WebWord Editor

– Marian Veţeanu

Nu cred că este utilizator Internet care să nu fi observat tendinţa generală a web-ului de îndreptare rapidă către multimedia. Vechile pagini statice, ce prezentau doar informaţiile într-o modalitate nu tocmai placută ochiului, sunt rapid înlocuite de noile pagini în care informaţia încearcă să pătrundă cât mai puternic în mintea utilizatorului prin folosirea a tot felul de efecte grafice şi de animaţie.

La o analiză atentă se poate observa că această modalitate de prezentarea a informaţiilor se regăseşte doar pe direcţia server-client şi nu şi invers. Dacă un vizitator al unui site sau un utilizator al unei aplicaţii Web poate primi de la server documente HTML cu formatări şi imagini spectaculoase nu acelaşi lucru se poate spune despre conţinutul pe care poate el să-l trimită către server. Singura modalitate standard de trimitere de informaţii către server constă în folosirea clasicelor formulare care nu prea au suferit modificări de la introducerea loc ca parte standard a limbajului HTML.

Acestea formulare se definesc în interiorul paginii cu ajutorul tagurilor: `FORM`, `INPUT`, `SELECT` şi `TEXTAREA` şi permit implementarea principalelor controale disponibile în orice mediu grafic (Windows, X-Windows, etc.) : câmpuri pentru introducere de text simplu şi text pe mai multe linii, checkbox-uri, radiobutton-e, combobox-uri, listbox-uri, câmpuri pentru introducerea de parole şi câmpuri pentru transferul de fişiere de la client la server (câmpuri *file*). În ultimii ani web-ul a evoluat mult, dar aceste controale şi-au păstrat aspectul şi comportarea iniţială, de exemplu un `TEXTAREA` poate fi folosit doar pentru introducerea de texte simple ca cele create cu programul NotePad, nici vorbă de aplicarea de comenzi de formatare asupra textului introdus.

Controalele clasice sunt suficiente pentru majoritatea siturilor web: situri comerciale cu formulare de comandă pentru produse, situri web cu formulare de login şi parolă, motoare de căutare, etc. dar sunt insuficiente pentru crearea de aplicaţii web multimedia. Este drept ca prin intermediul câmpurilor de tip *file* pot fi transferate către server şi alt tip de informaţii decât cel care se poate obţine folosind controalele standard, dar această modalitate de lucru implică existenţa de aplicaţii suplimentare folosite atât pentru crearea conţinutului cât şi pentru citirea acestuia.

Dezvoltatorii IE au realizat până la urmă acest handicap şi au introdus în browserele *Internet Explor 5.0* dar mai ales *5.5* posibilitatea editării online de HTML făcând astfel posibilă trimitea pe server a unui conţinut asemănător cu cel folosit în programele moderne. Faptul că această facilitate este prezentă doar în browserul IE îngreunează modul de implementare a ei pe Internet unde vizitatorii pot avea orice browser scris vreodată dar dă posibilitatea aplicaţiilor Intranet de a oferi mai multe funcţii utilizatorilor.

Întreg articolul se va concentra asupra creerii în interiorul unei pagini web a unui editor de fişiere HTML cu imagini şi salvarea documentelor create într-o bază de date. Utilizările practice ale unei asemenea aplicaţii sunt practic nelimitate:

- folosirea editorului în aplicaţii Intranet multimedia
- folosirea editorului pe situri de WebHosting pentru a oferii utilizatorilor posibilitatea de aş edita paginile vizual direct în browser
- folosirea editorului pe siturile cu grupuri de discuţii pentru a da posibilitatea membrilor de a posta articole HTML complexe cu imagini interioare
- etc.

Conţinutul HTML produs de acest editor va putea fi vizualizat de orice browser, doar cei care editează documentele având nevoie de *Internet Explorer*. Pentru realizarea părţii de server a editorului s-a optat pentru tehnologia ASP (*Active Server Pages*) alături de o bază de date *Access*.

Pe lângă realizare propriuzisă a editorului se vor puncta şi alte câteva aspecte ale web-ului precum: uploadarea fişierelor binare prin browser şi securitatea impusă de IE la astfel de upload-uri.

