
</BODY>
</HTML>
```

Metoda Command.Execute

Metoda `Execute` a obiectului `Connection` nu poate executa proceduri stocate care necesită parametri de intrare sau de ieşire, deoarece ea nu acceptă parametri. Metoda `Execute` a lui `Connection` poate executa o comandă SQL, iar rezultatele sunt depuse într-un obiect `Recordset`. Acest mod de execuție al comezilor SQL se numeşte SQL dinamic. Problema SQL-ului dinamic este viteza scăzută. Viteza de execuție a comenzilor SQL poate fi mărită prin pre-compilare. În acest caz comanda SQL se numește *procedură stocată*, deoarece ea va fi stocată în baza de date. SQL-ul dinamic este mai lent deoarece motorul bazei de date trebuie să analizeze comanda SQL, iar apoi, să genereze planul interogării. Planul interogării conține interogarea, structurile de tabelele şi indexurile în vederea determinării celei mai eficiente metode de a obține datele cerute. Atunci când se creează o procedură stocată, planul interogării este stocat şi el în baza de date.

Metoda `Execute` a obiectului `Command` ştie să lucreze cu parametri, ca urmare poate asigura o viteză mai mare de lucru. Problema procedurilor stocate este aceea că ele trebuie recompilate în cazul în care se modifică structura tabelelor sau indexurile care participă la procedura stocată. Fie comenzile SQL:

```
SELECT LastName, FirstName FROM Employees WHERE EmployeeID=1;
SELECT LastName, FirstName FROM Employees WHERE EmployeeID=7;
```

Acestea sunt două şiruri SQL care pot fi folosite pentru extragerea din tabelul **Employees** a doi angajați, primul are în câmpul EmployeeID valoarea 1, iar ar doilea are, în acelaşi câmp, valorea 7. Putem privi aceste şiruri sub forma unei proceduri care are un parametru de intrare, anume valoarea lui EmployeeID, prin care diferă cele două comenzi SQL. În Microsoft Access se va crea o interogare (aceasta este denumirea folosită de Microsoft Access pentru procedurile stocate) cu următorul cod SQL:

```
PARAMETERS ID Long;
SELECT LastName, FirstName, Address, City
FROM Employees
WHERE EmployeeID=ID;
```

Pentru aceasta deschideți baza de date *Northwind.mdb*, închideți ferestrele (Northwind TRADERS şi Main Switchboard) afişate la pornirea aplicației, apoi în fereastra bazei de date selectați **Queries**. Apăsați butonul **New**, apoi selectați opțiunea **DesignView**. Apăsați tasta **Esc**, apoi din **View** selectați **SQL View**. Aici scrieți în locul lui `SELECT;` comanda SQL anterioară. Închideți fereastra interogării, iar atunci când Access întreabă dacă doriți salvarea interogării

selectați **Yes**. În fereastra **Save As**, la **Query Name**, dați numele de `QueryExtrageAngajatID` procedurii stocate. Fişierul `aspado4.asp` prezintă modul în care se poate executa o pocedură stocată într-o bază de date Access.

Fişierul `aspado4.asp`:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ADO si ASP</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<!-- #INCLUDE FILE ="..\Program Files\Common Files\System\ado\adovbs.inc" -->

<H1>Baza de date: Northwind.mdb, Tabel: Employees</H1>
<%
      Dim cnn
      Dim cmd
      Dim rst
      Dim sirConex
      Dim par

      Set cnn = Server.CreateObject("ADODB.Connection")
      sirConex = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=C:\cursasp\Northwind.mdb;"

      cnn.Mode = adModeRead
      cnn.Open sirConex
      cnn.CursorLocation = adUseClient

      Set cmd = Server.CreateObject("ADODB.Command")
      Set cmd.ActiveConnection = cnn
      Set par = cmd.CreateParameter("EmployeeID",adInteger,adParamInput,4,1)
      cmd.Parameters.Append par
      cmd.CommandType = adCmdStoredProc
      cmd.CommandText = "QueryExtrageAngajatID"
      Set rst = cmd.Execute

      If Not rst.EOF Then
        Response.write "<P>" & rst("EmployeeID") & " -> "
        Response.write rst("LastName") & " " & rst("FirstName") & "<BR>"
      End If

      Set rst = Nothing
      Set cmd = Nothing
      Set cnn = Nothing
%>
</BODY>
</HTML>
```

În codul prezentat, metoda `CreateParameter` se foloseşte pentru crearea unui parametru. Dacă procedura stocată are mai mulți parametri, fiecare parametru va avea propriul lui `CreateParameter`. Presupunând că `cmd` este obiect `Command`, forma generală a metodei este:

```
cmd.CreateParameter(Nume, Tip, Sens, Mărime, Valoare)
```

În exemplul prezentat, `Nume`, care este numele parametrului, are valoarea "`EmployeeID`". `Tip` este o valoare de constantă predefinită care trebuie să fie aceeaşi cu tipul corespunzător câmpului `EmployeeID` din baza de date. Câteva dintre numele constantelor permise pentru `Tip` sunt prezentate în tabelul care urmează:

Constantă	Descriere
adBigInt	Câmp întreg, stocat pe 8 octeți.
adBinary	Câmp de lungime fixată care acceptă date binare.
adBoolean	Câmp de tip boolean.
adChar	Câmp şir, cu lungime fizată, de cel mult 255 de caractere.
adDate	Câmp dată (8 octeți)
adDecimal	Câmp de valoare numerică reală, reprezentat în virgulă fixă.
adDouble	Câmp de valoare numerică reală în dublă precizie, reprezentat în virgulă flotantă pe 8 octeți.
adInteger	Câmp întreg stocat pe 4 octeți.
adSingle	Câmp de valoare numerică reală în simplă precizie, reprezentat în virgulă flotantă.
adSmallInt	Câmp întreg pe 2 octeți.
adTinyInt	Câmp întreg pe 1 octet.
adVarChar	Câmp şir cu lungime variabilă, de până la 255 de caractere.

`Sens` poate lua una dintre cele cinci valori constante:

Constantă	Descriere
adParamUnknown	Nu se foloseşte.
adParamInput	Parametrul este de intrare. Aceasta este valoarea implicită.
adParamOutput	Parametrul este de ieşire. Adică, valoarea este generată de SQL server şi poate fi citită după treminarea procedurii stocate.
adParamInputOutput	Parametrul poate fi folosit pentru citirea şi scrierea unei valori dintr-o procedură stocată.
adParamReturnValue	Procedurile stocate pot întoarce o singură valoare întreagă. Ea poate fi obținută printr-un parametru de revenire (`return parameter`).

Metoda `Append` se foloseşte pentru adăugarea unui nou obiect `Parameter` colecției `Parameters` ale obiectului `Command`. În cazul în care există mai mulți parametri, ei trebuie adăugați cu `Append` în ordinea în care ei apar în definiția procedurii stocate.

Metoda `Execute` a obiectului `Command` are trei parametri opționali. Presupunând că `cmd` este un obiect `Command`, sintaxa este:

```
cmd.Execute [nr_înreg_afectate], [parametri], [opțiuni]
```

`nr_înreg_afectate` este o variabilă în care ADO întoarce numărul înregistrărilor afectate de

procedura stocată, `parametri` este un tablou, opțional, de parametri de intrare, iar `opțiuni` are aceeaşi semnificație ca şi cea din metoda `Execute` a obiectului `Connection`.

Crearea unei aplicaţii ASP ce interacţionează, prin ADO, cu o bază de date Microsoft Access

Aplicația care urmează asigură gestionarea unui grup de proiecte stocate într-o baza de date Microsoft Access prin Internet. O persoană poate să facă o propunere de proiect dacă se poate loga la baza de date. Logarea este posibilă numai dacă utilizatorul este înscris în baza de date. Un proiect are un director şi cel mult trei membri. Proiectul poate fi modificat de către director după stocarea lui în baza de date. Membri unui proiect pot vizualiza caracteristicile lui, dar nu pot să-i aducă modificări.

Crearea bazei de date Microsoft Access

Baza de date care trebuie creată în Microsoft Access, se numeşte *Pr.mdb* şi are cel trei tabele (**Membri**, **Persoane** şi **Proiecte**) din **Figura 66**.

