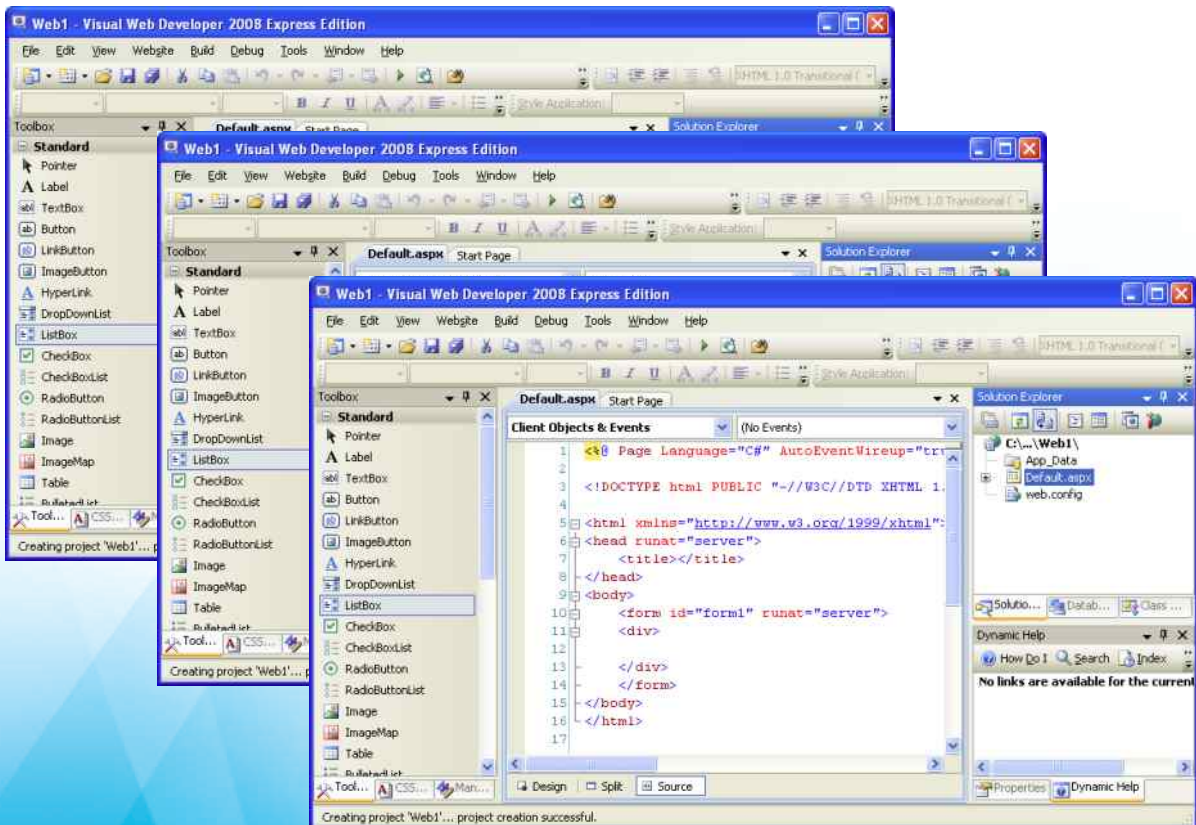


Programarea WEB cu



ASP.NET



I.	PRINCIPII GENERALE ALE PROIECTĂRII INTERFEȚELOR WEB	5
I.1.	INTRODUCERE.....	5
I.2.	REALIZAREA INTERFEȚELOR WEB UTILIZÂND LIMBAJUL DE MARCARE HTML	6
I.2.1.	<i>Ce este HTML ?</i>	6
I.2.2.	<i>Structura unui document HTML</i>	7
I.2.3.	<i>Elemente HTML avansate</i>	8
I.2.3.1	<i>Tabele.....</i>	8
I.2.3.2	<i>Cadre</i>	9
I.2.3.2.1	<i>Cadre interne</i>	10
I.2.3.2.2	<i>Deschiderea documentelor în alte cadre</i>	11
I.2.3.3	<i>Layeres.....</i>	12
I.2.3.4	<i>Formulare.....</i>	12
I.2.4.	<i>Evaluare.....</i>	16
I.3.	UTILIZAREA TEHNICII CSS PENTRU FORMATAREA DOCUMENTELOR WEB	19
I.3.1.	<i>Ce este un stil?</i>	19
I.3.2.	<i>Definiții de stil.....</i>	20
I.3.2.1	<i>Definiții de stil inline</i>	21
I.3.2.2	<i>Definiții de stil încapsulate (interne)</i>	21
I.3.2.3	<i>Definiții de stil extern</i>	23
I.3.3.	<i>Stiluri în cascadă</i>	24
I.3.4.	<i>Clase de stiluri</i>	24
I.3.5.	<i>Stiluri identificator.....</i>	25
I.3.6.	<i>Pseudoclase și pseudoelemente.....</i>	26
I.3.7.	<i>Stiluri pentru liste.....</i>	28
I.3.8.	<i>Casete în CSS.....</i>	30
I.3.9.	<i>Poziționare în CSS.....</i>	31
I.3.10.	<i>Notatii și unități de măsură.....</i>	32
I.3.11.	<i>Evaluare.....</i>	34
II.	MEDIUL DE LUCRU VISUAL WEB DEVELOPER EXPRESS 2008	36
II.1.	MEDIUL DE LUCRU.....	36
II.2.	CUM MANEVRĂM PANOURILE	37
II.3.	MENIUL VIEW	38
II.4.	PAGINA DE START	38
II.5.	PUBLICAREA UNUI SITE WEB	38
II.6.	COMPILAREA DINAMICĂ A SITE-ULUI.....	39
II.7.	WEB SITE/WEB PROJECT	39
II.8.	CREAREA UNUI SITE WEB	40
II.9.	DESPRE OPȚIUNEA LOCATION(FILE SYSTEM, HTTP, FTP).....	41
II.10.	CREAREA ȘI UTILIZAREA DIRECTOARELOR.....	42
II.11.	EDITAREA PAGINILOR	42
II.12.	SCHIMBAREA PROPRIETĂȚILOR.....	43
II.13.	SALVAREA MODIFICĂRILOR	43
II.14.	DESPRE FIȘIERELE COD	43
II.15.	VIZUALIZAREA PAGINILOR ÎNTR-UN BROWSER WEB.....	44
II.16.	MASTER PAGES	44
II.17.	SERVERE WEB ÎN VISUAL WEB DEVELOPER.....	45
II.18.	ASP.NET DEVELOPMENT SERVER	45
II.19.	RULAREA SERVERULUI INTEGRAT	46
II.20.	SECURITATE ÎN ASP.NET DEVELOPMENT SERVER	46
III.	LIMBAJUL DE SCRIPTING SERVER-SIDE ASP.NET.....	47
III.1.	STRUCTURA UNEI PAGINI ASP.NET	47
III.1.1.	<i>Controale ASP.NET.....</i>	49
III.1.2.	<i>Ciclul de viață al unei pagini web.</i>	50
III.1.3.	<i>Aplicații rezolvate</i>	51
III.2.	LIMBAJUL C#.....	53
III.2.1.	<i>Vocabularul limbajului.....</i>	54
III.2.2.	<i>Tipuri de date.....</i>	55
III.2.3.	<i>Operatori</i>	56

III.2.4.	<i>Conversii</i>	58
III.2.5.	<i>Funcții matematice</i>	59
III.2.6.	<i>Instrucțiuni C#</i>	60
III.2.7.	<i>Tablouri în C#</i>	67
III.2.8.	<i>Șiruri de caractere</i>	71
III.2.9.	<i>Date calendaristice</i>	73
IV.	MODELUL CLIENT-SERVER	75
IV.1.	CONTROALE SERVER WEB.....	75
IV.1.1.	<i>Label</i>	76
IV.1.2.	<i>Button, LinkButton, ImageButton</i>	76
IV.1.3.	<i>TextBox</i>	78
IV.1.4.	<i>CheckBox, CheckBoxList</i>	79
IV.1.5.	<i>RadioButton</i>	81
IV.1.6.	<i>RadioButtonList</i>	82
IV.1.7.	<i>BulletList</i>	83
IV.1.8.	<i>Image</i>	84
IV.1.9.	<i>DropDownList</i>	84
IV.1.10.	<i>HyperLink</i>	85
IV.1.11.	<i>Table, TableRow, TableCell</i>	86
IV.1.12.	<i>MultiView, View</i>	87
IV.1.13.	<i>FileUpload</i>	87
IV.1.14.	<i>Evaluare</i>	89
IV.2.	POST BACK.....	90
IV.2.1.	<i>Evaluare</i>	92
IV.3.	CONTROALE PENTRU VALIDAREA DATELOR.....	93
IV.3.1.	<i>RequiredFieldValidator</i>	93
IV.3.2.	<i>RangeValidator</i>	93
IV.3.3.	<i>RegularExpressionValidator</i>	93
IV.3.4.	<i>CompareValidator</i>	93
IV.3.5.	<i>CustomValidator</i>	94
IV.3.6.	<i>Evaluare</i>	97
IV.4.	CONTROALE SERVER WEB AVANSATE.....	98
IV.4.1.	<i>ImageMap</i>	98
IV.4.2.	<i>Ad Rotator</i>	101
IV.4.3.	<i>Calendar</i>	103
IV.4.4.	<i>Evaluare</i>	107
IV.5.	CONECTAREA LA O SURSĂ DE DATE A CONTROALELOR.....	107
IV.6.	PĂSTRAREA INFORMAȚIILOR ÎNTRE PAGINILE WEB	109
IV.6.1.	<i>Controlul HiddenField</i>	110
IV.6.2.	<i>ViewState</i>	111
IV.6.3.	<i>Cookies</i>	113
IV.6.4.	<i>Query String</i>	114
IV.6.5.	<i>Session</i>	116
IV.6.6.	<i>Application</i>	119
IV.6.7.	<i>Evaluare</i>	121
V.	INTERACȚIUNEA CU BAZE DE DATE WEB	122
V.1.	ROLUL BAZELOR DE DATE.....	122
V.2.	ACCESAREA BAZELOR DE DATE WEB.....	123
V.3.	PROIECTAREA BAZELOR DE DATE.....	125
V.3.1.	<i>Entități, instanțe, attribute, identificator unic</i>	125
V.3.2.	<i>Relații între entități</i>	126
V.3.3.	<i>Evaluare</i>	127
V.4.	CONFIGURAREA BAZEI DE DATE	128
V.4.1.	<i>Evaluare</i>	136
V.5.	ACCESUL DIRECT LA DATE.....	137
V.5.1.	<i>Limbajul SQL- Elemente de bază</i>	137
V.5.2.	<i>Comenzi de manipulare a datelor</i>	140
V.5.2.1	Comanda SELECT.....	140
V.5.2.2	Gruparea datelor	143