Aşa cum am amintit anterior realizarea acestui editor este posibilă datorită facilităţii lui IE5+ de editare online de HTML direct pe pagina web. Trecerea în browserului în modul de editare se realizează prin setarea atributului/proprietăţii `CONTENTEDITABLE`. Acest atribut disponibil la majoritatea elementelor HTML printre care: `A`, `SPAN`, `DIV`, `BUTTON`, `FONT`, etc. În vederea testării modului de funcţionare a acestui atribut vom crea o simplă pagină web cu următorul conţinut:

```
<span>Acest text nu poate fi modificat.</span><br>
<span contenteditable=true>Acest text POATE fi modificat</span>
```

Vom observa în cazul celui de-al doilea *span* posibilitatea de modificare a conţinutului acestuia prin folosirea tastaturii. Pagina editorului ce-l vom realiza va fi conţinută într-un astfel de *span*. Pentru ca editarea să se realizeze vizual vom înzestra editorul cu un *toolbar* de pe care se vor putea selecta comenzile clasice de editare: aplicare bold, italic, underline, liste numerotate, liste nesortate, etc. precum şi prin intermediul căruia vom insera imagini în documentul ce se editează.

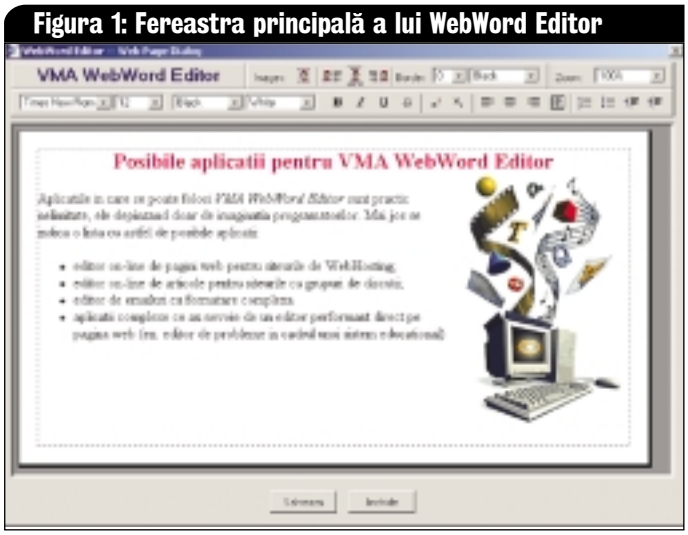
Trimiterea comenzilor de formatare, selectate de pe toolbar, către textul documentului o realizăm prin folosirea metodei *execCommand* a obiectului document. Sintaxa ei este următoarea:

```
bSuccess = object.execCommand(sCommand [, bUserInterface] [,
    vValue])
```

Parametrii primiţi de aceasta au semnificaţia următoare:

`sCommand` – string cu comanda ce urmează a se executa (parametru necesar)

Figura 1: Fereastra principală a lui WebWord Editor



`bUserInterface` – în cazul în care comanda are posibilitatea afișarii unei ferestre de dialog atunci aceasta se va afișa doar dacă parametrul are valoarea true (parametru opțional)

`vValue` – variant ce specifică valori suplimentare folosite de comanda specificată.

Listă completă de comenzi ce pot fi folosite de această metodă se poate găsi pe site-ul Microsoft *http://msdn.microsoft.com*. În cazul editorului nostru se vor folosi doar comenzile *Bold, Italic, Underline, StrikeThrough, SuperScript, SubScript, JustifyLeft, JustifyCenter, JustifyRight, InsertOrderedList, InsertUnorderedList, Outdent, Indent, FontName, FontSize, ForeColor, BackColor*, și *InsertImage*.

Modul în care va arăta acest editor este prezentat în *figura 1*.

Pentru ca acest editor să poată fi folosit și alte aplicații se va încerca o implementare generală a subrutinelor folosite pentru funcționarea editorului.

Primul pas în realizare constă în crearea interfeței cu utilizatorul (pagina ce se poate edita, toolbar-urile, etc). Marea parte a interfeței este creată pe server în ASP. În acest sens au fost create câteva subrutine generale prezentate pe scurt în tabel.

Subrutine genrale

Nume subrutină	Descriere
OpenHTMLEditControl (iLeft, iTop, iWidth, iHeight, HTCName, ImagesDir, strComponentName)	Aceasta este subrutina principală care deschide controlul de editare. Parametrii de intrare au următoarea semnificație: iLeft, iTop - sunt folosiți pentru poziționarea absolută controlului; dacă lipsesc atunci controlul este pus cu pozitie relativă ; iWidth - lungimea controlului ; iHeight - înălțimea controlului ; HTCName - calea către fișierul .htc ce conține partea activă de client ; ImagesDir - calea către directorul ce conține imaginile folosite la crearea toolbar-urilor strComponentName - numele componentei
OpenPage(iWidth, iHeight, strWaterMark)	Subrutină folosită pentru deschiderea unei noi pagini de editare în cadrul controlului. Dacă strWaterMark este diferit de șirul vid atunci el indică calea către o imagine ce va fi afișata pe post de watermark pe pagină.
ClosePage	Subrutină apelată pentru închiderea unei pagini deschise cu OpenPage
CloseHTMLEditControl	Subrutină pentru închiderea întregului control
CreateHTMLEditToolBar (iLeft, iTop)	Definește toolbar-ul cu comenzi rapide pentru formatarea textului. Parametrii specificați determină poziția toolbar-ului.
CreateHTMLPicturesToolBar (iLeft, iTop)	Definește toolbar-ul cu comenzi rapide pentru introducerea și manipularea imaginilor în document. Parametrii specificați determină poziția toolbar-ului.
CreateHTMLZoomToolBar (iLeft, iTop)	Definește toolbar-ul cu comanda de Zoom.