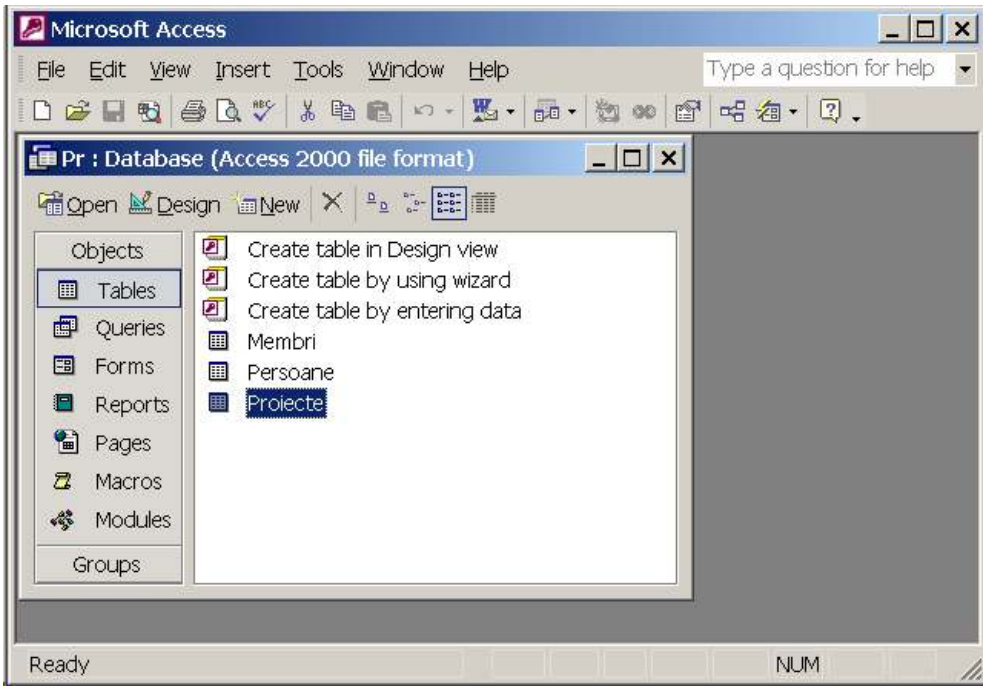


Figure 66
Cele trei tabele ale bazei de date *Pr.mbd*

Pentru crearea unui tabel, având selectată din coloana stângă opțiunea **Tables**, se face clic pe butonul **New**. Apoi, din fereastra **New Table** se alege opțiunea **Design View**. Aici, în coloana din stânga se vor completa numele câmpurilor corespunzătoare tabelului, apoi în coloana din dreapta fiecărui nume de câmp se va selecta tipul datei ce se stochează în câmp. Atunci când se face clic în dreapta unui nume de câmp, în partea dreaptă a coloanei **Data Type** se va afişa o săgeată în jos. Facând clic pe aceasta se va derula lista tuturor tipurilor de date disponibile în Microsoft Access. Dintre aceasta se vor alege tipurile conform celor arătate în **Figura 67**, **Figura 68** şi în **Figura 69**. În cazul lui **Membri** cheia primară este formată din două câmpuri. Pentru aceasta se ține butonul **Shift** apăsat, apoi se vor selecta liniile corespunzătoare câmpurilor `IDProiect` şi `IDPersoana`, în continuare din **Edit** selectați **Primary Key**. După crearea celor trei tabele se trece la crearea relațiilor dintre acestea. Pentru aceasta din **Tools** se va selecta **Relationships**..., se vor adăuga cele trei tabele în fereastra relațiilor, apoi se va trage cheia primară peste cheia străină. În final, fereastra relațiilor va fi asemenea celei din **Figura 70**.

Figure 67
Câmpurile, cu tipurile de date corespunzătoare, tabelului **Membri**

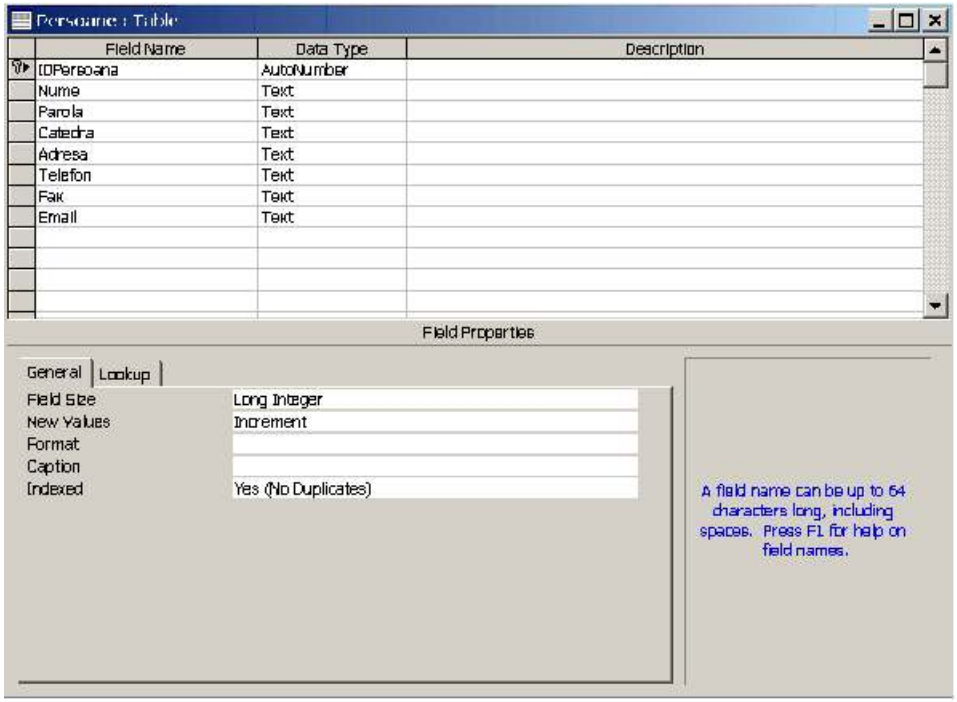


Figure 68
Câmpurile, cu tipurile de date corespunzătoare, tabelului **Persoane**

Figure 69
Câmpurile, cu tipurile de date corespunzătoare tabelului **Proiecte**

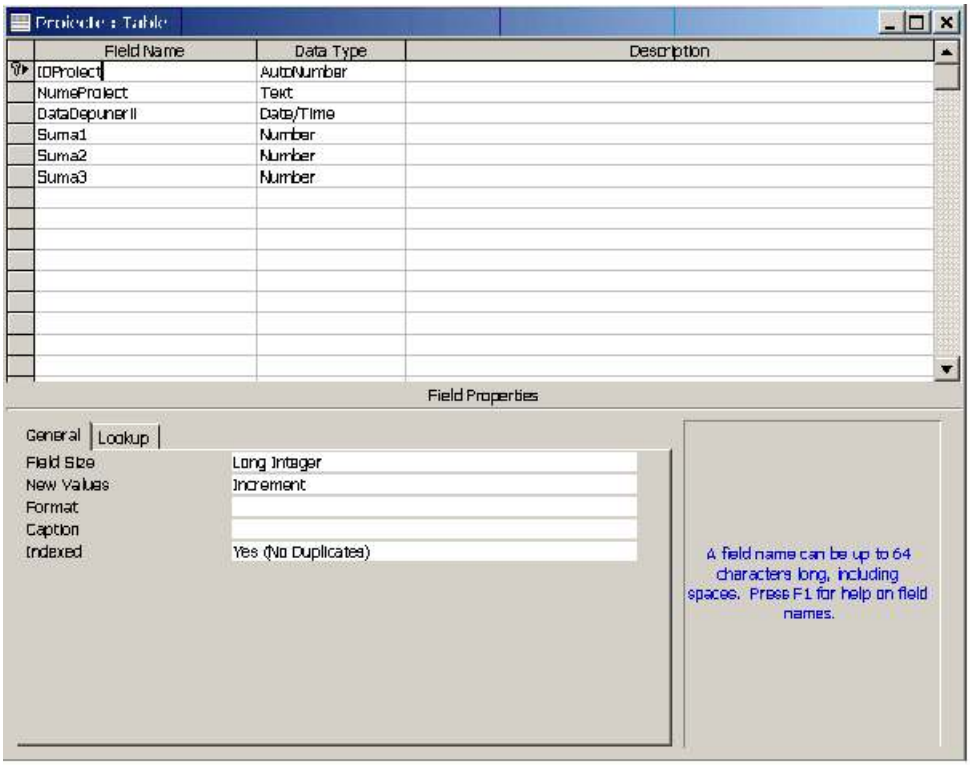
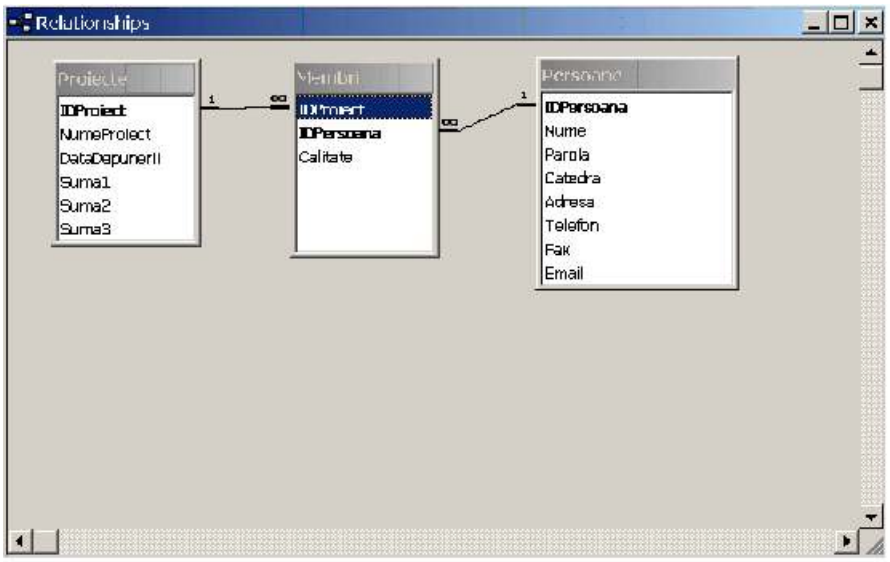


Figure 70
Fereastra relațiilor bazei de date *Pr.mdb*



Scopul aplicației este acela de a exemplifica modul în care se pot adăuga, modifica şi şterge înregistări ale unei baze de date Microsoft Access folosind ASP. Baza de date nu a trecut testele primelor trei forme normale. Chiar şi sub această formă ea poate fi utilizată fără prea multe probleme pe Web server.