V.5.2.3	Sortarea datelor.....	145
V.5.2.4	Interogări multiple.....	146
V.5.2.5	Comanda UPDATE.....	149
V.5.2.6	Comanda INSERT.....	149
V.5.2.7	Comanda DELETE.....	150
V.5.3.	<i>Comenzi de deținire a datelor</i>	151
V.5.3.1	Crearea tabelor.....	151
V.5.3.2	Modificarea structurii unei tabele.....	152
V.5.3.3	Redenumirea și ștergerea unei tabele.....	153
V.5.3.4	Acordarea / revocarea unor privilegii.....	154
V.5.4.	<i>Evaluare</i>	156
V.6.	MANIPULAREA BAZELOR DE DATE WEB PRIN INTERMEDIUL OBIECTELOR ADO.NET	158
V.6.1.	<i>Arhitectura ADO.NET</i>	158
V.6.2.	<i>Furnizori de date (Data Providers)</i>	158
V.6.3.	<i>Accesul direct la date prin intermediul ADO.NET</i>	159
V.6.4.	<i>Crearea unei conexiuni</i>	160
V.6.5.	<i>Command</i>	168
V.6.5.1	Selectarea datelor.....	170
V.6.5.2	Inserarea datelor.....	170
V.6.5.3	Actualizarea datelor.....	171
V.6.5.4	Ștergerea datelor.....	171
V.6.6.	<i>DataReader</i>	172
V.6.7.	<i>Comenzi parametrizate</i>	173
V.6.8.	<i>Studiu de caz</i>	174
V.7.	LUCRUL ÎN MOD DECONECTAT.....	184
V.7.1.	<i>DataAdapter</i>	184
V.7.2.	<i>DataSet</i>	185
V.7.3.	<i>Proiectare DataSet în mediu vizual</i>	188
V.8.	LUCRUL CU MAI MULTE TABELE.....	193
V.9.	PROCEDURI STOCATE (STORED PROCEDURES).....	195
V.10.	CONTROALE .NET LEGATE LA DATE.....	198
V.10.1.	<i>Controale pentru sursa de date</i>	198
V.10.2.	<i>Controlul GridView</i>	200
V.10.3.	<i>Controalele DetailsView și FormView</i>	207
V.10.4.	<i>Alte controale legate la date</i>	208
V.10.4.1	Repeater.....	208
V.10.4.2	DataList.....	209
V.10.4.3	DropDownList.....	210
V.10.4.4	CheckBoxList.....	211
V.10.4.5	RadioButtonList.....	212
V.10.5.	<i>Evaluare</i>	213
VI.	SECURITATEA APLICAȚIILOR ASP.NET	214
VI.1.	WINDOWS AUTHENTICATION.....	214
VI.2.	FORMS-BASED AUTHENTICATION.....	215
VI.3.	SECURIZAREA DIN APLICAȚIA WEB.....	215
VII.	PROIECTAREA ȘI REALIZAREA UNEI APLICAȚII WEB	216
VII.1.	REALIZAREA INTERFEȚEI.....	216
VII.1.1.	<i>MasterPages</i>	216
VII.1.2.	<i>Foi de stiluri</i>	222
VII.1.3.	<i>Controalele web server din MasterPage</i>	225
VII.1.4.	<i>Conectarea la sursa de date a controalelor din MasterPage</i>	229
VII.2.	HOME.ASPX.....	233
VII.3.	MOVIE.ASPX.....	234
VII.4.	DETALII.ASPX.....	242
VII.5.	UPLOAD.ASPX.....	246
VII.6.	ADAugĂFILM.ASPX.....	248
VII.7.	CONTACT.ASPX.....	254
VII.8.	EVALUARE.....	259
VIII.	TEST DE VERIFICARE A CUNOȘTIINȚELOR	259

I. Principii generale ale proiectării interfețelor Web

I.1. Introducere

ASP.NET este un set de tehnologii care ne permit crearea de aplicații web. Este evoluția de la Microsoft Active Server Pages (ASP), dar beneficiază de suportul platformei de dezvoltare Microsoft .NET.

Una dintre cele mai importante calități ale ASP.NET este timpul redus necesar dezvoltării aplicațiilor web. Atât tehnologia în sine, cât și uneltele de dezvoltare de aplicații web de la Microsoft (cum ar fi Visual Web Developer Express - VWD) – reduc considerabil timpul de dezvoltare al aplicațiilor web față de alte tehnologii – prin simplitatea unui limbaj de programare "managed" de genul C# sau Visual Basic .NET, prin colecția bogată de biblioteci de clase și controale .NET care oferă foarte multă funcționalitate "out of the box", prin orientarea pe construirea de aplicații web a mediului de dezvoltare VWD.

Chiar dacă ASP.NET este gândit pentru a dezvolta aplicații web foarte complexe – prin faptul că se bazează pe .NET, prin faptul că se insistă pe un model de dezvoltare OOP, respectiv pe separarea interfeței de logica aplicației – totuși, este extrem de simplu ca folosind ASP.NET să dezvoltăm aplicații mici, de genul magazinelor online, al aplicațiilor care sunt pur și simplu un "front-end" pentru o bază de date, sau al site-urilor personale.

ASP.NET cuprinde toate tehnologiile necesare pentru a dezvolta o aplicație web, scriind cantitatea minimă de cod. Limbajele de programare care pot fi utilizate pentru a crea aplicații ASP.NET sunt cele suportate de platforma .NET – cum sunt Visual Basic .NET și C#, iar o altă caracteristică importantă a acestor limbaje (înafara faptului că sunt "managed") este că au fost create având în vedere paradigma programării orientată pe obiecte. Totul din .NET, și evident din ASP.NET, este un obiect.

Evident, orice site / aplicație web trebuie să fie găzduită pe un server pentru a putea fi utilizată. Chiar dacă în capitolele viitoare vom discuta mai mult despre instalarea aplicațiilor ASP.NET, aici aș dori să prezint pe scurt variantele de găzduire.

Pentru o persoană / companie care dorește să beneficieze de o aplicație ASP.NET, după ce a fost dezvoltată, trebuie instalată undeva. Presupunând că ea a fost dezvoltată pentru .NET, aplicația are nevoie de un server web IIS (internet Informations Services).

Există două variante de a găzdui aplicațiile ASP.NET: (1) intern, pe serverele proprii sau (2) extern, la o firmă care oferă servicii de găzduire (hosting). Decizia trebuie luată ținând cont de câțiva parametri:

- Costul de mentenanță. Intern costă administrarea serverelor cu tot ce presupune asta: hardware, software, specialiști. Extern costă un abonament fix pe lună/an.
- Securitatea. Dacă vorbim de o aplicație extrem de importantă și care manipulează informații sensibile, probabil vom dori să fie sub controlul propriu.
- Etc.

Pentru programatorii care doresc să aibă un site dezvoltat din pasiune sau pentru a învăța, dar vor totuși să îl aibă instalat undeva, variantele de mai sus devin: (1) acasă, pe Windows XP / Vista cu IIS; (2) la o firmă care oferă servicii de hosting gratuit. Da, există variante de acest gen, unde vă puteți instala propriul site ASP.NET fără să vă coste nimic (căutați pe www.live.com "asp.net free hosting").

Pe perioada dezvoltării unei aplicații web, nu este nevoie ca aceasta să fie găzduită pe un server IIS, ci poate fi rulată din Visual Web Developer folosind serverul web integrat (vezi mai multe în capitolul II).

I.2. Realizarea interfețelor Web utilizând limbajul de marcare HTML

De ce HTML?

- este simplu de înțeles și de utilizat
- poate fi creat utilizând orice editor de texte (este un fișier ASCII)
- oferă structurarea formatării
- este independent față de platformă

I.2.1. Ce este HTML ?

HTML¹ reprezintă scheletul oricărei pagini Web, el descriind formatul primar în care documentele sunt vizualizate și distribuite pe Internet. HTML nu este un limbaj de programare, deci nu veți lucra aici cu variabile, expresii, tipuri de date, structuri de control. HTML este un limbaj descriptiv, prin care sunt descrise elementele structurale ale paginii Web: titluri, liste, tabele, paragrafe, legături cu alte pagini, precum și aspectul pe care îl are pagina din punct de vedere grafic. Fiind un limbaj de marcare, el utilizează etichete (marcaje²) ce dau indicații browsere-lor cu privire la ierarhizarea și afișarea informațiilor.

¹ *Hypertext Markup Language* în engleză

² *tags* în engleză

I.2.2. Structura unui document HTML

Etichetele HTML sunt de două tipuri:

- etichete **container**³

Etichetele **container** sunt de forma: `<tag> bloc de text </tag>` specificând formatul în care va fi afișat textul conținut între eticheta de început și cea de final. Majoritatea etichetelor sunt de acest tip. O particularitate a etichetelor container este că, în cazul unora dintre ele, prezența etichetei de închidere (`</tag>`) este opțională.

- etichete **vide**⁴

Etichetele **vide** au forma: `<tag>`. Acest tip de etichete nu se referă la formatul textelor, ci la introducerea anumitor elemente, ca de exemplu: început de paragraf, sfârșit de linie, linie orizontală și altele. Astfel ele dau indicații browserului despre elementul introdus și despre cum să afișeze acel element.

Un document HTML este structurat astfel:

1. zona **head** (antet) cu etichetele `<head> </head>`
2. zona **body** (corp) cu etichetele `<body> </body>` sau `<frameset> </frameset>`

Exemplu: codul HTML prezentat în acest exemplu utilizează următoarele marcaje⁵ :

`<p>`-pentru definirea unui paragraf
`<hr>`-pentru trasarea unei linii orizontale
``-pentru formatarea fontului
``-pentru inserarea unei imagini
`<i>`-pentru definirea unui stil înclinat

```
<html>
<head><title>Exemplu</title></head>
<body bgcolor=gray leftmargin="100" topmargin="50">
<body >
<p>Linie orizontala de culoare albastra si grosime 2
<hr color=blue size=3>
<p><font face="Arial" color="red" size="4">
Textul este scris cu fontul "Arial", culoare rosie si marime 7.
<p><i>Am inserat o imagine</i>
</body>
</body>
</html>
```

Linie orizontala de culoare albastra si grosime 2

Textul este scris cu fontul "Arial", culoare rosie si marime 6.



Am inserat o imagine

³ *container tags* în engleză

⁴ *empty tags* în engleză

⁵ aceste elemente au fost studiate la Tehnologia informației și comunicațiilor, în clasa a IX-a

I.2.3. Elemente HTML avansate

I.2.3.1 Tabele

Tabelele ne permit să construim o rețea dreptunghiulară de domenii, fiecare domeniu având propriile opțiuni de formatare: culoarea fondului, culoarea textului, alinierea textului etc. Prezentarea datelor sub formă de tabele oferă importante avantaje: claritate, sistematizare, posibilități de comparare.

Marcarea unui tabel se efectuează printr-un tag de introducere a tabelului și de definire a atributelor globale. Acesta include și definițiile pentru liniile și celulele tabelului.