Aceste subrutine se găsesc implementate în fișierul *HTMLEditControl.asp* inclus în arhiva atașată. După încărcarea pe client a control-ului sarcinile revin scripturilor client implementate în fișierele *HTMLEditor.htc* și *HTMLEditorUtils.vbs*.

Pentru a permita editarea documentelor unui număr cât mai mare de utilizatori, editorul a fost înzestrat și cu o funcție de *Zoom* a paginii ce se editează, apelabilă prin intermediul toolbar-ului. Acest lucru este posibil datorită extensiei CSS implementate în IE, și anume a atributului Zoom cu următoarea sintaxă:

```
object.style.zoom [ = vMagnification ]
```

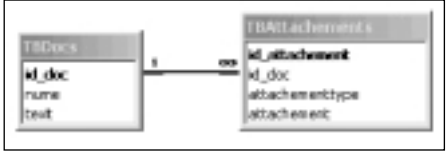
`vMagnification` poate fi setat în procente pentru a varia gradul de mărire sau micșorare. 100% semnifică afișare normală.

Structura documentului ce se editează

La prima vedere structura documentului este una simplă: în interiorul unui ferestre se editează un document HTML prin aplicarea de comenzi de formatare și inserarea de imagini de pe calculatorul local sau din rețeaua locală. Totuși această editare de HTML ridică unele probleme. Din însăși natura documentelor HTML se poate constata că acestea nu sunt unitare precum documentele .RTF sau .DOC ci compuse din mai multe părți/fișiere: în general unul care conține textul HTML și mai multe fișiere cu imagini.

Această observație ne ajută la stabilirea structurii bazei de date în care vom stoca aceste informații. Întrucât un document ce se editează poate avea un număr variabil de imagini nu ne va ajunge un singur tabel în baza de date pentru stocarea optimă a informațiilor. Vom folosi două tabele: unul pentru stocarea textului documentului și a altor informații suplimentare și altul pentru stocare imaginilor incluse în document, tabele legate printr-o relație *One-To-Many* (*figura 2*)

Figura 2: Structura bazei de date



La salvarea unui document textul acestuia după o serie de prelucrări se va duce în tabelul *TBDocs* iar imaginile asociate în tabelul *TBAttachements*. Prelucrările amintite se referă la parcurgerea textului HTML și înlocuirea stringurilor de tipul:

```
<img src="C:\images\img1.gif">
```

ce reprezintă calea către o imagine locală așa cum a fost ea introdusă de cel care editează documentul cu stringuri de tipul:

```
<img src="showimg.asp?FileID=123">
```

unde *showimg.asp* este o pagină ASP ce afișează o imagine din tabelul `TBAttachements`, iar parametru transmis prin QueryString reprezintă `ID`-ul acestei imagini. Aceste prelucrări ne obligă la următoarea modalitate de inserarea a întreg documentului în baza de date:

- se adaugă o nouă înregistrare în `TBDocs` și i se află id-ul din câmpul `id_doc`
- se inserează toate imaginile în tabelul `TBAttachements` împreună cu `id_doc` și la fiecare imagine i se află `id`-ul din câmpul `id_attachement`
- se prelucrează textul HTML prin înlocuirea căilor către imaginile locale cu calea către pagina showimg.asp cu parametru unul din `id`-urile imaginilor
- se updatează inregistrarea id_doc prin aplicarea textului paginii HTML.

Toate cele enunțate mai sus se pot aplica doar în cazul în care se salvează un document nou creat. Totuși, pentru a fi utilă, aplicația trebuie să aibă implementată și o funcție de editare a documentelor deja existente în baza de date. În cazul acestora situația se complică puțin datorită faptului că în urma editării, utilizatorul poate șterge din document una sau mai multe imagini ce s-au incărcat din baza de date și/sau poate adăuga pe lângă aceste imagini altele noi luate de pe calculatorul local.