Formularele HTML şi ASP ale aplicaţiei

Aplicația ASP conține următoarele fişiere:

- ☐ `logare.htm`: permite legarea utilizatorului la aplicație. Legarea este permisă numai dacă utilizatorul este deja membru al utilizatorilor bazei de date. Informațiile folosite pentru legare sunt numele şi parola utilizatorului. Acesta se definesc atunci când membrul, încă inexistent, se creează prin clic pe hiperlegătura Creare membru nou. (care va deschide fişierul `AdaugaPersoana1.asp`).
- ☐ `VerificaUtilizator.asp`: verifică dacă utilizatorul face parte dintre membrii ce pot utiliza baza de date. Dacă acesta există, atunci următoarea pagină care se deschide este `AfisareProiecte.asp`. Dacă numele şi/sau parola utilizatorului sunt greşite sau acesta nu este înscris în baza de date, se afişează un mesaj de eroare, după care utilizatorul este trimis din nou la pagina de legare.
- ☐ `AfisareProiecte.asp`: realizează afişarea proiectelor introduse în baza de date. Dacă utilizatorul este director sau membru în proiect, numele proiectului se afişează ca o hiperlegătură. Dacă se face clic pe aceasta se va deschide pagina `VizulizareProiect.asp` în care se pot vedea caracteristicile proiectului. Numele proiectelor în care utilizatorul curent nu ia parte sunt afişate fără posibilitatea de vizualizare sau de editare a detaliilor. Hiperlegătura Adauga un proiect nou. permite ca utilizatorul curent să adauge un proiect nou. El va fi automat, directorul noului proiect.
- ☐ `VizualizareProiect.asp`: realizează afişarea detaliilor specifice unui proiect. Acestea vor putea fi editate dacă utilizatorul este chiar directorul proiectului. În cazul în care utilizatorul este doar un membru din proiect, butoanele **Modificare**, **Stergere** şi **Reset** lipsesc. Hiperlegătura Selectare alt proiect. face revenirea la `AfisareProiecte.asp`.
- ☐ `AdaugaProiect1.asp`: permite adăugarea unui proiect nou. Numele directorului şi catedra din care face parte acesta sunt automat înscrise în formular şi nu vor putea fi modificate. Un proiect poate avea cel mult trei membri. Numele membrilor se vor selecta din cele trei liste. Dacă proiectul are mai puțin de trei membri varianta NU se foloseşte pentru marcarea neselecării membrului. La acest nivel, într-o aplicație reală, ar trebui verificate datele de intrare introduse. Din motive de spațiu, aici nu se mai fac aceste verificări. Pentru realizarea inserării proiectului fără erori este esențial ca data să fie scrisă sub forma lună/zi/an (de ex. 1/1/2003).
- ☐ `AdaugaProiect2.asp`: realizează adăugarea unui proiect nou în baza de date. Deoarece adăugarea necesită inserarea de valori în mai multe tabele, întregul grup de operații se declară tranzacție.
- ☐ `AdaugaPersoana1.asp`: permite adăugarea unui utilizator nou. Numai persoanele care fac parte din baza de date pot să vizualizeze, modifice sau să creeze proiecte. La acest nivel se face o verificare sumară a valorilor introduse, însă numai cu scopul de a arăta modul în care ar putea fi afişate unele nereguli.
- ☐ `AdaugaPersoana2.asp`: realizează adăugarea unui nou utilizator. Acesta este inserat în baza de date numai în cazul în care un utilizator cu acelaşi nume şi parolă nu există deja în baza de date.
- ☐ `ModificaProiect.asp`: permite modificarea unor caracteristici ale unui proiect. Modificarea poate fi realizată numai de către directorul de proiect.
- ☐ `ConstCon.inc`: constante specifice conexiunii la baza de date.
- ☐ `ConstAdo.inc`: constante specifice lui ADO.
- ☐ `Util.inc`: câteva proceduri specifice tuturor aplicațiilor.

Fişierul `logare.htm`:

Figure 71
Formularul logare.htm

```
<% Option Explicit %>
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM   NAME="Legare"   ONSUBMIT="return   Test()"   METHOD="POST"   ACTION
="http://localhost/CursASP/bd/VerificaUtilizator.asp">
<TABLE  BGCOLOR="AQUA"cellSpacing=3  cellPadding=3  width="40%"  align=center
border=2>
  <THEAD>
      <TH COLSPAN=2>Formular de intrare</TH> </TR>
  </THEAD>
  <TR>

    <TD>Nume:</TD>
    <TD><INPUT TYPE=TEXT NAME=Nume SIZE=50></TD>
  </TR>
  <TR>
    <TD>Parola:</TD>
    <TD><INPUT TYPE=PASSWORD NAME=Parola SIZE=10></TD>
  </TR>
  <TR>
    <TD><INPUT type=submit value=Trimite></TD>
    <TD ALIGN=RIGHT><INPUT type=reset value=Reset></TD>
  </TR>
  </TABLE>
  </P>
</FORM>
<TABLE width="40%" align=center>
<TR>
      <TD>
            <A HREF="http://localhost/CursASP/bd/AdaugaPersoana1.asp">Creare
membru nou.</A>
      </TD>
</TR>
</TABLE>

<SCRIPT LANGUAGE="VBScript">
      Function Test()
            Dim Frm
            Dim rasp
            rasp=True
            Set Frm = document.Legare
            If Len(Trim(Frm.Nume.Value))= 0 Then
                  alert("Nume nu poate fi vid")
                  rasp=False
```

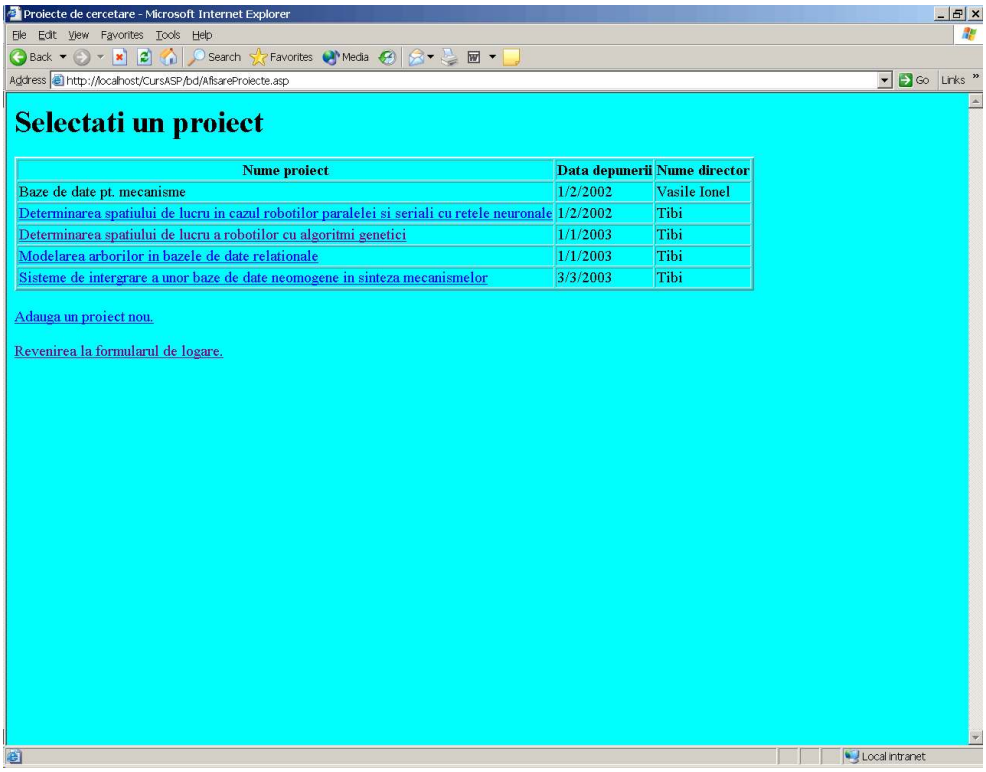
```
            ElseIf Len(Trim(Frm.Parola.Value))= 0 Then
                  alert("Parola nu poate fi vida")
                  rasp=False
            End If
            Test=rasp
      End Function
</SCRIPT>
</BODY>
</HTML>
```

Fişierul `VerificaUtilizator.asp`:

```
<%@ Language=VBScript %>
<!--#include file="ConstADO.inc"-->
<!--#include file="Util.inc"-->
<!--#include file="ConstCon.inc"-->

<BODY BGCOLOR="AQUA">

<%
      Function ExistaUtilizator(nume,parola)
            Dim cnn
            Dim rst
            Dim sirSQL

            Set cnn = Server.CreateObject("ADODB.Connection")
            Set rst = Server.CreateObject("ADODB.Recordset")
                        http://www.east.utcluj.ro/mb/mep/antal
            sirSQL = "SELECT * FROM Persoane" & _
              " WHERE Nume = '" & nume & "'" &
              " AND Parola = '" & parola & "'"
                        Universitatea Tehnica din Cluj-Napoca
                        Catedra Mecanica si Programare
            cnn.Open sirCnn  Conf. Dr. Ing. ANTAL Tiberiu Alexandru
            rst.Open sirSQL, cnn, adOpenKeyset, adLockReadOnly, adCmdText
            If Not rst.EOF Then
                  ExistaUtilizator = True
            Else
                  ExistaUtilizator = False
            End If

            rst.Close
            cnn.Close

            Set rst = Nothing
            Set cnn = Nothing
      End Function
%>

<%
      Dim nume
      Dim parola
      nume=Trim(Request.Form("Nume"))
      parola=Trim(Request.Form("Parola"))

      If ExistaUtilizator(nume,parola) Then
            Session("nume")=nume
            Session("parola")=parola
            Server.Transfer "AfisareProiecte.asp"
      Else
            Response.Write "Utilizator inexistent sau parola gresita!"
            Server.Transfer "logare.htm"
      End If
%>

</BODY>
</HTML>
```

Fişierul `AfisareProiecte.asp`:

Figure 72 Formularul AfisareProiecte.asp



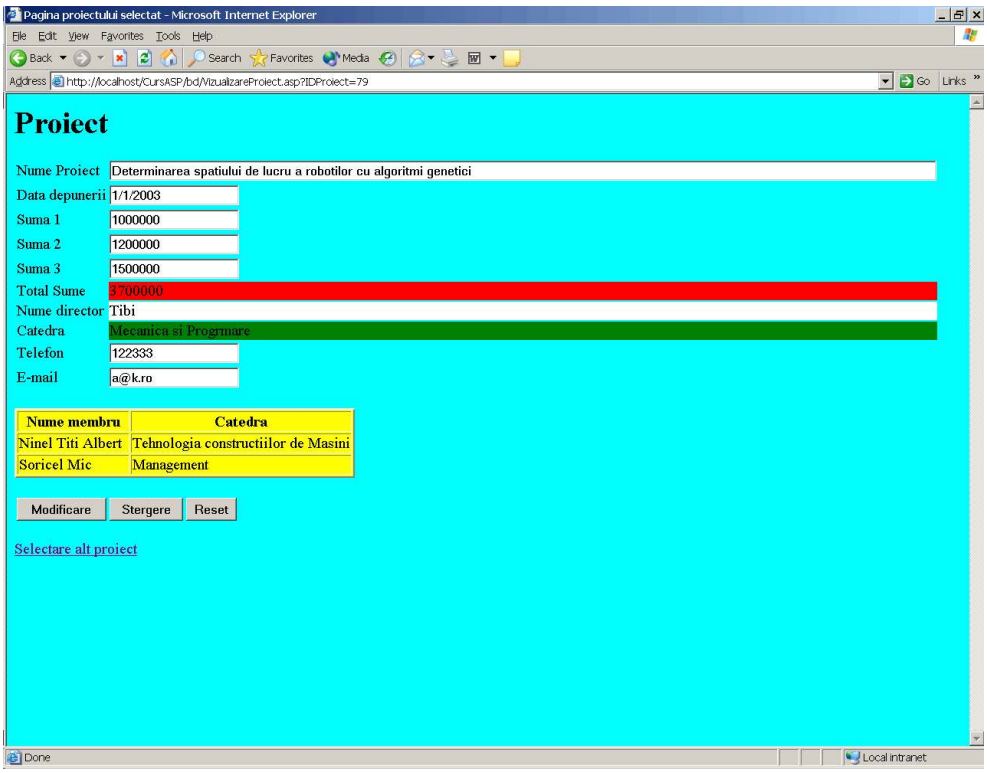

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>Proiecte de cercetare</TITLE>
</HEAD>