Sintaxa generală pentru declararea unui tabel este:

```
<table>
  <caption>...</caption>
  <TR><TH><TH> ... </TR>
  <TR><TD><TD> ... </TR>
  ...
  <TR><TD><TD> ... </TR>
</table>
```

unde etichetele:

<table></table>	delimitează tabelul
<tr></tr>	delimitează o linie a tabelului
<td></td>	delimitează o celulă de date a tabelului
<th></th>	delimitează o celulă a primei linii din tabel (a capului de tabel)
<caption></caption>	delimitează titlul tabelului

Atributele etichetelor **table** și **td** sunt:

Atribut	Semnificație	table	td
border	stabilește lățimea bordurii	*	
width	stabilește lățimea	*	*
height	stabilește înălțimea	*	*
bgcolor	stabilește culoarea de fundal	*	*
background	stabilește imaginea de fundal	*	*
cellspacing	stabilește distanța dintre celule	*	
cellpadding	stabilește distanța dintre marginea celulei și conținut	*	
align	aliniază pe orizontală conținutul (left, right, center)		*
valign	aliniază pe verticală conținutul (top, bottom, middle)		*
colspan	unește celula cu cea din dreapta ei		*
rowspan	stabilește lățimea bordurii		*

Exemplu: pagina următoare conține un tabel fără formătări.

```
<body>
  <h3 align=left >tabel</h3>
  <table>
    <tr><td>HTML</td><td>TABELE</td></tr>
    <tr><td>FORMULARE</td>
      <td>CADRE</td></tr>
  </table>
</body>
```

tabel

```
HTML      TABELE
FORMULARE ASP
CADRE
```


Exemplu: pagina următoare conține un tabel cu bordură, având culoare stabilită de fundal, cu celule unite.

```
<body>
<h3 align=left >tabel</h3>
<table border=2 bgcolor=gray>
<tr><td rowspan=3>HTML</td><td>TABELE</td></tr>
<tr><td>FORMULARE</td><td bgcolor=red>ASP</td></tr>
<tr><td>CADRE</td><td bgcolor=red>Visual Web DevExpress
2008</td>
</tr>
</table></body></html>
```

tabel

	TABELE	
HTML	FORMULARE	ASP
	CADRE	Visual Web DevExpress 2008

I.2.3.2 Cadre⁶

Exemplele prezentate anterior încărcă o singură pagina HTML în fereastra browserului. Sunt și situații în care imaginea afișată de browser este formată din mai multe pagini HTML numite **cadre**. Caracteristica acestor pagini este că perechea de etichete `<body> </body>` este înlocuită de `<frameset> </frameset>`, iar în interiorul lor cadrele sunt delimitate de `<frame>` și `</frame>`.

Observații:

Există browsere care nu suportă cadre. Pentru acestea se utilizează în interiorul blocului `<frameset>` eticheta `<noframes> </noframes>`. Dacă browserul poate să interpreteze cadre, va ignora ce se găsește în această porțiune, iar dacă nu, materialul cuprins în zona `<noframes> </noframes>` va fi singurul care va fi recunoscut și afișat.

Atributele etichetei **frameset** sunt:

Atribut	Semnificație
cols	împarte pagina în coloane și are valori exprimate în procente din dimensiunea ferestrei sau număr de pixeli sau * ⁷ (spațiul rămas)
rows	împarte pagina în rânduri cu aceleași valori ca la cols
bordercolor	stabilește culoarea tuturor chenarelor conform modelului #rrggbb ⁸
frameborder	permite/inhibă afișarea chenarelor cu valorile yes sau no

⁶ în engleză frames

⁷ dacă mai multe elemente din listă sunt configurate cu *, atunci spațiu disponibil rămas se va împărți în mod egal între ele

⁸ culorile pot fi precizate prin nume sau prin construcția #rrggbb, unde r(red), g(green) și b(blue) sunt cifre hexazecimale

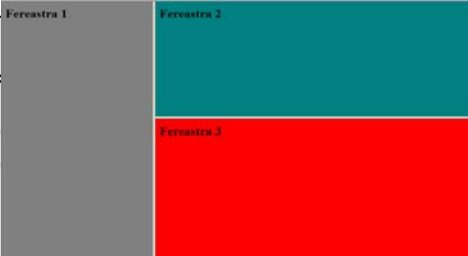
Cadrele sunt introduse prin perechea de etichete <frame> </frame>, și suportă atributele:

Atribut	Semnificație
name	stabilește numele asociat cadrului
src	stabilește fișierul sau adresa fișierului introdus
bordercolor	stabilește culoarea chenarului cadrului curent conform
noresize	dezactivează posibilitatea vizitatorului de a redimensiona cadrul
scrolling	permite/inhibă adaugarea barelor de defilare cu valorile yes no si auto
frameborder	stabilește dacă se afișează chenarul cadrului (1-implicit) sau nu (0)
marginheight	permite stabilirea distanței în pixeli dintre conținutul unui cadru și marginile verticale ale cadrului
marginwidth	permite stabilirea distanței în pixeli dintre conținutul unui cadru și marginile orizontale ale cadrului

Exemplu: pagină cu două cadre verticale în proporția 30% și 70% din lățimea totală.

Cadrul din dreapta este împărțit la rândul său în două cadre orizontale.

```
<frameset cols="30%,*">
<frame src="f1.html" name="f1">
<frameset rows="40%,*">
  <frame src="f2.html" name="f2">
  <frame src="f3.html" name="f3">
</frameset >
</frameset>
```



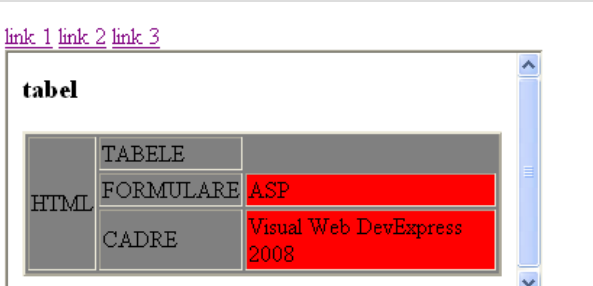
I.2.3.2.1 Cadre interne

Un cadru intern este specificat prin intermediul marcajului <iframe> </iframe>. Acesta definește o arie rectangulară în interiorul documentului, arie în care browserul va afișa un alt document HTML complet, inclusiv marginile și barele de derulare. Un cadru intern se inserează într-o pagină Web în mod asemănător cu o imagine, în interiorul blocului <body>.

Exemplu: pagină cu trei link-uri; acestea deschid paginile referite de ele în cadrul intern din centrul paginii.

```
<html>
<head><title>Cadre interne</title></head>
<body>
  <a href="t1.html" target="mijloc">link 1</a>
  <a href="t2.html" target="mijloc">link 2</a>
  <a href="c1.html" target="mijloc">link 3</a>
  <center>
    <iframe width="60%" height="50%" name="mijloc" src="c2.html">
    </iframe>
  </center>
</body>
</html >
```

[link 1](#) [link 2](#) [link 3](#)



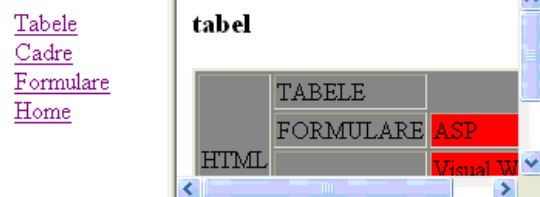
I.2.3.2.2 Deschiderea documentelor în alte cadre

Dacă într-unul dintre documentele deschise în cadru există link-uri, acestea vor deschide paginile referite de ele în cadrul curent. Acest comportament se poate schimba prin plasarea în eticheta `<a>` a atributului **target**, care precizează numele ferestrei (cadrului) în care se va încărca pagina nouă referită de legătură, conform sintaxei:

```
<a href="URL" target="nume_cadru"> </a>
```

Exemplu: pagină cu două cadre de tip coloană. În cel din stânga se va deschide documentul **c5.html**, iar în cel din dreapta, documentul **c7.html**. Cel de-al doilea cadru a fost numit "cadru_dreapta".

```
<frameset cols="20%, *">
<frame src="c5.html">
<frame src="c7.html"
  name="cadru_dreapta">
</frameset>
```



c5.html: acest document conține patru link-uri. Prin intermediul atributului **target** am specificat faptul că toate legăturile încarcă paginile referite în cadrul din dreapta.

```
c5.html
<body>
<a href="t2.html" target="cadru_dreapta">Tabele</a><br>
<a href="t1.html" target="cadru_dreapta">Cadre</a><br>
<a href="t3.html" target="cadru_dreapta">Formulare</a><br>
<a href="c7.html" target="cadru_dreapta">Home</a><br>
</body>
```

Atributul **target** al etichetei `<frame>` acceptă anumite valori predefinite:

Atribut	Semnificație
<code>_self</code>	încărcarea noii pagini are loc în cadrul curent
<code>_blank</code>	încărcarea noii pagini are loc într-o fereastră nouă, anonimă
<code>_parent</code>	încărcarea noii pagini are loc în cadrul părinte al cadrului curent dacă acesta există, altfel are loc în fereastra curentă a browserului
<code>_top</code>	încărcarea noii pagini are loc în fereastra browserului ce conține cadrul curent

Exemplu: pagină cu 3 link-uri în care atributul **target** ia 3 valori.

```
<a href="t1.html" target="_blank">Aceasta legatura incarca pagina
t1.html in alta fereastra</a><br>
<a href="c1.html" target="_self">Aceasta legatura incarca pagina
c1.html in fereastra curenta</a><br>
<a href="c2.html" target="_parent"> Aceasta legatura incarca pagina
c2.html in fereastra parinte</a><br>
```

Observații:

Utilizarea cadrelor prezintă o serie de avantaje:

- ușurința în navigare
 - reducerea timpilor de încărcare
- și o serie de dezavantaje:
- indexarea paginii de către motoarele de căutare este mai dificilă
 - există browsere care nu suportă cadrele; este recomandat să existe pentru fiecare astfel de pagină și o versiune fără cadre, ceea ce implică un efort suplimentar.

I.2.3.3 Layere⁹

Layer-ele sunt elemente HTML asemănătoare frame-urilor, adică sunt niște containere pentru orice altceva ar putea intra într-o pagina HTML, dar spre deosebire de acestea, se pot suprapune (ca și regiunile unei hărți). Etichetele **<layer></layer>** definesc un layer.

Atributele etichetei **<layer>** sunt:

Atribut	Semnificație
name	stabilește numele asociat layer-ului
left/ top	stabilesc poziția în pagină a layer-ului
bgColor	stabilește culoarea de fundal a layer-ului
width/ height	stabilesc dimensiunile layer-ului
visibility	permite/inhibă vizibilitatea layer-ului prin valorile SHOW-implicită/HIDE
src	stabilește adresa fișierului care conține informațiile ce sunt preluate în layer

 **<layer left=100 top=100 bgcolor=red>**

Am definit un layer

</layer>

Observații:

Putem avea un layer în alt layer. În acest caz valorile atributelor **left** și **top** ale layer-ului din interior vor indica poziția acestuia față la marginea de sus și marginea din stânga a layer-ului care îl cuprinde.