Rezolvarea acestei probleme o vom face prin crearea unei liste cu imaginile ce s-au încărcat din BD la momentul deschiderii documentului în vederea editării. La salvarea lui mai facem odată o listă cu aceste imagini și din compararea acesteia cu lista originală se poate afla care imagini au fost sterse și prin urmare va trebui să le ștergem și din baza de date. Tot la salvare se face și o listă cu imaginile care s-au introdus local și care trebuie introduse în baza de date.

Concret formularul care face transportul documentului de pe client pe server are următoarele câmpuri:

`idopenpb` – ID-ul documentului ce se editează. Dacă acesta un document nou atunci acest câmp se lasă necompletat

`delimglist` – Listă cu ID-urile imaginilor ce au fost șterse din document și care trebuie șterse și din baza de date.
`uploadlist` – Listă cu numele câmpurilor de tip *file* ce conțin noile imagini ce se vor introduce in baza de date
`doctext` – Textul HTML al documentului care după prelucrarea amintită va fi introdus în baza de date.

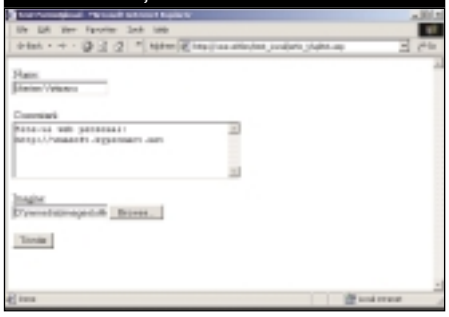
Pe lângă aceste câmpuri mai există un număr variabil de câmpuri file ce conțin imaginile ce trebuie introduse în baza de date. Despre modul de completare a acestor câmpuri se va vorbi pe larg în secțiunea referitoare la *ocolirea* protecției impuse de IE la upload-ul de fișiere.

Upload-area prin Web a fișierelor binare

Problema upload-ului de fișiere este una specială ce merită a fi tratată detaliat, ea fiind și una din sarcinile pe care trebuie să o rezolve editorul la momentul trimiterii către server a imaginilor ce se introduc în document. Acest lucru se realizează folosind un formular HTML cu câmpuri de tip *file*. Pentru ca upload-ul să funcționeze este necesar ca formul să aibă următoarele atribute:

```
method=post enctype="multipart/form-data"
```

Fig. 3.a: Testarea upload-ului de fișiere prin Web



Atributul *enctype* specifică tipul de codare MIME. În mod implicit atributul are valoarea *application/x-www-form-urlencoded*.

Întrucât *Microsoft* nu a prevăzut pentru ASP o modalitate de preluare pe server a datelor ce vin încapsulate într-un form cu codare *multipart/form-data*, va trebui să dezvoltăm o metoda de a extrage aceste date din formular. Mai întâi vom analiza structura formularului, așa cum ajunge el la serverul de web, prin crearea unei simple pagini web ce va trimite către server prin *post* conținutul unui form (*vezi figura 3.a*).

Acest form are următoarea definiție în HTML:

```
<form method=post enctype="multipart/form-data" action=
   "upltst.asp">
Nume:<br><input type=text name="nume"><br><br>
Comentarii:<br><textarea cols=40 rows=5 name=
   "comentarii"></textarea><br><br>
Imagine:<br><input type=file name="imagine"><br><br>
<input type=submit name="submit" value="Trimite"></form>
```

În el am adăugat două câmpuri normale (un *edit-text* și un *textarea*) precum și un câmp de tipul *file*. Acum să apăsăm butonul Submit și să așteptăm pe server sosirea datelor pe care le vom retransmite înapoi spre browser în următorul mod:

```
Response.BinaryWrite(Request.BinaryRead(Request.TotalBytes))
```

În fereastră va apare ceva asemănător cu imaginea din *figura 3.b*.

Deși pare complicat, după o analiză atentă se poate deduce modul în care sunt prezentate datele. Câmpurile formularului trimis apar delimitate în acest pachet de date de un text de genul următor:

```
-----------------------------7d11a9274e0100
```

iar formularul se încheie prin același text urmat de încă două semne --.

```
-----------------------------7d11a9274e0100--
```

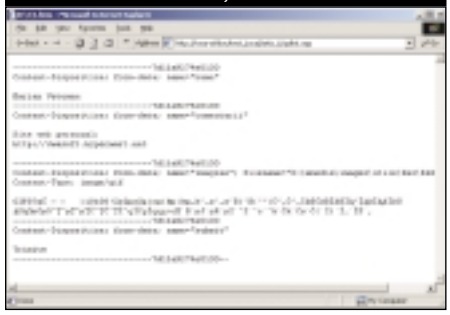
Față de primele câmpuri puse în formular (*edit-text* și *textarea*), câmpul de tip *file* are în structura de date 2 informații suplimentare:
`filename=` – Calea și numele fișierului local. Vom folosi această informație pentru a deosebi un câmp tip file de restul câmpurilor.
`Content-Type`: - Tipul conținutului din fișier