<BODY BGCOLOR="AQUA">
<!--#include file="ConstADO.inc"-->
<!--#include file="ConstCon.inc"-->

<H1>Selectati un proiect</H1>

<%
Function ExtrageCalitate(IDProiect, Nume)
      Dim cnn
      Dim rst
      Dim sirSQL

      Set cnn = Server.CreateObject("ADODB.Connection")
      Set rst = Server.CreateObject("ADODB.Recordset")

      cnn.Open sirCnn

      sirSQL = "SELECT Proiecte.[NumeProiect], Proiecte.[DataDepunerii],"
      sirSQL= sirSQL + " Persoane.Nume, Membri.Calitate, Proiecte.IDProiect"
      sirSQL= sirSQL + " FROM Proiecte INNER JOIN "
      sirSQL= sirSQL + "(Persoane INNER JOIN Membri ON Persoane.IDPersoana ="
      sirSQL= sirSQL + " Membri.IDPersoana) ON"
      sirSQL= sirSQL + " Proiecte.IDProiect = Membri.IDProiect"
      sirSQL= sirSQL + " WHERE Proiecte.IDProiect = " & CStr(IDProiect)
      sirSQL= sirSQL + " AND Nume = '" & CStr(Nume) & "'"
```

```
      rst.Open sirSQL, cnn, adOpenDynamic , adLockReadOnly, adCmdText

      If rst.EOF Then
            ExtrageCalitate = False
      Else
            ExtrageCalitate = True
      End If

      rst.close

      Set rst = Nothing
      Set cnn = Nothing
End Function
%>

<%

Dim cnn
Dim rst
Dim sirSQL
Dim nume

Set cnn = Server.CreateObject("ADODB.Connection")
Set rst = Server.CreateObject("ADODB.Recordset")

cnn.Open sirCnn

sirSQL = "SELECT Proiecte.[NumeProiect], Proiecte.[DataDepunerii],"
sirSQL= sirSQL + " Persoane.Nume, Membri.Calitate, Proiecte.IDProiect FROM"
sirSQL= sirSQL + " Proiecte INNER JOIN (Persoane INNER JOIN"
sirSQL= sirSQL + " Membri ON Persoane.IDPersoana = Membri.IDPersoana) ON"
sirSQL= sirSQL + " Proiecte.IDProiect = Membri.IDProiect"
sirSQL= sirSQL + " WHERE (((Membri.Calitate)=Yes));"

rst.Open sirSQL, cnn, adOpenForwardOnly, adLockReadOnly, adCmdText

If rst.EOF Then
      Response.Write "Nu exista proiecte introduse in baza de date."
Else
      nume=Session("nume")
%>
<TABLE BORDER=2>
      <TR>
            <TH>   Nume proiect   </TH>
            <TH>   Data depunerii </TH>
            <TH>   Nume director </TH>
      </TR>
<%
      Do While Not rst.EOF
%>
      <TR>
        <TD>
        <% If ExtrageCalitate(rst("IDProiect"),nume) Then %>
       <A HREF="VizualizareProiect.asp?IDProiect=<%=rst(4)%>"><%=rst(0)%></A>
            <%Else%>
                  <%=rst(0)%>
            <%End If%>
        </TD>
        <TD><%=rst(1)%></TD>
        <TD><%=rst(2)%></TD>
      </TR>
<%
            rst.MoveNext
      Loop
%>
</TABLE>
<%
End If
```

```
Set rst = Nothing
Set cnn = Nothing
%>

<P>
<A HREF="AdaugaProiect1.asp">Adauga un proiect nou.</A>
<P>
<A HREF="logare.htm">Revenirea la formularul de logare.</A>
</BODY>
</HTML>
```

Fişierul `VizualizareProiect.asp`:

Figure 73 Formularul VizualizareProiect.asp




```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>Pagina proiectului selectat</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<!--#include file="ConstADO.inc"-->
<!--#include file="ConstCon.inc"-->

<%
Function EsteDirector(numeproiect)
      Dim cnn
      Dim rst
      Dim sirSQL

      Set cnn = Server.CreateObject("ADODB.Connection")
      Set rst = Server.CreateObject("ADODB.Recordset")


      sirSQL ="SELECT Persoane.Nume, Persoane.Parola, Membri.Calitate, & " _
      " Proiecte.NumeProiect FROM Proiecte INNER JOIN (Persoane INNER" & _ "
      "JOIN Membri ON Persoane.IDPersoana = Membri.IDPersoana) ON " & _
```

```
      " Proiecte.IDProiect = Membri.IDProiect WHERE Membri.Calitate=Yes" & _
      " AND Proiecte.NumeProiect= '" & numeproiect & "'" & _
      " AND Persoane.Nume = '" & CStr(Session("nume")) & "'" & _
      " AND Persoane.Parola = '" & CStr(Session("parola")) & "';"

      cnn.Open sirCnn
      rst.Open sirSQL, cnn, adOpenKeyset, adLockReadOnly, adCmdText

      If rst.EOF Then
            EsteDirector=False
      Else
            EsteDirector=True
      End If

      rst.Close
      cnn.Close
      Set rst = Nothing
      Set cnn = Nothing

End Function
%>

<H1>Proiect</H1>
<%

Dim cnn
Dim rst
Dim sirSQL
Dim NumePr
```



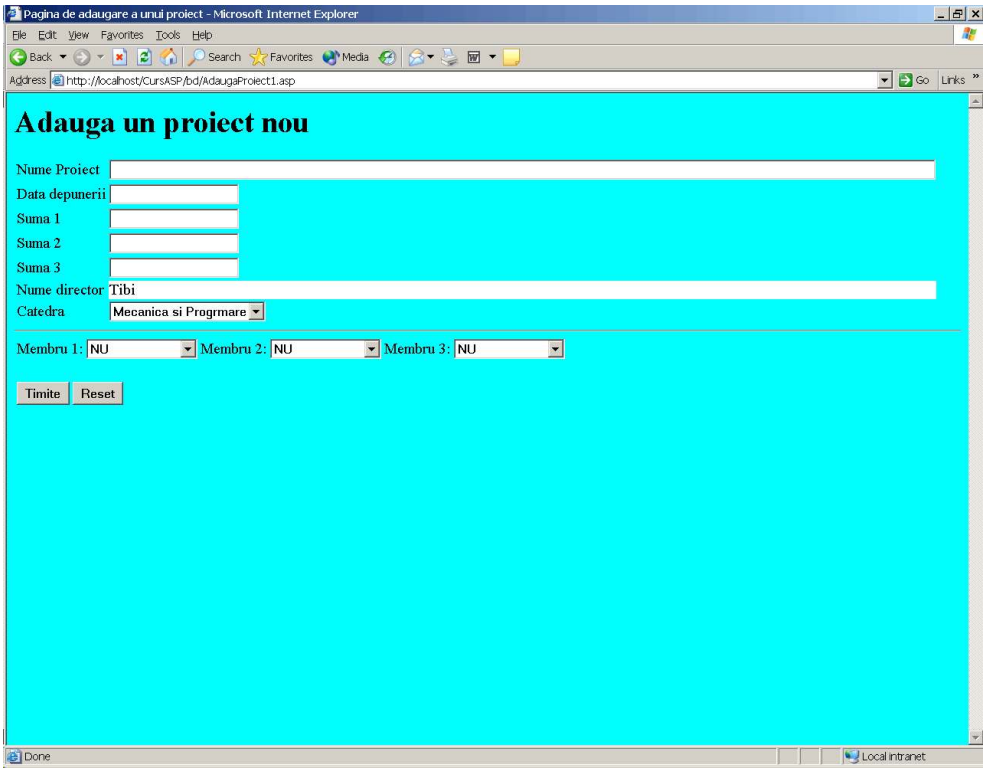
```
Set cnn = Server.CreateObject("ADODB.Connection")
Set rst = Server.CreateObject("ADODB.Recordset")

If IsEmpty(Request.QueryString("IDProiect")) Then
      Response.Write "Pagina nu a fost deschisa corect" & _
       "<A HREF=""logare.htm"">logare.asp</A>."
      Response.End
End If

sirSQL = "SELECT Proiecte.[NumeProiect], Proiecte.[DataDepunerii]," & _
" Proiecte.Suma1, Proiecte.Suma2, Proiecte.Suma3, Proiecte.IDProiect, " & _
" Membri.Calitate, Persoane.Nume, Persoane.Catedra, Persoane.Adresa," & _
" Persoane.Telefon, Persoane.Email FROM Proiecte INNER JOIN " & _
" (Persoane INNER JOIN Membri ON Persoane.IDPersoana = " & _
" Membri.IDPersoana) ON Proiecte.IDProiect = Membri.IDProiect WHERE" & _
" (Membri.Calitate=Yes) AND " & _
" ((Proiecte.IDProiect) = " & CStr(Request.QueryString("IDProiect")) &");"

cnn.Open sirCnn
rst.Open sirSQL, cnn, adOpenKeyset, adLockReadOnly, adCmdText

If rst.EOF Then
      Response.Write "Nu am gasit proiectul."
      Response.End
End If
NumePr = rst("NumeProiect")
%>