Layer-ele sunt acceptate doar de versiunile de la Netscape 4.0 în sus.

I.2.3.4 Formulare

Un formular este un ansamblu de zone active alcătuit din casete combinate, câmpuri de editare, butoane radio, butoane de comandă etc. Formularele asigură construirea unor pagini Web care permit utilizatorilor să introducă informații și să le transmită serverului. O sesiune cu o pagină Web ce conține un formular cuprinde două etape:

⁹ straturi

- utilizatorul completează formularul și trimite serverului (prin apăsarea butonului de expediere) datele înscrise în formular.
- o aplicație dedicată de pe server (un script) analizează informațiile transmise și, în funcție de configurația scriptului, fie stochează datele într-o bază de date, fie le transmite la o adresă de mail indicată de dumneavoastră. Dacă este necesar, serverul poate expedia și un mesaj de răspuns utilizatorului.

Un formular este definit într-un bloc delimitat de etichetele **<form>** **</form>**. În interiorul blocului sunt incluse:

- elementele formularului, în care vizitatorul urmează să introducă informații,
- un buton de expediere, la apăsarea căruia, datele sunt transmise către server,
- opțional, un buton de anulare, prin care utilizatorul poate anula datele înscrise în formular.

Cele mai importante atribute ale etichetei **<form>** sunt:


Atribut	Semnificație	
action	comunică browserului unde să trimită datele introduse în formular. În general valoarea atributului action este adresa URL a scriptului aflat pe serverul care primește datele formularului: <code><form action="http://www.yahoo.com/cgi-bin/fisier.cgi"></code>	
method	precizează metoda utilizată de browser pentru expedierea datelor formularului	
	get (valoarea implicită) - datele din formular sunt adăugate la adresa URL precizată de atributul action (nu sunt permise cantități mari de date)	post - folosită cel mai des. În acest caz datele sunt expediate separat. Sunt permise cantități mari de date.

Calea prin care informațiile introduse într-un formular pot parveni creatorului paginii este folosirea comenzii **mailto:**


 ``

Majoritatea elementelor unui formular sunt definite cu ajutorul etichetei **<input>**. Aceasta este utilizată împreună cu următoarele atribute:

Atribut	Valoare	Element introdus	Semnificație
type	text	casetă de text	permite introducerea unui șir de caractere pe un singur rând
	radio	buton radio	permite alegerea, la un moment dat, a unei singure variante din mai multe posibile
	checkbox	căsuță de validare	permite selectarea sau deselectarea unei opțiuni
	button	buton de comandă	permite declanșarea unei operații atunci când utilizatorul execută click sau dblick pe suprafața

			acestui
	submit	buton de transmitere	este butonul a cărui activare declanșează operațiunea de trimitere a datelor catre server
	reset	buton de resetare	este butonul a cărui activare readuce controalele din formular la valorile lor inițiale
	image	imagine	permite înlocuirea unui buton submit cu o imagine specificată
	password	casetă de text specială	este similară controlului text, diferențele constând în faptul că datele introduse de utilizator vor fi afișate printr-un caracter "mască" (ex: "***") pentru a oferi un anumit grad de confidențialitate. Este folosit de obicei la introducerea unor parole.
	hidden	câmp ascuns	permite introducerea în formular a unui câmp ascuns
	file	permite expedierea conținutului unui fișier a cărui adresă URL este transmisă prin intermediul atributului value sau poate fi tastată într-un câmp de editare ce apare odată cu formularul sau poate fi selectată prin intermediul unei casete de tip File Upload sau Choose File care apare la apăsarea butonului Browse din formular.  <code><input type="file" name="file"></code>	
name	permite atașarea unui nume fiecărui element al formularului		
value	permite atribuirea unei valori inițiale unui element al formularului		
checked	are rolul de a preseta o anumită opțiune, pe care însă utilizatorul o poate schimba, dacă dorește		
size	setează numărul de caractere al căsuței de text afișate		
maxlength	setează numărul maxim de caractere al căsuței de text afișate		

Cu ajutorul etichetei `<textarea>` `</textarea>` puteți insera în pagină o casetă de text multilinie care permite vizitatorului să introducă un text mai lung, care se poate întinde pe mai multe linii.

 `<textarea name="adresa" rows=2 cols=30>` `</textarea>`

Exemplu: pagina următoare conține elemente de mai multe tipuri încadrate într-un formular unic. Pentru alinierea elementelor utilizate pentru informațiile personale am utilizat un tabel.

Comandati Pizza

Cate pizza doriti?

Mare Medie Mica

Preferinte

Salam Carnati
 Sunca Ciuperci
 Ceapa Masline
 Ardei Branza

Numele

Telefonul

Adresa

```

<form action="mailto:pizza_star@yahoo.com"
      method="post">
<h3>Comandati Pizza</h3>
<p>Cate pizza doriti?
<input name="nrpizza" value="0" size=3 maxlength=3><P>
<input type="radio" name="marime" value="mare" checked>Mare
<input type="radio" name="marime" value="medie">Medie
<input type="radio" name="marime" value="normala">Mica
<h4>Preferinte</h4>
<input type="checkbox" name="topping" value="salam">Salam
<input type="checkbox" name="topping" value="carnati">Carnati<BR>
<input type="checkbox" name="topping" value="sunca">Sunca
<input type="checkbox" name="topping" value="ciuperci">Ciuperci<BR>
<input type="checkbox" name="topping" value="ceapa">Ceapa
<input type="checkbox" name="topping" value="masline">Masline<BR>
<input type="checkbox" name="topping" value="ardei">Ardei
<input type="checkbox" name="topping" value="branza">Branza<p>
<table width="60%">
  <tr><td width="20%">Numele<td><input type="text" name="nume">
  <tr><td width="20%">Telefonul<td><input type="text" name="telefon">
  <tr><td width="20%">Adresa
    <td><textarea name="adresa" rows=2 cols=30></textarea>
</table>
<p><input type="submit" value="Trimite comanda"></form>


```

Etichetele **<select>** și **<option>** permit introducerea într-un formular a unui meniu derulant. Fiecare opțiune care face parte din blocul **<select>** se introduce prin eticheta **<option>**.

Atributele etichetei **<select>**:

Atribut	Semnificație
name	atașează listei un nume (utilizat în perechile "name=value") expediat serverului
size	precizează câte elemente din listă sunt vizibile la un moment dat pe ecran, valoarea implicită fiind 1
multiple	permite selectarea mai multor valori din cadrul unei liste de selecție

Atributele etichetei **<option>**:

Atribut	Semnificație
value	primește ca valoare un text care va fi expediat serverului în perechea "name=value"; dacă acest atribut lipsește, atunci către server va fi expediat textul ce urmează după <option> :  <code><option value="1">Exemplu</option></code> <code><option value="2" selected>Exemplu</option></code>
selected	permite selectarea prestabilită a unui element al listei

Exemplu: pagina următoare conține o listă derulantă.

```

<form action="mailto:adresa@yahoo.com" method="post">
Care este zodia dumneavoastra corespunzatoare<br>
zodiacului chinezesc?<br>
<select name="zodia" size="1">
<option>DRAGON</option>
<option>SARPE</option>
<option>CAL</option>
<option>CAPRA</option>
<option>MAIMUTA</option>
<option>COCOS</option>
<option>CAINE</option>
<option>MISTRET</option>
<option>SOBOLAN</option>
<option>BIVOL</option>
<option>TIGRU</option>
<option>IEPURE</option>
</select>
<input type="submit" value="expediaza">
</form>

```



1.2.4. Evaluare

1. Care dintre următoarele expresii HTML pentru introducerea unui hyperlink, este corectă din punct de vedere sintactic?

- a) `Visual Studio 2008 Express Editions`
- b) `Visual Studio 2008 Express Editions`
- c) `Visual Studio 2008 Express Editions `
- d) `<a>http://www.microsoft.com/express/download/default.aspx `

2. În care dintre codurile HTML următoare chenarul tabelului nu este afișat?

- a) `<table border></table>`
- b) `<table border="0"></table>`
- c) `<table border="-1"></table>`
- d) `<table border=no></table>`

3. Următorul cod HTML generează un tabel cu o linie și două celule.

```

<table bgcolor="white">
<tr bgcolor="red">
<td bgcolor="gray">c1
<td>c2
</table>

```

Ce culoare vor avea cele două celule?

- a) ambele vor fi albe
- b) ambele vor fi roșii

c) c1 va fi albă și c2 gri

d) c1 va fi gri și c2 roșie

4. Cu ce putem completa codul HTML pentru a obține tabelul din figură?

Nume	Limbaie de programare studiate	
Mihai	C#	Java

```
<table border="1">
<tr>
<th>Nume</th>
<th _____>Limbaie de programare studiate</th>
</tr>
<tr>
<td>Mihai</td><td>C#</td>
<td>Java</td>
</tr>
</table>
```

a) rowspan=1

b) colspan=2

c) rowspan=2

d) colspan=1

5. Secvența :

```
<frameset cols=20%,*>
<frame src="A.html">
<frameset rows=40%,*>
<frame src="B.html">
<frame src="C.html">
</frameset>
</frameset>
```

a	b	c	d

are ca rezultat :

6. Câte cadre sunt create prin secvența de cod HTML de mai jos?

```
<frameset rows="60%,*">
<frame src="Despre.html" name="titlu" id="titlu" scrolling="No" noresize>
<frameset cols="150,*">
<frame src="Frame.html" name="continut" id="continut" scrolling="Auto">
<frame name="main">
</frameset>
</frameset>
```

a) 2

b) 5

c) 3

d) 4

7. Fie următoarea secvență de cod:

```
<form action="mailto:adresa@yahoo.com" method="post">
<input type="radio" name="opt" value="inf">Informatica
<input type="radio" name="opt" value="fiz">Fizica
<input type="radio" name="opt" value="chim">Chimia
<input type="radio" name="opt" value="bio">Biologia
</form>
```

Dacă este bifată opțiunea Informatica, care este perechea nume/valoare care va fi transmisă serverului?

a) "opt=Informatica"

c) "inf=Informatica"

b) "opt=inf"

d) "radio=inf"

8. Câte elemente sunt vizibile și câte sunt selectate în lista următoare?

```
<select name="lista" size=2>
<option selected>optiunea1
<option >optiunea2
```

```
<option>optiunea3
<option selected>optiunea4
</select>
```

- a) un element vizibil, două selectate c) patru elemente vizibile, două selectate
b) două elemente vizibile, două selectate d) două elemente vizibile, unul selectat

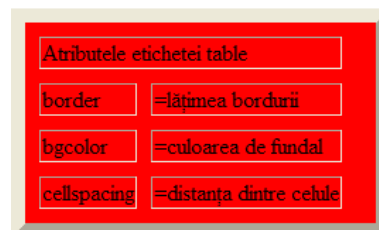
9. Pentru a grupa un număr de butoane radio, acestea trebuie să aibă aceeași valoare pentru atributul:

- a) type b) value c) id d) name

10. Secvența următoare de cod generează:

```
<ul>
<li>TIC</li>
<ul><li>Internet</li>
<li>HTML</li></ul>
<li>Informatica</li></ul>
```

- a) o listă cu 4 elemente
b) 2 liste imbricate cu câte 2 elemente fiecare
c) 2 liste neimbricate cu câte 2 elemente fiecare
d) 4 liste



11. Scrieți codul HTML care să afișeze tabelul:

12. Scrieți codul HTML care să afișeze cadrele de mai jos. Imaginile afișate sunt distincte și nu sunt neapărat identice cu cele din exemplu.