În general pentru fiecare câmp din formular putem extrage pe server următoarele informatii:
• Numele câmpului
• Valoarea sa (pentru câmpuri simple)
• Lungimea conținutului
• Conținutul binar (pentru câmpuri de tip file)
• Tipul conținutului (pentru câmpuri de tip file)
• Numele fișierului uploadat (pentru câmpuri de tip file)

Aceste date le vom cuprinde în structuri de tip *class* fără nici o metodă în felul următor:

```
Class clsFormItem
 Public FileName
 Public ContentType
 Public Value
 Public FieldName
 Public Length
 Public BinaryData
End Class
```

Figura 3.b: Structura datelor recepționate pe server



Pentru preluarea datelor din formular se va face o citire completă a acestuia în memorie: `ReceivedForm = Request.BinaryRead( Request.TotalBytes)` după care se va trece la interpretarea datelor. *ReceivedForm* nu se poate accesa ca un șir normal deoarece este în format binar iar *VBScript* lucrează cu string-uri Unicode. Din fericire acesta pune la dispoziție și funcții pentru lucrul cu date binare. Acestea sunt asemănătoare cu cele pentru lucrul cu string-uri cu excepția prezenței în numele acestora a unui B: `LenB, InstrB, MidB`, etc.

Totuși, dezvoltatorii *VBScript* au omis crearea unor funcții pentru conversia unui șir binar în șir Unicode și invers. Aceste funcții se realizează însă destul de ușor, codul lor putând fi urmărit în fișierul *Form-Upload.asp* din interiorul arhivei atașată acestui articol.

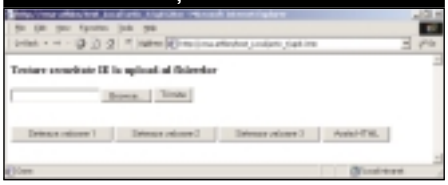
Protecția IE la upload-ul fișierelor

Pe partea client browserul Internet Explorer aplică asupra câmpurilor de tip *file* câteva constrângeri legate de securitatea datelor. Acest lucru complică munca programatorilor web care vor să trimită către server fișiere de pe mașina client într-un mod simplu întocmai ca dintr-o aplicație Windows. Restricția principală impusă câmpurilor de tip *file* constă în faptul că ele nu pot fi completate prin script în vederea prevenirii furtului de fișiere de pe calculatorul local de către situri web rău intenționate. Pentru a testa această protecție vom face câteva teste cu un formular la care încercăm prin diferite metode să setăm valoarea dintr-un câmp de tip *file*. Formularul și celelalte elemente din pagina de test au structura indicată în continuare:

```
<form  name="formupload" method=post
   enctype="multipart/form-data" action="upltst.asp">
<input type=file   name="imagine"
   value="d:\images\catz.gif">
<input type=submit name="submit"  value="Trimite">
</form>

<br>
```

```
<input id=but1 type=button value="Seteaza valoare 1">
<input id=but2 type=button value="Seteaza valoare 2">
<input id=but3 type=button value="Seteaza valoare 3">
<input id=but4 type=button value="Arata HTML">

<script language=vbscript>
sub but1_onclick
 formupload.imagine.value = "d:\images\catz.gif"
end sub

sub but2_onclick
 formupload.insertAdjacentHTML "beforeEnd", "<input
   type=file value='d:\images\catz.gif'>"
end sub

sub but3_onclick
 formupload.imagine.click
end sub

sub but4_onclick
 msgbox formupload.innerHTML
end sub
</script>
```

Figura 4.a: Testarea securității IE aplicate upload-ului de fișiere



Prima tentativă de setare a atributului *value* am făcut-o chiar din codul HTML prin specificarea valorii acestui atribut. După încărcarea acestei pagini în browser vom constata că imaginea specificată nu este trecută în câmpul *file*, acesta apărând gol (vezi figura 4.a.)

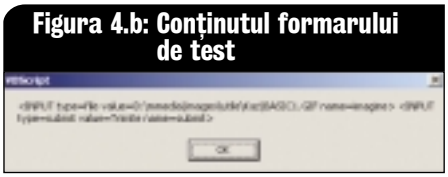
Sub form am plasat patru butoane: primele trei pentru încercarea setării din *VBScript* a acestui atribut prin trei metode iar al patrulea pentru afișarea conținutului HTML al form-ului.