<FORM NAME ="FDirector" METHOD="Post" ACTION="ModificaProiect.asp">
      <INPUT  Name="IDProiect"  Type=Hidden  VALUE="<%=rst("IDProiect")%>"
SIZE=5>
<TABLE>
      <TR>
            <TD>Nume Proiect</TD>
            <TD><INPUT Name="NumeProiect"
            VALUE="<%=rst("NumeProiect")%>" SIZE=150</TD>
      </TR>
      <TR>
```

```
            <TD>Data depunerii</TD>
            <TD><INPUT Name="DataDepunerii"
            VALUE="<%=rst("DataDepunerii")%>" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 1</TD>
            <TD><INPUT Name="Suma1"
            VALUE="<%=rst("Suma1")%>" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 2</TD>
            <TD><INPUT Name="Suma2"
            VALUE="<%=rst("Suma2")%>" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 3</TD>
            <TD><INPUT Name="Suma3"
            VALUE="<%=rst("Suma3")%>" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Total Sume</TD>
            <% Dim Total
                  Total = rst("Suma1") + rst("Suma2") +rst("Suma3")
            %>
            <TD BGCOLOR=RED><%=Total%></TD>
      </TR>
      <TR>
            <TD>Nume director</TD>
            <TD BGCOLOR=WHITE><%=rst("Nume")%></TD>
      </TR>
      <TR>
            <TD>Catedra</TD>
            <TD BGCOLOR=GREEN><%=rst("Catedra")%></TD>
      </TR>
      <TR>
            <TD>Telefon</TD>
            <TD><INPUT Name="Telefon"
            VALUE="<%=rst("Telefon")%>" SIZE=20></TD>
      </TR>
      <TR>
            <TD>E-mail</TD>
            <TD><INPUT Name="Email"
            VALUE="<%=rst("Email")%>" SIZE=20></TD>
      </TR>
</TABLE>

<%
Set rst = Nothing
Set cnn = Nothing

Set cnn = Server.CreateObject("ADODB.Connection")
Set rst = Server.CreateObject("ADODB.Recordset")

sirSQL = "SELECT Proiecte.IDProiect, Persoane.Nume, Persoane.Catedra," & _
" Membri.Calitate FROM Proiecte INNER JOIN (Persoane INNER JOIN Membri" & _
" ON Persoane.IDPersoana = Membri.IDPersoana) ON Proiecte.IDProiect =" & _
" Membri.IDProiect WHERE (((Proiecte.IDProiect)= " & _
CStr(Request.QueryString("IDProiect")) & ") AND ((Membri.Calitate)=No));"

cnn.Open sirCnn
rst.Open sirSQL, cnn, adOpenKeyset, adLockReadOnly, adCmdText
%>
<P>
<TABLE BORDER = 2 BGCOLOR=YELLOW>
      <THEAD>
            <TH>  Nume membru   </TH>
            <TH>   Catedra   </TH>
      </THEAD>
```

```
<% Do While Not rst.EOF %>
      <TR>
            <TD><%=rst(1)%></TD>
            <TD><%=rst(2)%></TD>
      </TR>
<%
rst.MoveNext
Loop

Set rst = Nothing
Set cnn = Nothing
%>
</TABLE>
<P>
<% If EsteDirector(NumePr) Then %>
      <TABLE>
            <TR>
                  <TD><INPUT name="cmdSubmit"
                  type=submit value="Modificare"></TD>
                  <TD><INPUT name="cmdSubmit"
                  type=submit value="Stergere"></TD>
                  <TD><INPUT name="cmdSubmit"
                  type=reset value="Reset"></TD>
            </TR>
      </TABLE>
<% End If %>
</FORM>

<A HREF="AfisareProiecte.asp">Selectare alt proiect.</A>
</BODY>
</HTML>
```


Fişierul `AdaugaProiect1.asp`:

Figure 74 Formularul AdaugaProiect1.asp

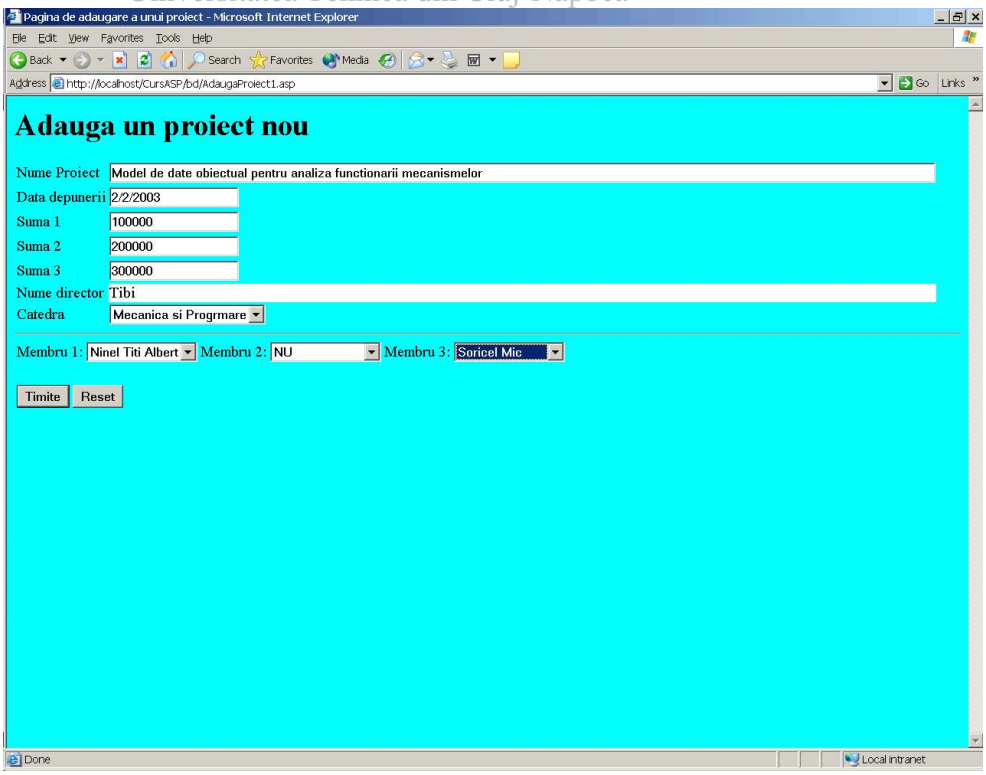



```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<!--#include file="ConstADO.inc"-->
```

```
<!--#include file="ConstCon.inc"-->

<HTML>
<HEAD>
<TITLE>Pagina de adaugare a unui proiect</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">

<%
      Dim cnn
      Dim rst
      Dim sirSQL
      Dim i

      Set cnn = Server.CreateObject("ADODB.Connection")
      Set rst = Server.CreateObject("ADODB.Recordset")

      cnn.Open sirCnn

      sirSQL= "SELECT Persoane.Catedra FROM Persoane " & _
      " WHERE Persoane.Nume='" & Session("nume") & _
      "' AND Persoane.Parola= '" & Session("parola") & "'"

      rst.Open sirSQL, cnn, adOpenDynamic , adLockReadOnly, adCmdText
%>

<H1>Adauga un proiect nou</H1>
<FORM NAME ="FAdauga" METHOD="Post" ACTION="AdaugaProiect2.asp">
<TABLE>
      <TR>
            <TD>Nume Proiect</TD>
            <TD><INPUT Name="NumeProiect" SIZE=150</TD>
      </TR>
      <TR>
            <TD>Data depunerii</TD>
            <TD><INPUT Name="DataDepunerii" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 1</TD>
            <TD><INPUT Name="Suma1" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 2</TD>
            <TD><INPUT Name="Suma2" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Suma 3</TD>
            <TD><INPUT Name="Suma3" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Nume director</TD>
            <TD BGCOLOR=WHITE> <%=Session("Nume")%></TD>
      </TR>
      <TR>
            <TD>Catedra</TD>
            <TD>
                  <SELECT Name="Catedra">
                        <% If rst.EOF Then%>
                              Nu exista catedre in baza de date
                        <% Else%>
                              <OPTION><%=Trim(rst(0))%></OPTION>
                        <% End If %>
                  </SELECT>
            </TD>
      </TR>