13. Scrieți codul HTML pentru realizarea următorului formular:

14. Pornind de la conținutul fișierului alăturat, îmbunătățiți aspectul paginii parcurgând următorii pași:

- Utilizați o imagine de fundal .
- Introduceți tag-urile speciale de deplasare spre dreapta a titlului.
- Modificați structura tabelului astfel încât primul rând să conțină o singură celulă.
- Alegeți o culoare pentru fundalul celulei.
- Introduceți în celulele tabelului de mai jos câte o imagine.
- Creați câte o legătură de la fiecare imagine la fișierul grafic corespunzător.
- Modificați dimensiunea bordurii tabelului la 5 și centrați tabelul.
- Introduceți în documentul HTML următoarea listă utilizând culoarea preferată pentru font:
 1. HTML-inițiere
 - Formatare
 - paragrafe
 - Imagini
 2. HTML-elemente avansate
 - Liste
 - Tabele
 - Cadre

```
<head>
<title>proiect</title>
</head>
<body>
<h1>Proiect cu tabel si imagini</h1>
<br><br><br>
<table border=2 >
<tr><td>x</td><td>x</td></tr>
<tr><td>x</td><td>x</td></tr>
<tr><td>x</td><td>x</td></tr>
</table>
```

- Formulare

I.3. Utilizarea tehnicii CSS¹⁰ pentru formatarea documentelor Web

De ce CSS?

- Aplicarea stilurilor reprezintă o extindere importantă a posibilităților de stilizare, evitând utilizarea de fișiere grafice mari ce determină încetinirea încărcării paginilor și manipularea lor greoaie.
- Folosind stilurile HTML puteți fi siguri că vizitatorii paginii dumneavoastră vor vedea pagina exact așa cum a fost ea proiectată.


I.3.1. Ce este un stil?

A **stiliza** înseamnă a da unui obiect o formă corectă și expresivă. Un **stil** reprezintă o colecție de valori ale atributelor elementelor unui document, valori care pot fi aplicate întregului document sau doar unei părți din acesta. Aceste atribute pot fi: mărimea, grosimea, tipul și culoarea fontului, marginile, paragrafele și orice altceva ce poate influența aspectul unui element, deci a unei pagini. Gruparea lor în stiluri permite designer-ului să aplice aceeași colecție la diferite părți ale unui document. Folosirea stilurilor reduce considerabil efortul depus atunci când doriți să aduceți modificări aspectului și aranjării elementelor din paginile dumneavoastră. În loc să parcurgeți fiecare document în parte și să faceți modificări asupra fiecărui element, operați modificări numai asupra foii de stiluri care stilizează aceste elemente.

Sintaxa generală a unei declarații de stil este:

```
listă_selectori{proprietatea1:listă_valori1;  
                proprietatea2:listă_valori2;  
                ...  
                proprietatean:listă_valorin;  
                }
```

Selectorii sunt utilizați pentru determinarea elementelor HTML asupra cărora vor fi aplicate stilurile.

 Următorul segment de cod definește proprietățile font, font-size, color și text-align pentru nivelele de titlu H1, H2 și H3:

```
H1,H2,H3 {font-family:Arial,Garamond;text-align:center}11  
H1 {font-size:18px;color:red;background-color:gray}  
H2 {font-size:16px;color:blue}
```

¹⁰ *Cascading Style Sheets* în engleză

¹¹ în scrierea definiției unui stil este posibil să grupăm selectorii în liste, separați prin virgulă

```
H3 {font-size:14px;color:blue}
```

I.3.2. Definiții de stil

Stilurile pot fi aplicate elementelor unui document în trei moduri:

La nivel de element¹²: stilurile sunt incluse ca atribute în cadrul etichetelor HTML din document. Aceasta înseamnă că ele vor afecta doar elementul asupra căruia sunt aplicate. Este o modalitate mai puțin utilizată, deoarece contrazice principiul general al stilurilor, acela de a simplifica și de a face mai lizibil codul documentului HTML.

Încapsulate¹³: stilurile sunt incluse în documentul asupra căruia se aplică, și anume în secțiunea <head> a documentului, prin utilizarea marcajului <style>.

Legate¹⁴: stilurile sunt definite în fișiere separate de documentul HTML. Documentul face apel la foaia de stiluri prin intermediul etichetei <link>. Avantajul folosirii foilor de stiluri externe este dublu. Pe de-o parte, ele se pot aplica la nivelul mai multor documente HTML, realizând astfel o legătură de stil între ele, lucru deosebit de util la construirea unui site. Pe de altă parte, același document poate folosi foi de stiluri diferite, oferind vizitatorului posibilitatea de a opta la un moment dat, pentru unul sau altul dintre ele, în funcție de propriile preferințe.

Includerea stilurilor într-un document HTML:

```
<html>
<head>
```

```
<link rel="stylesheet" type="text/css" href="numefisier.css">
```

legate

```
<style>
H1 {color: #008000; font-weight: bold}
P {font-family: Arial; color: #800080; font-size: 14px}
</style>
```

încapsulate

```
</head>
<body>
```

```
<P style="color: red; font-family: arial; font-weight: bold">Textul din
acest paragraf este scris cu fonturi arial, ingrosate, de culoare rosu
</P>
```

inline

```
</body>
```

¹² *inline* în engleză

¹³ *embedded* în engleză


¹⁴ *linked* în engleză

</html>

1.3.2.1 Definiții de stil inline

Spre deosebire de stilurile încapsulate și de foile de stiluri externe, stilurile **inline** fac parte chiar din corpul documentului HTML. Ele se aplică prin folosirea atributului **style** în asociere cu etichetele HTML standard.

Definițiile de stil inline se aplică numai asupra elementelor incluse între etichetele care au asociat atributul **style**. Din acest motiv, dacă dorim să repetăm în alt loc din cuprinsul documentului aceleași definiții de stil, ele vor trebui scrise din nou, încărcând astfel documentul HTML. Totuși, utilitatea stilurilor **inline** este aceea că fiind definite chiar în cuprinsul documentului, definițiile lor sunt prioritare față de cele din stilurile încapsulate sau externe.

 <P style="color: red; font-family: Arial; font-weight: bold">
Stil inline pentru acest paragraf</P>

Exemplu: pagina următoare conține o definiție de stil pentru al II-lea titlu de nivel 3¹⁵ care nu se aplică și celorlalte titluri de același nivel.

```
<body style="color:red; background-color:gray">
<h3>Titlu de nivel 3 formatat implicit</h3>
<table border="2" cellspacing="2" cellpadding="2">
<tr align="center">
<td></td>
<td></td>
<td></td>
</tr>
<tr><td><h3 style="color: yellow; font-family: Garamond; font-size: 20;
text-decoration: underline">Titlu de nivel 3 formatat inline</h3></td></tr>
</table>
<h3>Titlu de nivel 3 formatat implicit</h3>
</body>
```

Titlu de nivel 3 formatat implicit



Titlu de nivel 3 formatat implicit

1.3.2.2 Definiții de stil încapsulate (interne)

Crearea unui astfel de stil se realizează folosind eticheta **<style> </style>**. Eticheta de stil este plasată în antetul documentului adică în secțiunea **<head>**.

¹⁵ textele cuprinse între etichetele <H3> </H3>

Exemplu: pagina conține o definiție de stil care realizează afișarea tuturor titlurilor de nivel 1 cu caractere bold și culoare gri. Textele incluse între etichetele `<p>` `</p>` vor fi afișate cu fontul arial, de mărime 12 și culoare violet. De asemenea, este creat un stil "stil_text" care poate fi aplicat oricărui text. Prin intermediul său, textul este afișat cu caractere de dimensiuni mai mari și culoare roșie.

```
<head> <style>
  H1 {color: gray; font-weight: bold;}
  P {font-family: Arial; color: #800080; font-size: 12px;}
  .stil_text {font-size: large; color: #F00000;}
</style></head>

<body>
<h3>Stiluri încapsulate (interne)</h3>
<P>Textul din acest paragraf este formatat cu ajutorul stilurilor</P>
Text neformatat
<H1>Titlu formatat</H1>
<span class="stil_text">Acesta este stilul care se aplica fiecarui
text din document</span>
</body>
```

Stiluri încapsulate (interne)

Textul din acest paragraf este formatat cu ajutorul stilurilor

Text neformatat

Titlu formatat

Acesta este stilul care se aplica fiecarui text din document

Exemplu: pagina conține o definiție de stil încapsulat și două de stil inline.

```
<head>
  <style>
    P {font-family: Arial; color:orange; font-size: 14px}
  </style>
</head>
<body>
  <H3 style="color: red; font-family: Western, Verdana">Titlu de nivel 3 formatat
  inline</H3>
  <P>Text formatat cu ajutorul unui stil încapsulat</P>
  <SPAN style="font-family: Garamond; color: teal; font-weight: 600">Text formatat
  inline</SPAN> <br>
</body>
```

Titlu de nivel 3 formatat inline

Text formatat cu ajutorul unui stil încapsulat

Text formatat inline

Atunci când lucrați cu documente HTML deja existente, cărora doriți să le aplicați stiluri **inline**, este recomandat să folosiți etichetele `<div>` și ``. Acestea vă permit să aplicați stilurile fără a afecta codul HTML deja existent sau aspectul paginii în browserele care nu suportă stiluri.

Eticheta `<div>` funcționează asemănător cu eticheta `<p>`, marcând un întreg bloc de conținut, dar fără a genera linii albe între paragrafe.

Eticheta **** este similară cu eticheta **** aplicându-se elementelor dintr-o porțiune mică a documentului, de la câteva cuvinte, la câteva linii.

I.3.2.3 Definiții de stil extern

O foaie de stiluri este un fișier text care conține reguli de stil definite în aceeași manieră ca la stilurile incluse în pagină. Odată creată o foaie de stiluri, ea trebuie salvată cu extensia **.css**.

Apelul foilor de stiluri se poate realiza în două moduri:

- folosind eticheta **<link>**
- folosind funcția **@import**

Metoda importului (**@import**) este puțin mai lentă, fiind posibil să dureze o secundă, două, până se încarcă foaia de stil, timp în care conținutul este afișat fără formatarea designer-ului.