Primul buton încearcă să scrie proprietatea value a câmpului de tip *file*. Se poate constata că apăsarea acestui buton nu are nici un fel de efect asupra formularului. Consultând documentația *Microsoft* se poate vedea ca proprietatea value este readonly deci este normal ca tentativa de a o modifica în acest fel să eșueze.

Cel de al doilea buton încearcă inserarea în form a unui conținut HTML ce reprezintă un câmp de tip *file* având atributul *value* setat. Într-un fel această metodă se aseamănă cu prima tentativă și ca și aceea și aceasta are același final: nereușirea setării câmpului.

Al treilea buton încearcă o abordare diferită a problemei, și anume încearcă ridicarea unui eveniment *onclick* ca și cum utilizatorul ar fi apăsat butonul *Browse* din dreapta câmpului file. Vom constata cu uimire cum va apare fereastra de tip *OpenDialog* cu ajutorul căreia putem selecta un fișier de pe calculatorul local. După această selectare câmpul file va fi completat automat. Apăsăm butonul *Arată HTML* și într-adevăr vedem că și conținutul HTML al formului conține informația de value în dreptul câmpului file (*vezi figura 4.b*)

Figura 4.b: Conținutul formarului de test

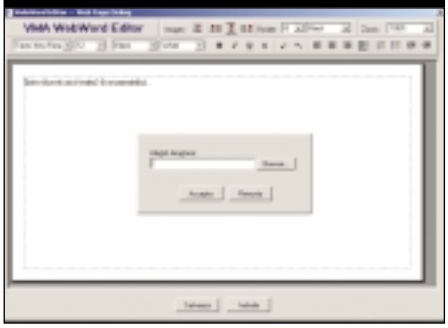


Am putea spune ca am setat valoarea atributului *value* al câmpului *file* prin script. Aceasta pare ca o gaură de securitate în browserul *Internet Explorer*. Pentru a duce experimentul până la capăt apăsăm acum butonul *Trimite* pentru a face upload la fișierul selectat prin script. Uimire! În loc să trimită conținutul formului la server IE va face ștergerea câmpului *file* efect constatat atât vizual cât și cu ajutorul lui *Arată HTML*. Mai apăsăm odată butonul trimite și de data asta reușim să trimitem formul pe server dar câmpul file este gol!

Așadar toate tentativele de setarea a câmpului *file* prin script au eșuat. Nu ne mai rămâne decât folosirea de astfel de elemente la care utilizatorul apasă explicit cu mouse-ul butonul *Browse* în vederea selectării unui fișier de pe calculator. O astfel de abordare ne îngreunează sarcina realizării aplicației deoarece utilizatorul poate introduce în documentul pe care îl realizează un număr variabil de imagini.

Ocolirea civilizată a protecției browserului. Vom rezolva această problemă prin folosirea unei pseudo-ferestre (un element DIV cu afișare ascundere) ce conține un form cu câmpuri file (vezi *figura 5*).

Figura 5: Inserarea unei imagini în document



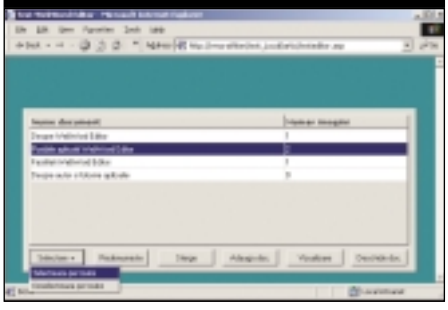
La fiecare apăsare a butonului *Insert-Image* aflat pe toolbar vom prezenta utilizatorului aceasta pseudo-fereastra în care îi dăm posibilitatea de a apăsa explicit pe butonul *Browse*. După selectarea imaginii se poate apăsa unul din butoanele *Acceptă* sau *Renunță* ambele având ca efect ascunderea pseudo-ferestrei. Deși fereastra s-a ascuns totuși câmpul *file* a rămas setat. La o nouă introducere de imagine vom afișa iar pseudo-fereastra în care anterior am creat un nou câmp de tip *file* pe care utilizatorul îl va completa în aceeași manieră.

În timpul operațiilor de editare a textului documentului se poate pierde sincronizarea între formularul ascuns cu câmpuri de tip file și imaginile introduse în document (de ex. prin ștergerea de imagini cu ajutorul tastei *Delete*) sau pot apărea în formularul ascuns dubluri de fișiere în cazul în care în document se introduce de mai multe ori aceeași imagine. Aceste probleme se rezolvă în momentul salvării prin inspectarea atât a conținutului documentului cât și a câmpurilor *file* din formularul ascuns. Câmpurile care apar în formular dar al căror conținut nu mai este prezent în document se șterg alături de dublurile câmpurile ce conțin același fișier.