</TABLE>
<HR>
```

```
<TABLE NAME>
      <TR>
      <%For i = 1 To 3%>
            <TD>Membru <%=i%>:</TD>
            <TD>
                  <SELECT Name = <%="Membru" & CStr(i)%>>
                        <%
                              If i = 1 Then
                                    rst.Close
                                    sirSQL=  "SELECT  Persoane.Nume  FROM ✂
Persoane WHERE Nume <> '" & Session("nume") & "' ORDER BY Nume"
                                    rst.Open sirSQL, cnn, adOpenDynamic ,✂
adLockReadOnly, adCmdText
                              Else
                                    rst.MoveFirst
                              End If
                              If rst.EOF Then%>
                                    Nu exista membri in baza de date
                        <%    Else %>
                                    <OPTION>NU</OPTION>
                        <%          Do While Not rst.EOF
                        %>
                                    <OPTION><%=Trim(rst(0))%></OPTION>
                        <%
                                    rst.MoveNext
                                    Loop
                              End If
                        %>
                  </SELECT>
            </TD>
```


Figure 75 Formularul de adăugare a unui proiect nou completat

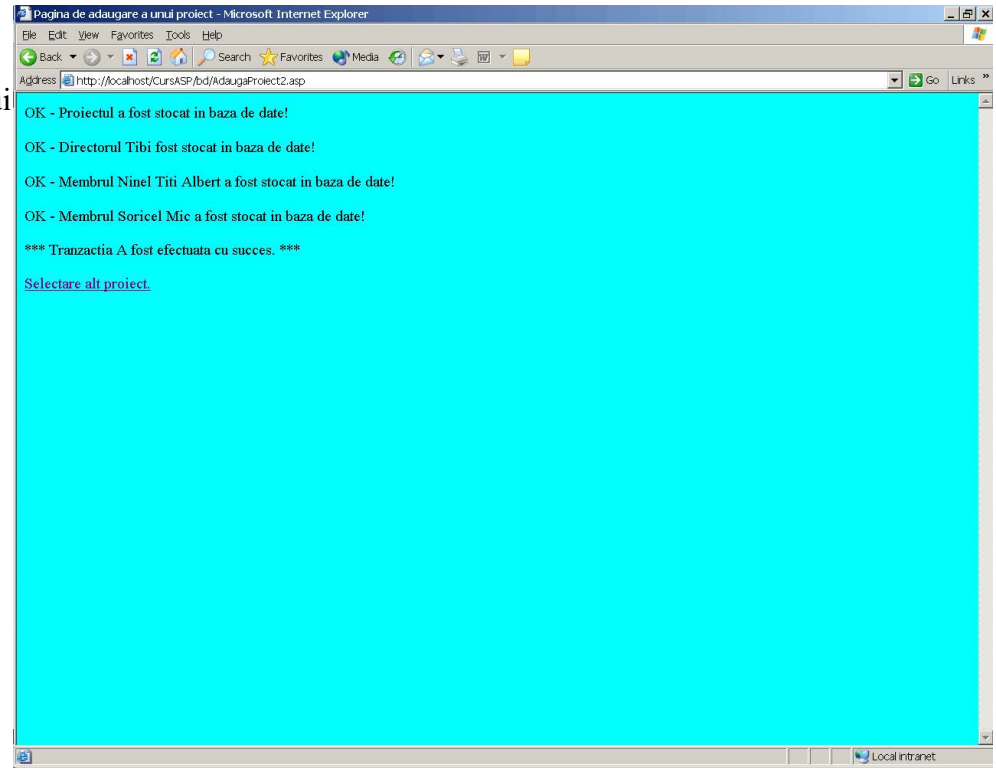



```
      <%
      Next
      Set rst = Nothing
      Set cnn = Nothing
      %>
      </TR>
</TABLE>
```

```
<P>
<TABLE>
      <TR>
            <TD><INPUT name="cmdSubmit" type=submit value="Timite"></TD>
            <TD><INPUT name="cmdSubmit" type=reset value="Reset"></TD>
      </TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Fişierul `AdaugaProiect2.asp`:

Figure 76 Mesajele afişate la adăugarea unui proiect



Porţiunea de cod care urmează este cea care înscrie date în mai multe tabele, motiv pentru care lucrează folosind tranzacţii. Prima oară, se adaugă datele specifice proiectului în tabelul **Proiecte**. Observând relaţiile bazei de date din **Figura 70**, cheia primară `IDProiect` a tabelului **Proiecte** este cheie străină în tabelul **Membri**, dar şi parte din cheia primară compusă a tabelului **Membri**. Aceasta înseamnă că după înscrierea unui proiect nou în tabelul **Proiecte**, valoarea dată de Access, automat, lui `IDProiect` (tipul lui `IDProiect` este **AutoNumber**; pentru acest tip Access generează şi stochează un număr cu valoare unică la fiecare inserare a unei înregistări noi în tabel în scopul identificării unice a fiecăriei înregistări) trebuie găsită şi folosită pentru stocarea noilor membri. Fiecare valoare este unică şi are proprietatea că este cea mai mare valoare dintre cele existente (Access generează valorile unice în ordine crescătoare). Deci, pentru a extrage valoarea lui `IDProiect` corespunzătoare ultimului proiect adăugat, se va căuta maximul din coloana `IDProiect`. Valoarea maximă se extrage cu ajutorul funcţiei `MAX` prin instrucţiunea `SELECT Max(IDProiect) AS MaxIDProiect FROM Proiecte`. Aceasta permite extragerea valorii maxime dintr-un grup de valori care vor fi tratate ca şi un întreg. Valoarea maximă a lui `IDProiect`, împreună cu valoarea lui `IDPersoana` corespunzătoare membrului selectat din lista membrilor stocaţi în baza de date, vor fi inseraţi în tabelul **Membri**.

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<!--#include file="ConstADO.inc"-->
<!--#include file="ConstCon.inc"-->
<!--#include file="Util.inc"-->
<HTML>
<HEAD>
<TITLE>Pagina de adaugare a unui proiect</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">
<%
      Function ExtrageID(Nume)
            Dim cnn
            Dim rst
            Dim sirSQL

            Set cnn = Server.CreateObject("ADODB.Connection")
            Set rst = Server.CreateObject("ADODB.Recordset")
            cnn.Open sirCnn
            sirSQL = "SELECT IDPersoana FROM Persoane "
            sirSQL = sirSQL & " WHERE Nume = '" & CStr(Nume) & "'"
            rst.Open sirSQL, cnn, adOpenDynamic , adLockReadOnly, adCmdText
            ExtrageID=rst(0)

            rst.Close
            cnn.Close
            Set rst = Nothing
            Set cnn = Nothing
      End Function
%>

<%
      Dim cnn
      Dim cmd
      Dim rst
      Dim sirSQL
      Dim inregMod
      Dim IDProiect
      Dim i
      Dim NumeMem

      Set cnn = Server.CreateObject("ADODB.Connection")
      cnn.Mode = adModeReadWrite
      cnn.Open sirCnn
      cnn.BeginTrans

      Set cmd = Server.CreateObject("ADODB.Command")
      Set cmd.ActiveConnection = cnn
      cmd.CommandType = adCmdText
      sirSQL = "INSERT INTO Proiecte (NumeProiect, DataDepunerii, Suma1,"
      sirSQL = sirSQL & " Suma2, Suma3) VALUES "
      sirSQL = sirSQL & "('" & Z2Null(Request.Form("NumeProiect")) & "',#"
      sirSQL = sirSQL &  Z2Null(Request.Form("DataDepunerii")) & "#,"
      sirSQL = sirSQL & Z2Null(Request.Form("Suma1")) & ","
      sirSQL = sirSQL & Z2Null(Request.Form("Suma2")) & ","
      sirSQL = sirSQL & Z2Null(Request.Form("Suma3")) & ")"
      cmd.CommandText = sirSQL
      cmd.Execute inregMod

      If inregMod <> 0 Then
            Response.Write "<P>OK - Proiectul a fost stocat in baza de✂
date!<BR>"'
      Else
            Response.Write "<P><FONT COLOR=RED>Eroare - Proiectul NU a fost✂
stocat in baza de date!</FONT><BR>"
      End If

      Set rst = Server.CreateObject("ADODB.Recordset")
```

```
      sirSQL = "SELECT Max(IDProiect) AS MaxIDProiect FROM Proiecte"
      rst.Open sirSQL, cnn, adOpenDynamic , adLockReadOnly, adCmdText
      IDProiect=rst(0)
      rst.Close

      cmd.CommandType = adCmdText
      sirSQL = "INSERT INTO Membri (IDProiect, IDPersoana, Calitate)" & _
      " VALUES (" & IDProiect & "," & ExtrageID(Session("nume")) & ",True)"
      cmd.CommandText = sirSQL
      cmd.Execute inregMod

      If inregMod <> 0 Then
            Response.Write "<P>OK - Directorul " & Session("nume") & " fost✂
stocat in baza de date!<BR>"'
      Else
            Response.Write "<P><FONT COLOR=RED>Eroare - Directorul NU a fost✂
stocat in baza de date!</FONT><BR>"
      End If

      For i= 1 to 3
            NumeMem = Request.Form("Membru" & CStr(i))
            If NumeMem <> "NU"  Then
                  sirSQL = "INSERT INTO Membri (IDProiect, IDPersoana," & _
                  " Calitate) VALUES (" & _
                  IDProiect & "," & _
                  ExtrageID(Request.Form("Membru"&CStr(i))) & ",False)"
                  cmd.CommandText = sirSQL
                  cmd.Execute inregMod
                  If inregMod <> 0 Then
                        Response.Write "<P>OK - Membrul " & NumeMem & " a✂
 fost stocat in baza de date!<BR>"'
                  Else
                        Response.Write "<P><FONT COLOR=RED>Eroare - Membrul✂
 NU a fost stocat in baza de date!</FONT><BR>"
                  End If
            End If
      Next

      If cnn.Errors.Count = 0 Then
            cnn.CommitTrans
            Response.Write "<BR>*** Tranzactia A fost efectuata cu succes.✂
 ***<BR>"
      Else
            cnn.RollbackTrans
            Response.Write "<BR>??? Tranzactia NU a fost efectuata ✂
 cu succes. ???<BR>"
      End If

      Set cmd = Nothing
      Set cnn = Nothing
%>

<BR>
<A HREF="AfisareProiecte.asp">Selectare alt proiect.</A>
</BODY>
</HTML>
```

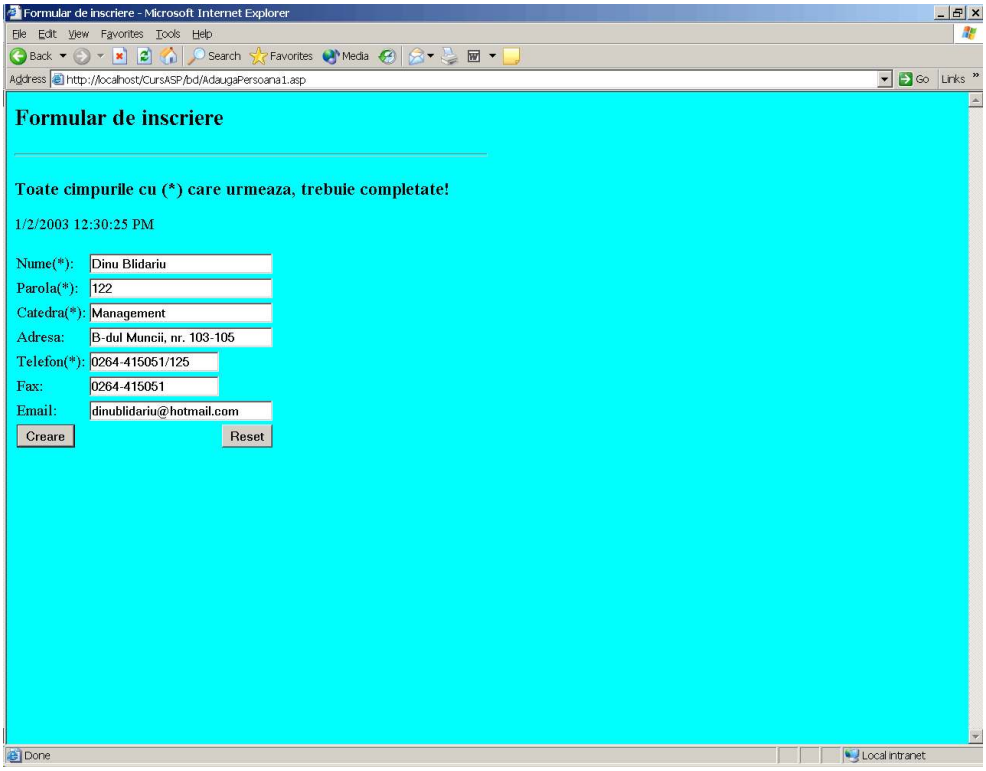


Figure 77 Formularul AdaugaPersoana1.asp

Fişierul `AdaugaPersoana1.asp`:

```
<%@ Language=VBScript %>
<HTML>
<HEAD>
<TITLE>Formular de inscriere</TITLE>
</HEAD>

<BODY BGCOLOR="AQUA">

<H2>Formular de inscriere</H2>

<HR SIZE=4 WIDTH="50%" ALIGN=LEFT>
<BR>

<H3>Toate campurile cu (*) care urmeaza, trebuie completate!</H3>

<% Response.write month(now) & "/" & day(now) & "/" & year(now) & " "✂
 & time() & "<BR>"      %>

<FORM NAME="Prs" METHOD="Post" ACTION="AdaugaPersoana2.asp"✂
 ONSUBMIT="return(Verifica());">
<TABLE>
      <TR>
            <TD>Nume(*):</TD>
            <TD><INPUT Name="Nume" SIZE=30></TD>
      </TR>
      <TR>
            <TD>Parola(*):  </TD>
            <TD><INPUT Name="Parola" SIZE=30></TD>
      </TR>
      <TR>
            <TD>Catedra(*):  </TD>
            <TD><INPUT Name="Catedra" SIZE=30></TD>
```

```
      </TR>
      <TR>
            <TD>Adresa:  </TD>
            <TD><INPUT Name="Adresa" SIZE=30></TD>
      </TR>
      <TR>
            <TD>Telefon(*):  </TD>
            <TD><INPUT Name="Telefon" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Fax:  </TD>
            <TD><INPUT Name="Fax" SIZE=20></TD>
      </TR>
      <TR>
            <TD>Email:  </TD>
            <TD><INPUT Name="Email" SIZE=30></TD>
      </TR>

      <TR>
            <TD><INPUT NEME="cmdSubmit" type=submit value="Creare"></TD>
            <TD ALIGN=RIGHT><INPUT name="cmdSubmit" type=submit ✂
 value="Reset"></TD>
      </TR>
</TABLE>
<A HREF
</FORM>
<FONT COLOR="RED" SIZE=5><P ID="Er"> </P></FONT>
<SCRIPT LANGUAGE="VBScript">
      Function Verifica()
            Dim err
            Dim frm
            Dim el

            Set frm = document.Prs
            err=False
            Prob="EROARE, urmatoarele campuri sunt vide: "
            If (Trim(frm.Nume.Value) = "") then
                  prob = prob & "'Nume' "
                  err=True
            End If
            If (Trim(frm.Parola.Value) = "") then
                  prob = prob & "'Parola' "
                  err=True
            End If
            If (Trim(frm.Catedra.Value) = "") then
                  prob = prob & "'Catedra' "
                  err=True
            End If
            If (Trim(frm.Telefon.Value) = "") then
                  prob = prob & "'Telefon' "
                  err=True
            End If

            If err Then
                  Set el = document.all("Er")
                  el.innertext=prob
                  Verifica=false
            Else
                  Verifica=true
            End If

      End Function
</SCRIPT>
</BODY>
</HTML>
```


Fişierul `AdaugaPersoana2.asp`:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<% Response.Buffer = True %>

<HTML>
<HEAD>
<TITLE>F2</TITLE>

</HEAD>

<BODY BGCOLOR="AQUA">
<!--#include file="ConstADO.inc"-->
<!--#include file="Util.inc"-->
<!--#include file="ConstCon.inc"-->

<H2>Formular de inscriere</H2>
<HR SIZE=4 WIDTH="50%" ALIGN=LEFT>
<BR>

<%
      Sub Exista
            Dim cnn
            Dim rst
            Dim sir

            Set cnn = Server.CreateObject("ADODB.Connection")
            Set rst = Server.CreateObject("ADODB.Recordset")

            sir = "SELECT * FROM Persoane" & _
              " WHERE Nume = '" & Request.Form("Nume") & "'" & _
              " AND Parola = '" & Request.Form("Parola") & "'"

            cnn.Open sirCnn
            rst.Open sir, cnn, adOpenKeyset, adLockReadOnly, adCmdText
            If Not rst.EOF Then
               Response.Write "<P>Sinteti deja inscris in baza de date!"
               Response.Write "<P>Pentru reusita inscrierii modificati " & _
              "continutul campurilor: <FONT COLOR=RED> Nume, Parola </FONT>!"
               Response.End
            End If

            rst.Close
            cnn.Close

            Set rst = Nothing
            Set cnn = Nothing
      End Sub
%>

<%
      Dim cnn
      Dim cmd
      Dim inregMod
      Dim sir
      Dim raspuns

      Exista

      Set cnn = Server.CreateObject("ADODB.Connection")
      cnn.Mode = adModeReadWrite
      Set cmd = Server.CreateObject("ADODB.Command")

      cnn.Open sirCnn
      Set cmd.ActiveConnection = cnn
      cmd.CommandType = adCmdText

      sir = "INSERT INTO Persoane (Nume, Parola, Catedra, Adresa, Telefon,"
      sir = sir & " Fax, Email) VALUES"
```

```
      sir = sir & "('" & Request.Form("Nume")
      sir = sir & "','"& Request.Form("Parola")
      sir = sir & "','"& Request.Form("Catedra")
      sir = sir & "','"& Request.Form("Adresa")
      sir = sir & "','"& Request.Form("Telefon")
      sir = sir & "','"& Request.Form("Fax")
      sir = sir & "','"& Request.Form("Email") & "')"

      cmd.CommandText = sir
      cmd.Execute inregMod

      If inregMod <> 0 Then
            Response.Write "<P>OK - Ati fost inregistrat in baza de✄
 date!<BR>"'
      Else
            Response.Write "<P><FONT COLOR=RED>Eroare - NU ati fost✄
 inregistrat in baza de date!</FONT><BR>"
      End If

      cnn.Close
      Set cmd = Nothing
      Set cnn = Nothing
%>

<FORM METHOD = "POST">
<INPUT TYPE="BUTTON" NAME="btnIesire" VALUE="Iesire">
<SCRIPT FOR="btnIesire" EVENT="onClick" LANGUAGE="VBScript">
      top.window.close
</SCRIPT>
</FORM>
<A HREF="http://localhost/CursASP/bd/logare.htm">Revenire la formularul de✄
 logare</A>

</BODY>
</HTML>
```


Fişierul `ModificaProiect.asp`:

```
<%@ LANGUAGE="VBScript" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>Modificare proiect</TITLE>
</HEAD>
<BODY BGCOLOR="AQUA">

<!--#include file="ConstADO.inc"-->
<!--#include file="Util.inc"-->
<!--#include file="ConstCon.inc"-->

<H1>Actualizare proiect</H1>
<%

Dim cnn
Dim cmd
Dim inregMod
Dim NumeProiect
Dim Data
Dim Suma1
Dim Suma2
Dim Suma3
Dim Telefon
Dim Email
Dim sirSQL


Set cnn = Server.CreateObject("ADODB.Connection")
```

```
Set cmd = Server.CreateObject("ADODB.Command")

Select Case Request.Form("cmdSubmit")
Case "Modificare"
      NumeProiect=Z2Null(Request.Form("NumeProiect"))
      Data=Z2Null(Request.Form("DataDepunerii"))
      Suma1=Z2Null(Request.Form("Suma1"))
      Suma2=Z2Null(Request.Form("Suma2"))
      Suma3=Z2Null(Request.Form("Suma3"))
      Telefon=Z2Null(Request.Form("Telefon"))
      Email=Z2Null(Request.Form("Email"))

      sirSQL = "UPDATE Proiecte INNER JOIN (Persoane INNER JOIN Membri" & _
      " ON Persoane.IDPersoana = Membri.IDPersoana) " & _
      " ON Proiecte.IDProiect = Membri.IDProiect" & _
      " SET Proiecte.[NumeProiect] = '" & NumeProiect & _
      "' , Proiecte.[DataDepunerii] = #" & _
       Data & "#, Proiecte.Suma1 = "& Suma1 & _
      " , Proiecte.Suma2 = " & Suma2 & ", Proiecte.Suma3 =  " & Suma3 & _
      " , Persoane.Telefon = '" & Telefon &"', Persoane.Email = '" & _
       Email & "' WHERE (Proiecte.IDProiect) = " & _
       Z2Null(Request.Form("IDProiect"))

      Set cnn = Server.CreateObject("ADODB.Connection")
      cnn.Mode = adModeReadWrite
      cnn.Open sirCnn

      Set cmd = Server.CreateObject("ADODB.Command")
      Set cmd.ActiveConnection = cnn
      cmd.CommandType = adCmdText
      cmd.CommandText = sirSQL
      cmd.Execute inregMod

      'Linia urmatoare este echivalenta
      'cu cele 5 linii anterioare
      'cnn.Execute sirSQL, inregMod , adCmdText

      If inregMod <> 0 Then
            Response.Write "Proiectul a fost actualizat!"
      Else
            Response.Write "Actualizarea nu se poate realiza."
      End If

Case "Stergere"
      Set cnn = Server.CreateObject("ADODB.Connection")
      cnn.Mode = 3
      Set cmd = Server.CreateObject("ADODB.Command")

      cnn.Open sirCnn
      Set cmd.ActiveConnection = cnn
      cmd.CommandType = adCmdText

      sirSQL = "DELETE FROM Proiecte " & _
      " WHERE Proiecte.IDProiect = " & Z2Null(Request.Form("IDProiect"))

      cmd.CommandText = sirSQL
      cmd.Execute inregMod

      If inregMod <> 0 Then
            Response.Write "Proiectul a fost sters!"
      Else
            Response.Write "Stergerea nu se poate realiza."
      End If

End Select

Set cmd = Nothing
Set cnn = Nothing
```

```
%>
<P>
<A HREF="AfisareProiecte.asp">Selectare alt proiect.</A>
<P>
</BODY>
</HTML>
```