Cel mai folosit mod de apelare a unei foi de stiluri este cu ajutorul etichetei **<LINK>** conform următoarei sintaxe:

```
<LINK rel="stylesheet" href="nume_foai_e_stiluri.css">
```

Exemplu: pagina următoare utilizează foaia de stiluri **stil.css**, prin intermediul etichetei **<LINK>** cu atributul **rel="stylesheet"**. Atributul **href** al etichetei are ca valoare numele¹⁶ foi de stiluri apelate. Am utilizat și proprietatea **background-repeat** pentru introducerea unei imagini pe fundal și repetarea ei doar pe axa Ox.

```

stil.css
body{background-image:url(i4.gif);
  background-repeat:repeat-x;background-color:gray;}
h2{color:yellow}
.semafor{color:red;Font-Family:Arial Black;background-color:yellow}

<head>
<link rel="stylesheet" href="stil.css"></head>
<body topmargin=50>
<h2>Referirea unei foi de stil externe</h2>
<P>Pentru acest document am folosit un fisier de tip <span class=semafor>
foaie de stil</span> in interiorul caruia am definit 3 stiluri:
<ul>
  <li>pentru corpul documentului
  <li>pentru titlurile de nivel 2
  <li>un stil pentru punerea in evidenta a unor pasaje de text, stil numit <span
class=semafor>semafor</span></li></ul>
</body>

```

¹⁶ și adresa relativă, dacă este necesar



Referirea unei foi de stil externe

Pentru acest document am folosit un fisier de tip **foaie de stil** in interiorul caruia am definit 3 stiluri:

- pentru corpul documentului
- pentru titlurile de nivel 2
- un stil pentru punerea in evidenta a unor pasaje de text, stil numit **semafor**

I.3.3. Stiluri în cascadă

Cele trei tipuri de stiluri pot fi combinate în cadrul aceluiași document. Relațiile dintre diferitele tipuri de stiluri realizează efectul de cascadă care dă numele acestei metode.

Browserul rezolvă conflictul dintre definițiile de stiluri respectând următoarea regulă: stilurile inline au cea mai mare prioritate, apoi cele interne și, în cele din urmă, stilurile externe, cu prioritate minimă.

O regulă de stil poate să-și mărească prioritatea dacă este însoțită de declarația "**!important**":

```
P {font-size:12pt!important; font-style:italic }
```

Dacă ne referim la "care dintre cele 3 definiții de stil este mai bună", standarde WEB indică:

- utilizarea foilor **.css** pentru definirea caracteristicilor care se aplică la formatarea părții comune a tuturor paginilor unui document
- utilizarea stilurilor încapsulate pentru definirea caracteristicilor care se aplică la formatarea unei anumite pagini;
- utilizarea stilurilor inline pentru definirea caracteristicilor care se aplică la formatarea unui anumit element.

I.3.4. Clase de stiluri


Acestea permit definirea unui stil general (aplicabil în mai multe locuri în cadrul aceleiași pagini sau în pagini diferite) în vederea inserării lui oriunde este necesar prin intermediul unei simple referiri. Să presupunem că dorim să definim o clasă de stiluri "ftext" (pe care dorim să o aplicăm anumitor porțiuni de text pentru a le face să apară de culoare roșie, aliniate la stânga, având culoarea de fundal gri și mărimea fontului de 14).

```
<style>  
  all.ftext{text-align:left; color:red;}  
</style>
```

Cuvântul standard "**all**" aflat în fața clasei de stiluri "**ftext**" indică faptul că această clasă este aplicabilă tuturor blocurilor de text, atunci când este necesar. Practic clasa de stiluri "ftext" poate fi asociată tuturor tagurilor HTML care operează cu text (ca de exemplu: H2, H3, P, DIV, etc...) utilizând în interiorul fiecărui tag vizat o referire explicită la această clasă:

 `<tag class=ftext>`

Dacă dorim să aplicăm această clasă de stiluri unui titlu de nivel 2, atunci scriem:

 `<H2 class="ftext"> Acest header este aliniat la stanga si are culoarea rosie </H2>`

După cum s-a văzut, pentru apelarea unei clase de stiluri în vederea aplicării sale elementului tag curent se folosește atributul "**class**", având ca valoare numele clasei de stil. Împreună cu specificația "all" din definirea clasei de stiluri, atributul "class" devine un atribut universal, adică va putea fi asociat tuturor tagurilor HTML cărora li se poate aplica.

Dacă dorim ca o clasă de stiluri să fie aplicabilă numai pentru anumite elemente ale documentului (spre exemplu pentru paragrafele de text desemnate prin marcajul de paragraf "p"), atunci în construcția clasei va apărea acest element (de exemplu "p.ftext").

Exemplu: acest exemplu conține o definiție de clasă aplicabilă doar textelor desemnate prin marcajul `<p></p>`.

Acesta este un paragraf neformatat

```
<head>
<style>
  p.ftext{ text-align: left; color: red; background: gray; font-size: 14 pt}
</style>
</head>
<body>
  <p>Acesta este un paragraf neformatat</p> <br>
  <p class="ftext">Acesta este un paragraf formatat cu stilul ftext</p>
</body>
```

Acesta este un paragraf formatat cu stilul ftext

1.3.5. Stiluri identicator

Denumirea stilurilor identicator este dată de modul în care este asociat stilul respectiv unui element, prin intermediul atributului **id**. Definirea unui stil identicator este similară cu a unei clase de stiluri. Vom folosi id-uri doar pentru stilizarea elementelor dintr-o pagina care apar doar o singura dată, altfel, folosirea claselor este recomandată.

 `# nume_stil { descriere }`

Utilizarea stilului identicator se realizează incluzând în interiorul etichetei elementului de text secvența:

 `id = "nume_stil"`

Exemplu: acest exemplu conține două definiții de stil de tip identicator **cap** și **corp**.

```

<head>
  <style>
    #cap{color:red;font-weight:bold}
    #corp{color:black;font-style:italic}
  </style>
</head>
<body id="corp">
  <h3 id="cap">CSS</h3>
  <h4 id="cap" style="color:green">Stiluri identicator</h4>
  <p># nume_stil { descriere }
  <p>id = "nume_stil"
</body>

```

CSS*Stiluri identicator*

nume_stil { descriere }

id = "nume_stil"

I.3.6. Pseudoclasă și pseudoelemente

Pseudoclasă controlează comportamentul dinamic al unor elemente, cum ar fi legăturile. În CSS, o legătură poate avea 5 stări ce corespund la 5 pseudoclasă:

Atribut	Semnificație
:link	descrie starea normală a unei legături
:visited	descrie o legătură vizitată
:hover	descrierea aspectul legăturii când aceasta primește focusul
:active	descrie starea activă a unei legături
:focus	descrie o legătură selectată

Exemplu: în această pagină hiperlegăturile vor fi subliniate și vor avea culoarea roșie. Legăturile vizitate sunt scrise cursiv, cele active sunt afișate cu caractere aldine¹⁷, iar legătura care deține focusul este reprezentată cu majuscule.

```

<head>
  <style>
    a:link{color:red;}
    a:active{font-weight:bold}
    a:visited{font-style:italic}
    a:hover{font-weight:bold;text-transform:uppercase}
  </style>
</head>
<body>
  <h4>In acest curs veti studia despre</h4>
  <ul>
    <li><a href="HTML.html">HTML-tabele, formulare, cadre</a></li>
    <li><a href="CSS.html">CSS-definitii de stil, pozitionarea elementelor</a></li>
    <li><a href="ASPNET.html">ASP.NET</a></li>
  </ul></body>

```

In acest curs veti studia despre

- [HTML-TABELE, FORMULARE, CADRE](#)
- [CSS-definitii de stil, pozitionarea elementelor](#)
- [ASP.NET](#)

¹⁷ italice

Exemplu: pagina următoare utilizează foaia de stiluri **test.css**. În documentul HTML am inclus și o legătură, pentru a exemplifica modul în care foaia de stiluri schimbă culorile legăturii.

```
<head><title></title>
<link rel="stylesheet" href="test.css"></head>
<body>
<H1>Foi de stiluri externe</H1>
Acest exemplu ilustreaza modul in care este inclusa in pagina
o foaie de stiluri externa
<P>Daca vrei sa stii mai multe cauta in
<a href="css.html">Totul despre CSS</a>
</body>
```

Fișierul de stiluri **test.css**



```
BODY
{background-color: #CCFFCC;
font-family: Arial, sans-serif;
color: blue;
padding: 50px, 70px}
A:link {color: #CC9900}
A:visited {color: red}
A:hover {color: #F00F00}
A:active {color: #FF0000}
H1 {color: red;background-color: gray}
```

Foi de stiluri externe

Acest exemplu ilustreaza modul in care este inclusa in pagina o foaie de stiluri externa

Daca vrei sa stii mai multe cauta in [Totul despre CSS](#)

Pseudoelementele controlează aspectul anumitor porțiuni ale unui element, cum ar fi prima linie a unui paragraf sau prima literă a unui text. Astfel, pentru formatarea paragrafelor, avem următoarele pseudoelemente:

-  selector:first-line {descriere} – descrie aspectul primei linii a unui paragraf;
-  selector:first-letter {descriere} – descrie aspectul primei litere a unui paragraf;

unde selector poate descrie orice element de text.

Exemplu: în această pagină am utilizat pseudoelementul **first-letter**.

Pseudoclasele controleaza comportamentul dinamic al unor elemente

Pseudoelementele controlează aspectul anumitor porțiuni ale unui element

```
<head>
<style>
  p: first-letter { font-size: 200%; font-family: GhoticG, Courier; color: red; }
</style>
</head>
<body>
  <p><p: first-letter>P</p: firstletter>seudoclasele controleaza comportamentul
dinamic al unor elemente
</p>
<p><p: first-letter>P</p: first-letter>seudoelementele controlează aspectul anumitor
porțiuni ale unui element</p></body>
```

I.3.7. Stiluri pentru liste

list-style-type

Folosind eticheta **ol** din HTML putem crea liste ordonate sau numerotate. Adăugând comenzi CSS în zona HEAD putem adăuga pe lângă numere și cifre, alte simboluri. Browserul Netscape nu permite asocierea comenzilor CSS decât pentru eticheta **li**.

```
<style>
  li {list-style-type: valoare;}
</style>
```

valoare	semnificație
disc	disc
circle	cerc
square	pătrat
decimal	numere întregi
lower-roman	numere romane, caractere mici (i, ii, iii, iv)
upper-roman	numere romane, caractere mari (I, II, III, IV)
upper-alpha	litere mari (A, B, C, D)
lower-alpha	litere mici (a, b, c, d)
none	nimic

Exemplu: pagina următoare conține o listă ordonată care utilizează ca marcatori literele mari ale alfabetului englez

```
<head>
<style>
  li {list-style-type: upper-alpha; }
</style>
</head>
<body>
  Stiluri CSS:
  <ol>
    <li>stil inline</li>
    <li>stil înglobat</li>
    <li>stil extern</li>
  </ol> </body>
```

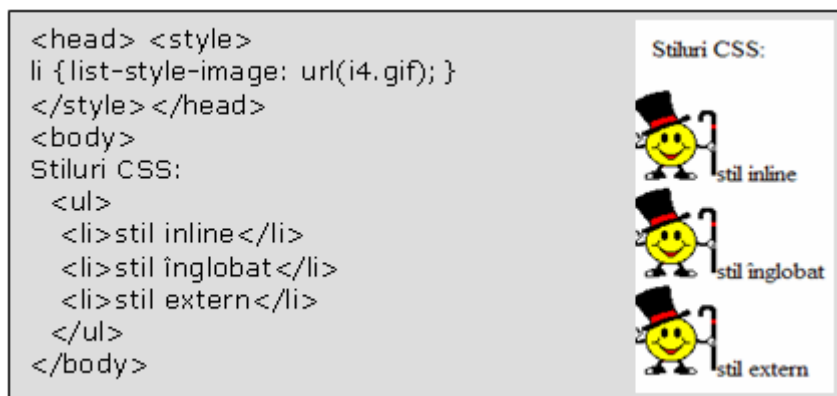
Stiluri CSS:

- A. stil inline
- B. stil înglobat
- C. stil extern

list-style-image

În afara simbolurilor de marcaj prestabilite cunoscute de browser pot fi folosite și imagini prin comanda CSS *list-style-image*. Imaginile sunt introduse prin adresa **url()**.