Folosirea editorului

Pentru a folosi *Web-Word Editor* mai dezvoltăm și o mică pagină din care să se poate gestiona documentele deja existente în baza de date prin comenzi de stergere, redenumire, adaugare document nou, editare document existent (vezi *figura 6*).

Figura 6: Gestionarea documentelor din baza de date



Ca și editorul prezentat anterior și acest manager de documente își bazează funcționarea pe facilități avansate de DHTML puse la dispoziție de browserul IE5+. Aceste facilități au ajutat la crearea controlului de tip *TableGrid* în care sunt prezentate documentele și a meniului ce apare la apăsarea butonului *Selectare*.

Controlul de tip *TableGrid* este creat asemănător ca editorul de documente prin generarea sa pe server prin ASP și folosind script client pentru interacțiune cu utilizatorul.

Subrutina ASP principală care crează acest control este următoarea:

```
CreateTableControl(iLeft, iTop, iHeight, strTabNames,
   iTabSizes, iTipTable, tdcURL, bAllowSorting,
   strComponentName)
```

Având următorii parametrii de intrare:

`iLeft, iTop` - `iLeft` și `iTop` sunt folosiți pentru poziționarea controlului. Dacă aceștia lipsesc atunci controlul este pus cu poziție relativă

`iHeight` - înalțimea controlului (lungimea se calculează ca sumă a lungimilor coloanelor)

`strTabNames` - Array cu denumirile coloanelor

`iTabSizes` - Array cu mărimile pe orizontală a coloanelor. Array-ul trebuie să aibă același număr de elemente ca precedentul

`iTipTable` - Tipul tabelului (0-fără interacțiune, 1-cu selecție simplă, 2-cu selecție multiplă)

`tdcURL` - URL-ul sau fișierul de la care obiectul TDC primește date

`bAllowSorting` - Dacă este true atunci se permite sortarea datelor din tabel pe client prin acționarea headerelor coloanelor

`strComponentName` - Numele componentei

Pentru a funcționa corect acest control trebuie să primească date de la un TDC ce are datele în următorul format:

```
id|nume|prenume|email
1|Veteanu|Marian|mveteanu@yahoo.com
etc.
```

Despre ceea ce este un TDC (*Tabular Data Control*) și la ce poate fi folosit s-a vorbit într-un articol anterior.

Al doilea element din această pagină creat prin folosirea facilităților IE este meniul ce apare la apăsarea butonului *Select*. Acest meniu, după cum se poate observa din figura 6, depășește limita normală a paginii HTML, el afișându-se și peste bara de stare a browserului. Acest lucru este posibil prin crearea meniului într-o fereastra de tip pop-up introdusă odată cu *Internet Explorer 5.5*.

Metoda *createPopup* a obiectului *window* dă posibilitatea realizări de astfel de pop-up-uri utilizabile pentru afișarea de informații precum hinturi sau meniuri.

```
oPopup = window.createPopup( [vArgs])
```

Închiderea unui pop-up se realizează prin acționarea cu mouse-ul în afara sa. Imediat după crearea trebuie să se apeleze și metoda *oPopup.Show* în vederea afișării acestuia. Codul sursă necesar pentru implementarea meniului se găsește în fișierul *menu.vbs*.

În vederea vizualizării rapide a documentelor conținute în baza de date s-a mai implementat și o funcție de *QuickView* apelabilă prin intermediul butonului *Vizualizare* din managerul de documente. Acționarea acestui buton va determina deschiderea viewer-ului de documente care față de editor este mai rapid și permite de asemenea vizualizarea simultană a mai multor documente.

Figura 7: Viewer-ul de documente

Protejarea aplicațiilor Web

Pe măsură ce aplicațiile Web cresc în complexitate trebuie luate măsuri în vederea protejării lor atât împotriva furtului surselor cât și împotriva acțiunilor de utilizare incorectă a aplicațiilor de către utilizatorii începători. Datorită faptului că aplicațiile rulează în browser, acestea moștenesc de la programul gazdă un anumit comportament care uneori nu este necesar. De exemplu într-o aplicație web se dorește navigarea între diferitele secțiuni ale ei prin folosirea controalelor puse la dispoziție de aplicație și nu prin folosirea comenzilor browserului precum: *Back*, *Forward* sau *Refresh*.

Indiferent împotriva cui se protejează aplicația web, în continuare vor fi descrise câteva modalități simple de rezolvare a problemei.