Fişierul `ConstCon.inc`:

```
<%
Dim sirCnn

' Aici, probabil, se vor face modificari.
sirCnn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
"Data Source=C:\cursasp\bd\Pr.mdb;"
%>
```

Fişierul `ConstAdo.inc`:

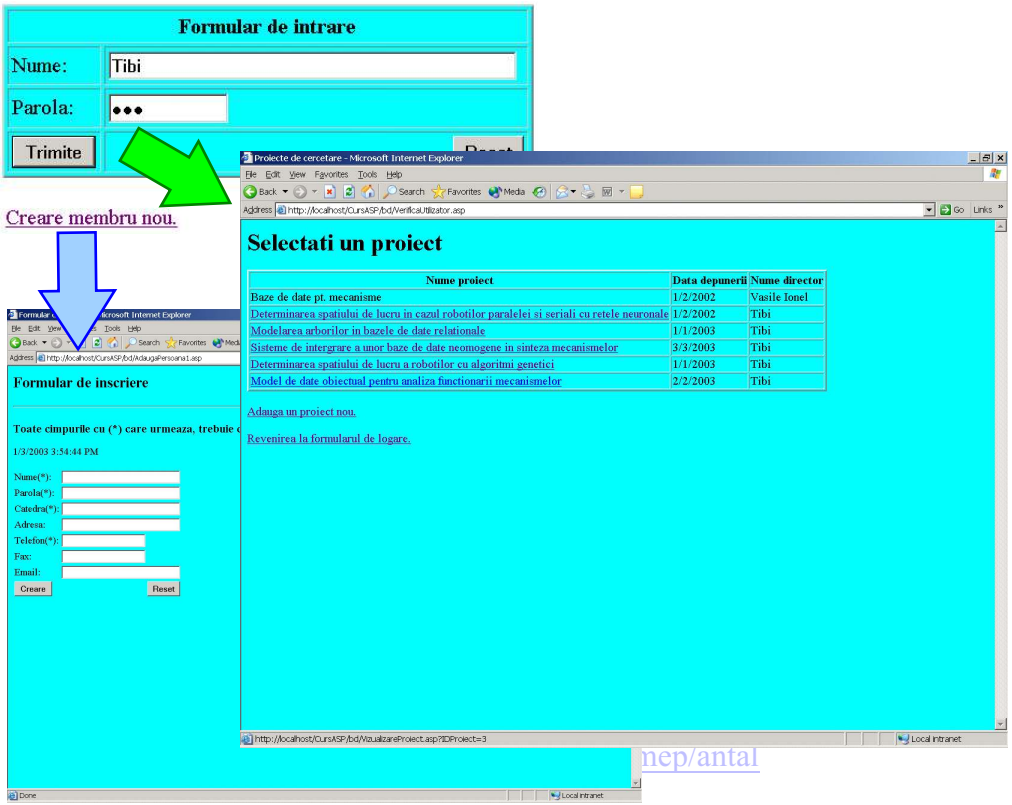
```
<%
'---- ConnectModeEnum Values ----
Const adModeUnknown = 0
Const adModeRead = 1
Const adModeWrite = 2
Const adModeReadWrite = 3
Const adModeShareDenyRead = 4
Const adModeShareDenyWrite = 8
Const adModeShareExclusive = &Hc
Const adModeShareDenyNone = &H10
Const adModeRecursive = &H400000

'---- CursorTypeEnum Values ----
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3

'---- CursorLocationEnum Values ----
Const adUseServer = 2
Const adUseClient = 3

'---- LockTypeEnum Values ----
Const adLockReadOnly = 1
Const adLockPessimistic = 2
Const adLockOptimistic = 3
Const adLockBatchOptimistic = 4

'---- CommandTypeEnum Values ----
Const adCmdUnknown = &H0008
Const adCmdText = &H0001
Const adCmdTable = &H0002
Const adCmdStoredProc = &H0004
Const adCmdFile = &H0100
Const adCmdTableDirect = &H0200

'---- ParameterDirectionEnum Values ----
Const adParamUnknown = &H0000
Const adParamInput = &H0001
Const adParamOutput = &H0002
Const adParamInputOutput = &H0003
Const adParamReturnValue = &H0004

'---- DataTypeEnum Values ----
Const adEmpty = 0
Const adTinyInt = 16
Const adSmallInt = 2
```

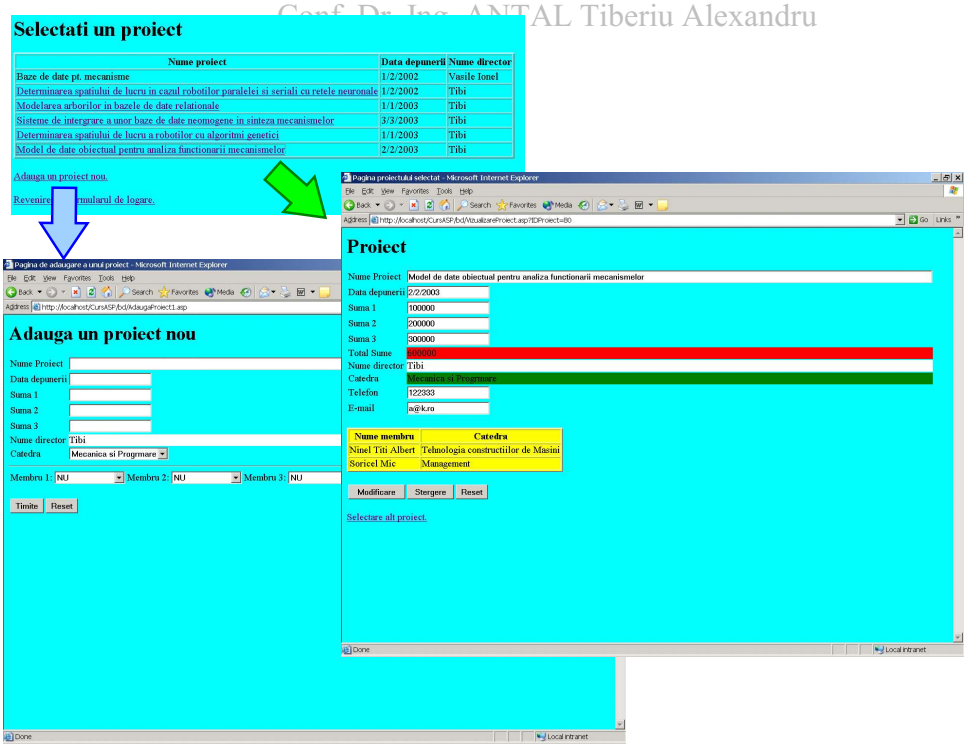
```
Const adInteger = 3
Const adBigInt = 20
Const adUnsignedTinyInt = 17
Const adUnsignedSmallInt = 18
Const adUnsignedInt = 19
Const adUnsignedBigInt = 21
Const adSingle = 4
Const adDouble = 5
Const adCurrency = 6
Const adDecimal = 14
Const adNumeric = 131
Const adBoolean = 11
Const adError = 10
Const adUserDefined = 132
Const adVariant = 12
Const adIDispatch = 9
Const adIUnknown = 13
Const adGUID = 72
Const adDate = 7
Const adDBDate = 133
Const adDBTime = 134
Const adDBTimeStamp = 135
Const adBSTR = 8
Const adChar = 129
Const adVarChar = 200
Const adLongVarChar = 201
Const adWChar = 130
Const adVarWChar = 202
Const adLongVarWChar = 203
Const adBinary = 128
Const adVarBinary = 204
Const adLongVarBinary = 205
Const adChapter = 136
Const adFileTime = 64
Const adDBFileTime = 137
Const adPropVariant = 138
Const adVarNumeric = 139

'---- PositionEnum Values ----
Const adPosUnknown = -1
Const adPosBOF = -2
Const adPosEOF = -3
%>
```


Fişierul `Util.inc`:

```
<%
Function Z2Null(v)
      ' Inlocuieste sirul vid cu
      ' Null.
      If Len(v)=0 Then
            Z2Null = Null
      Else
            Z2Null = v
      End If
End Function
%>
```

În figurile care urmează se prezintă modul în care se poate ajunge la anumite pagini ale aplicaţiei prin selectarea hiperlegăturilor sau a butoanelor. Atunci când baza de date este fără înregistrări, prima acţiune este aceea de introducere a utilizatorilor aplicaţiei. Numai dacă există deja utilizatori introduşi se vor putea crea noi proiecte.





INDEX

W

BIBLIOGRAFIE

[1] A. Russel Jones, Mastering Active Server Pages 3, SYBEX, 2000, ISBN: 0-77821-2619-7.

[2] Andrew S. Tanenbaum, Rețele de calculatoare, Ediția a treia, 1998, Computer Press AGORA, ISBN: 973-97706-3-0.

[3] Cary Prague, Michal Irwin, Access 2002 Bible, Hungy Minds, Inc. 2001, ISBN: 0-7645-3596-X.

[4] Antal Tiberiu Alexandru, Microsoft Access 97 şi 2000 în 14 cursuri. Ed. Todesco, 2000, ISBN: 973-99779-6-0.