Exemplu: pagina următoare conține o listă ordonată folosind ca marcaj imaginea **i4.gif**.



Exemplu: pagina următoare conține un meniu structurat ca o listă și stilizat folosind CSS. Pentru a crea un sistem de navigare bazat pe o listă neordonată¹⁸, mai întâi se creează lista, plasând fiecare legătură într-un element ``. Apoi, se încadrează textul cu `<div>` și i se asociază un **id** corespunzător.

¹⁸ marcată cu ajutorul tagului ``

```

<head>
<link rel="stylesheet" type="text/css" href="menu.css"/> </head>
<body>
<div id="menu">
<ul>
<li><a href="#">Principii generale ale proiectarii interfetelor Web </a></li>
<li><a href="#">Visual Web Dev Express 2008</a></li>
<li><a href="#">Modelul client-server. Protocoale de comunicatie </a></li>
<li><a href="#">Interactiunea cu baze de date Web </a></li>
<li><a href="#">Proiectarea si realizarea unei aplicatii Web </a></li>
</ul>
</div>
</body>

```

Fișierul de stiluri **menu.css**

```

#menu {width: 350px;}

#menu ul {list-style:none;margin:0;padding:0;}

#menu li {border-bottom: 1px solid #ED9F9F;}

#menu li a:link, #menu li a:visited {
font-size: 15px;
display: block;
padding: 0.4em 0 0.4em 0.5em;
border-left: 12px solid #711515;
border-right: 1px solid #711515;
background-color: #B51032;
color: #FFFFFF;
text-decoration: none;
font-weight:bold;
}

```

```

#menu li a:visited{ font-style:italic;color:gray }

```

Principii generale ale proiectarii interfetelor Web

Visual Web Dev Express 2008

Modelul client-server. Protocoale de comunicatie

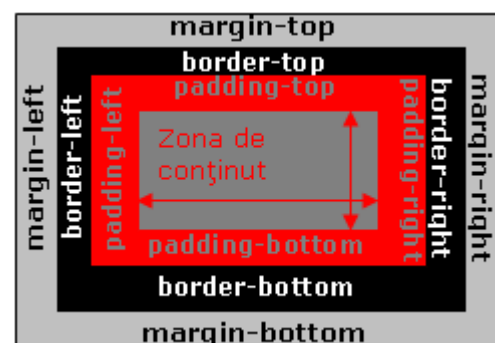
Interactiunea cu baze de date Web

Proiectarea si realizarea unei aplicatii Web

I.3.8. Casete în CSS

Elementele cu care lucrează HTML sunt afișate de browser în interiorul unei zone dreptunghiulare ca în figură alăturată, unde:

- marginea¹⁹ este spațiul exterior chenarului până la celelalte elemente,
- chenarul²⁰ este o bordură care înconjoară elementul,
- completarea²¹ stabilește distanța dintre conținut și chenar,



¹⁹ *margin* în engleză

²⁰ *border* în engleză

²¹ *padding* în engleză

- conținutul include informația utilă (text, tabele, imagini, formulare etc.) .

Exemplu: pagina următoare conține 3 definiții de clase utilizând proprietățile **border-width**, **border-style** și **border-color** și o imagine poziționată la 50px față de latura stângă și 25px față de latura de sus și bordată cu chenar portocaliu.

```

<head>
<style>
.clasa1 {border-width: 4px; border-style: dotted; border-color:orange}
.clasa2 {border-width: 3px; border-style: dashed; border-color:black}
.clasa3 {border-width: 4px; border-style: solid; border-color: green}
img {margin-left: 50px; margin-top: 25px;border-style: solid;border-color:orange}
</style></head>
<body>
<div class="clasa1">border-width</div> <br>
<div class="clasa2">border-style</div> <br>
<div class="clasa3">border-color</div> <br>

</body>
</html>

```



I.3.9. Poziționare în CSS

Poziționarea permite așezarea unui obiect într-un anumit loc folosind coordonatele sale. Totodată obiectele pot fi poziționate pe straturi diferite, unul deasupra celuilalt. O astfel de așezare se poate face utilizând atributul **position**.

Atât poziționarea absolută cât și cea relativă folosesc proprietățile **left** și **top** exprimate în px (pixeli), in (inci), pt (puncte), cm (centimetri).

Poziționarea absolută plasează obiectul în pagină exact în locația data de left și top. Astfel poate fi creat un element liber față de celelalte din pagină. Obiectul poate fi orice, de exemplu text sau imagine.

Exemplu: am aplicat poziționarea absolută etichetei **h3**.

CSS-casete

```

<body>
<h3 style="position: absolute;left: 25px; top: 50px">CSS-casete</h3>
<h3 style="position: absolute;left: 50px; top: 100px">CSS-pozitionare</h3>
</body>

```

CSS-pozitionare

Poziționarea relativă este poziționarea unui element în funcție de elementele anterioare. Un element poate fi deplasat față de altul folosind proprietățile **left** și **top**.

Exemplu: În pagina următoare cuvintele sunt poziționate relativ unul față de celălalt.

```
<head>
<style>
.sus {position: relative; top: -10px; }
.jos {position: relative; top: 10px; }
</style>
</head>
<body>
Acele <span class="sus">cuvinte</span> sunt
<span class="jos">pozitionate relativ</span>
</body>
```

Acele cuvinte sunt pozitionate relativ

Poziționarea tridimensională

Elementele sunt poziționate pe ecran pe o suprafață bidimensională dar pot fi așezate și unul deasupra celuilalt, într-o stivă utilizând un indicativ (index-z) începând cu 0, următorul 1 și tot așa, în continuare. Elementul cu indexul cel mai mare este așezat deasupra.

Exemplu: pagina următoare conține 3 imagini poziționate absolut și suprapuse.

```
<head>
<style>
.poz1 {position: absolute; left: 40px; top: 40px; z-index: 1}
.poz2 {position: absolute; left: 100px; top: 70px; z-index: 3}
.poz3 {position: absolute; left: 60px; top: 120px; z-index: 2}
</style>
</head>
<body>
<div class="poz1"></div>
<div class="poz2"></div>
<div class="poz3"></div>
</body>
```



I.3.10. Notații și unități de măsură

Foile de stil utilizează două tipuri de unități de lungime:

- **relative:** exprimă o dimensiune în raport cu altă dimensiune ,
- **absolute:** exprimă o dimensiune fixă.

Unități de măsură relative:

- **em**-reprezintă lățimea literei 'M' relativă la fontul utilizat²²,
- **ex**- reprezintă înălțimea literei 'X' relativă la fontul utilizat,
- **px**-pixeli (dimensiunea este dependentă de rezoluția calculatorului) .

Unități de măsură absolute:

- **in**-inch (1in=2.54cm),
- **cm**-centimetri,
- **mm**-milimetri,
- **pt**-punct tipografic(1pt=1/72 in).

Culori

Atributul culoare pentru un obiect poate fi specificat printr-un cuvânt cheie (*aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white si yellow*) sau prin intermediul unei specificații numerice RGB. Acestea sunt luate din paleta VGA Windows. Specificarea unei culori în forma hexazecimală RGB se prefixează cu caracterul # și conține șase cifre hexazecimale.

URL²³ este soluția aleasă de World Wide Web Consortium, pentru specificarea unei resurse (unui site sau a unei pagini) în Internet. Sintaxa generală este:

 <protocol>://<nume_DNS>/<nume_local>

unde

- **protocol** este protocolul folosit (de cele mai multe ori HTTP),
- **nume_DNS** este numele domeniului pe care se află resursa,
- **nume_local** este format din calea și numele resursei de pe discul local.

Care este diferența dintre CSS și HTML?

HTML-ul este utilizat pentru **structurarea** conținutului unei pagini, iar CSS-ul este utilizat pentru **formatarea** acestuia.



²² Mărimea celorlalte caractere este ajustată în funcție de acesta

²³ Uniform Resource Locator

I.3.11. Evaluare

1. Care dintre următoarele este o sintaxă corectă CSS?

- a) {body;color:gray}
- b) body {color:gray}
- c) body:color=gray
- d){body:color=gray(body)}

2. Care dintre următoarele expresii CSS aplică proprietatea bold elementului p?

- a) <p style="font-size:bold">
- b) p {text-size:bold}
- c) p {font-weight:bold}
- d) <p style="text-size:bold">

3. Cum definim o bordură dimensionată astfel: top border = 10 px; bottom border = 5 px; left border = 20 px; right border = 1px?

- a) border-width:10px 1px 5px 20px
- b) border-width:10px 5px 20px 1px
- c) border-width:5px 20px 10px 1px
- d) border-width:10px 20px 5px 1px

4. Care din următoarele variante definește un model de stil aplicabil tuturor elementelor **h3** dintr-un document?

- a) <h3 style="...">
- b) <style> h3{..} </style>
- c) <style> .h3{..} </style>
- d) <style> #h3{..} </style>

5. Să considerăm următoarea definiție a unui model de stil în antetul documentului HTML:

```
<STYLE>
  .ocean{...}
</STYLE>
```

Care din următoarele variante definește un paragraf căruia i se aplică stilul respectiv?

- a) <p name="ocean">Am aplicat stilul...
- b) <p class="ocean">Am aplicat stilul...
- c) <p style="ocean">Am aplicat stilul...
- d) <p id="ocean"> Am aplicat stilul...