Prima metodă de protecție, și cea mai simplă dintre toate, constă în inactivarea meniului contextual pentru a se preveni executarea comenzilor din acesta. Metoda larg folosită pe Internet în acest sens constă în tratarea evenimentului ridicat la click-dreapta pe document cu mouse-ul:

```
sub document_onmousedown
 if window.event.button = 2 then msgbox "Aceasta pagina
   este protejata."
end sub
```

Această abordare nu împiedică însa acțiunea utilizatorului de a apăsa tasta Win-Contextmenu care are același efect ca butonul drept al mouse-ului.

Din fericire, în IE, apariția meniului contextual ridică evenimentul *oncontextmenu* care poate fi tratat în vederea inactivării afișării acestui menu:

```
sub document_oncontextmenu
 window.event.returnValue = false
end sub
```

Un alt lucru util a se realiza într-o aplicație web constă în inactivarea tastelor *F5* și *Backspace*, care împreună cu deschiderea aplicației într-o fereastră full-screen sau fără bara cu comenzi de navigare asigură deplasarea în cadrul aplicației doar cu ajutorul controalelor puse la dispoziție de aceasta:

```
Sub document_onkeydown
 Dim key

 key = window.event.keyCode

 if key = 116 or key = 8 then     ' Daca se apasa F5 sau
   Backspace nu se face nimic...
   window.event.keyCode = 0
   window.event.returnValue = false
 end if
End Sub
```

Într-o aplicație reală subrutina prezentată mai sus trebuie îmbunătățită în vederea ne dezactivării tastei *Backspace* în controalele de editare. În sursele atașate articolului este prezentată o astfel de subrutină îmbunătățită.

Alte două lucruri interesante care se poate realiza în *Internet Explorer* în vederea setării browserului a unui comportament cât mai apropiat de aplicațiile *Windows* normale și care deasemenea pot fi considerate ca elemente de protecție constau în crearea de texte ce nu se pot selecta cu mouse-ul și inactivarea facilităților de completare automată a câmpurilor formularelor.

În exemplul următor textul din interiorul tag-ului `DIV` nu se poate selecta cu mouse-ul:

```
<body UNSELECTABLE="on">
...
<div UNSELECTABLE="on" style="cursor:default;">Acest text nu
   se poate selecta cu mouseul</div>
...
</body>
```

Din nefericire atributul `UNSELECTABLE` nu se poate moșteni, el trebuind aplicat manual tuturor elementelor care nu se doresc a fi selectate. Această facilitate de text neselectabil a fost folosită chiar în realizarea *WebWord Editor*-ului pentru a preveni selectarea unui text din pagina web din afara zonei de editare și aplicarea comenzilor de formatare.

În ceea ce privește dezactivarea facilității de autocompletarea a câmpurilor formularelor acest lucru poate fi privit ca o întărire a securității anumitor pagini. De exemplu în cazul unui formular în care se introduce numele, parola, adresa, etc. este de dorit dezactivarea autocompletării mai ales dacă se folosește un calculator public:

```
<input autocomplete="off" type="text">
```

O ultimă protecție care se poate folosi constă în ascunderea codului sursă al scripturilor de ochii utilizatorilor începători. Începând cu versiune 5 a lui IE, *Microsoft* a oferit posibilitatea criptării codului sursă al scripturilor server și client. Din păcate metoda folosită are drept scop ascunderea surselor de utilizatorii începători și nu de hackerii perseverenți.

Să presupunem că într-o pagină web avem următorul script:

```
<script language=vbscript>
 sub ShowMsg
   msgbox "Acesta este un mesaj."
 end sub
</script>
```

Prin folosirea utilitarului `screnc.exe` disponibil pe situl *Microsoft*, scriptul anterior se transformă în ceva de genul:

```
<script language=VBScript.Encode>#@~^MBcAAA==
   @#@&P~1WswV'6sGNsWl ... dgGAA==^#~@</script>
```

Browserele IE5+ vor ști să ruleze scripturile scrise cu *VBScript.Encode*, restul ignorandu-le. Ca o dovadă a punctelor slabe ale acestei metode de criptare, pe Internet există deja programe care permit decriptarea surselor criptate.

Sper că prin acest articol am reușit să prezint câteva din facilitățile importante introduse de browserele *Internet Explorer 5.0* și *Internet Explorer 5.5* și deasemenea să deschid cititorilor noi perspective în ceea ce privește realizarea siturilor și aplicațiilor web. Pentru mai multe informații analizați codul sursă complet al acestui articol pe care îl găsiți la `http://www.pcreport.ro` și deasemenea consultați sit-urile enumerate la bibliografie.

Marian Vețeanu este student la Universitatea din Pitești; poate fi contactat la: vmasoft@yahoo.com, http://vmasoft.hypermart.net.■76

Bibliografie

- ✓ http://msdn.microsoft.com
- ✓ http://www.asptoday.com
- ✓ http://www.winscriptingsolutions.com