6. Dacă într-un document HTML apar următoarele definiții de stil:

```
<STYLE>
  .ocean{background-color:gray;color:red;align:center}
  H1{color:yellow;align:left}
</STYLE>
```

...

```
<H1 class ="ocean" style="color:blue">
```

antetul definit mai sus va avea următoarele proprietăți:

- a) background-color:gray;color:yellow;align:center
- b) background-color:gray;color:red;align:left
- c) background-color:gray;color:blue;align:left
- d) background-color:gray;color:blue;align:center

7. Scrieți codul HTML care să afișeze o legătură stilizată ca în figură:

Mai multe despre CSS

8. Scrieți codul HTML care să realizeze pagina alăturată, utilizând următorul fișier extern de stiluri:

Dictionar CSS

- **foaie de stil** consta intr-un sir de reguli
- **regula** este definita de un selector si o declaratie
- **declaratie** este o multime de perechi proprietate:valoare separate prin ;
- **pseudo-clasa** se poate folosi la particularizarea legaturilor, a primei litere, a primului rand etc

```
body { text-decoration:none;color:black;
background-color:gray;font-family:"arial";
font-size: 12pt;font-weight:medium; }
h2 {text-decoration:underline;color:red; }
b {text-decoration:bold;color:#00FF00;
font-family: "courier";font-size:14pt;font-weight:heavy; }
```

9. Completați sursa HTML de mai jos astfel încât listele definite să apară cu mai multe tipuri de marcaje, ca în figură.

```
<head>
<style>
ul.disc {list-style-type: disc}
ul.circle {list-style-type: circle}
ul.square {list-style-type: square}
ul.none {list-style-type: none}
</style>
</head>
<body>
  <ul class="disc">
    <li>Definitii de stil incapsulate</li>
    <li>Definitii de stil externe </li>
    <li>Definitii de stil inline</li>
  </ul>
  ...
</body>
```

- Definitii de stil incapsulate
- Definitii de stil externe
- Definitii de stil inline

- Definitii de stil incapsulate
- Definitii de stil externe
- Definitii de stil inline

- Definitii de stil incapsulate
- Definitii de stil externe
- Definitii de stil inline

Definitii de stil incapsulate
Definitii de stil externe
Definitii de stil inline

10. Adăugați tabelului de mai jos încă o celulă (celula12) formatată ca în figură.

```
<table style="position: absolute;left: 300;top: 50">
<tr>
  <td style="background-color:gray;color: #EEE8AA;border-color:red;
border-width: 3px;border-style:solid;width: 100;height: 100;
text-align:center">celula11</td>
</tr>
</table>
```



II. Mediul de lucru Visual Web Developer Express 2008

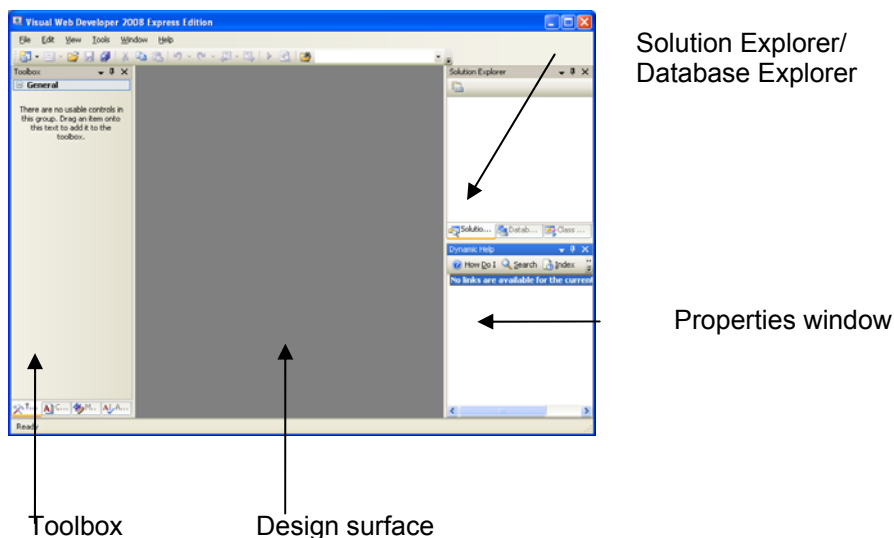
De ce Visual Web Developer?

- este simplu de înțeles și de utilizat
- conține un editor vizual
- conține un editor performant de cod C#/HTML
- conține suport pentru depanarea aplicațiilor Web²⁴
- conține ultima versiune a colecției de clase necesare pentru a putea crea pagini web folosind funcții gata implementate²⁵
- conține un Web Server propriu ce poate rula paginile Web create
- are integrat SQL Server Express, un mediu vizual pentru editarea bazelor de date
- conține facilități ca Master Page²⁶

II.1. Mediul de lucru

Visual Web Developer este un mediu de lucru dezvoltat de Microsoft, specializat pe realizarea site-urilor. Ediția Express este proiectată special pentru amatorii care vor să învețe să folosească Visual Web Developer și tehnologii asemănătoare, fără să aibă îndemânări în utilizarea instrumentelor folosite de programatorii profesioniști; ea conține instrumente de dezvoltare moderne, ușor de înțeles și ușor de folosit, de dimensiuni mici, cu interfețe de lucru simplificate dar în același timp beneficiare ale trăsăturilor clasice ale Visual Studio și ale noilor funcționalități implementate în .NET Framework .

Visual Web Developer se lansează printr-un simplu click pe butonul Start și alegerea opțiunii *Visual Web Developer 2008 Express Edition* din All Programs.



În figură pot fi vizualizate următoarele panouri:

²⁴ **debug** în engleză

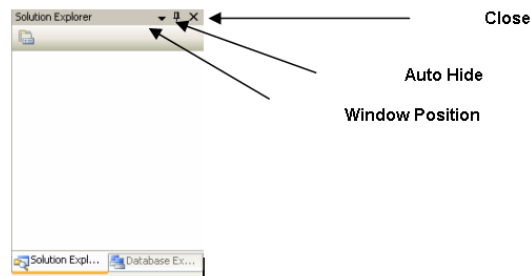
²⁵ .NET FRAMEWORK 3.5

²⁶ un **Master Page** este un formular WEB standard care acționează ca un tipar pentru paginile unui site

- **Toolbox:** când deschideți o pagină sau alt item pentru editare, Toolbox vă oferă instrumentele ce vă permit adăugarea de noi componente²⁷ paginii.
- **Design Surface:** este panoul de editare a paginilor .
- **Solution Explorer/Database Explorer:** Fiecare site Web pe care îl creați este organizat ca un grup de foldere care apare în **Solution Explorer**. Orice bază de date pe care o creați pentru site-ul dumneavoastră apare în **Database Explorer**. Pentru comutarea între cele două programe Explorer se folosește tabul din josul panoului.
- **Properties:** afișează proprietățile asociate obiectului sau paginii cu care se lucrează.

II.2. Cum manevrăm panourile

Panourile pot fi mutate, redimensionate, afișate sau ascunse. Dacă aveți două sau mai multe panouri suprapuse la marginea ecranului, puteți face panoul de la bază mai lung sau mai scurt deplasându-i marginea superioară .



Așa cum arată figura de mai sus, bara de titlu a unui panou conține trei butoane intitulate *Window Position*, *Auto Hide* și *Close*. Selectând *Window Position* puteți accesa următoarele opțiuni ale meniului în cascadă :

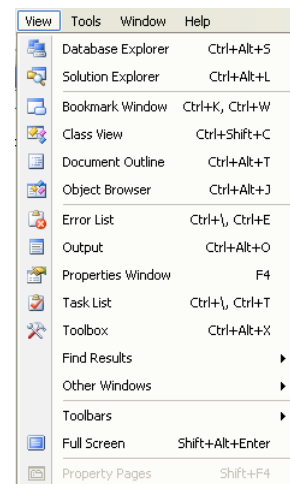
- **Floating:** transformă panoul într-o fereastră flotantă care se poate muta și redimensiona.
- **Dockable:** ancorează panoul.
- **Tabbed Document:** mută panoul în zona de editare, identificată printr-un tab în partea sa superioară. Accesarea tab-ului face panoul vizibil. Pentru a reancora panoul în fereastra programului se dă click dreapta pe tab și se alege *Dockable*.
- **Auto Hide:** convertește panourile deschise în panouri ascunse de-a lungul marginii programului.
- **Hide:** ascunde panoul.

Cu atâtea panouri opționale și atâtea moduri de a muta și a redimensiona lucrurile, este ușor să faceți o mare dezordine în fereastra programului. Pentru a aduce totul la normal, trebuie să alegeți **Window** → **Reset Window Layout** din bara de meniu.

²⁷ controale

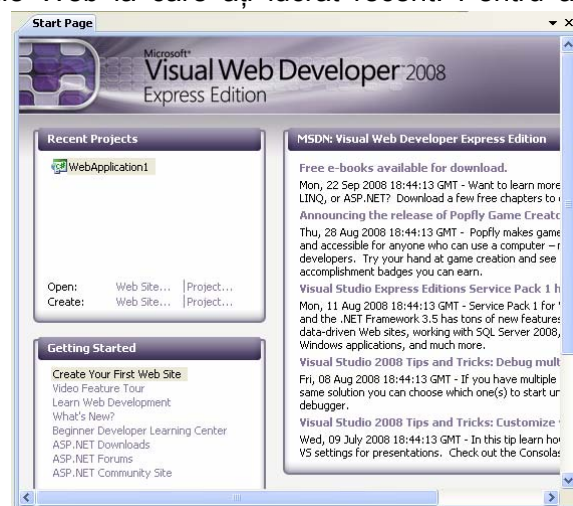
II.3. Meniul View

Opțiunea *View* din bara de meniu oferă acces la toate panourile²⁸ opționale. Dacă închideți un panou folosind butonul *Close*, acesta poate fi făcut din nou vizibil accesând numele lui din meniul *View*. Unele opțiuni ale meniului *View*, cum ar fi *Object Browser* și *Error List*, vor avea un rol semnificativ atunci când veți construi efectiv o pagină. Meniul *View* oferă și opțiunea *Toolbars*, opțiune pe care o puteți folosi pentru a face vizibile sau a ascunde diferite bare de instrumente.



II.4. Pagina de start

Pagina de start apare de fiecare dată când deschideți Visual Web Developer. Sub Recent Projects veți observa o listă cu site-urile Web la care ați lucrat recent. Pentru a deschide unul dintre site-uri selectați numele. Pagina de start nu conține nimic din ce este necesar pentru a construi un site, ci doar link-uri către cărți electronice online gratuite și MSDN Library for Visual Studio Express Editions. După ce ați creat sau ați deschis un site Web, pagina de start dispăre. Dacă vă răzgândiți și vreți să aduceți pagina de start înapoi pe ecran trebuie să alegeți, View→Other Windows→ Start Page din meniul View.



II.5. Publicarea unui site web

Construcția unui site pe propriul calculator e doar primul pas. Dacă doriți ca acesta să poată fi accesat de public trebuie să obțineți un domeniu și apoi să publicați site-ul pe un server aflat la adresa domeniului. Compania care vă asigură spațiul pentru publicarea site-ului este de obicei numită hosting service sau hosting provider. Serviciile de hosting care suportă tehnologiile specifice pe care le folosește Visual Web Developer se numesc ASP.NET²⁹ 2.0 Hosters. Dacă site-ul are și o bază de date în spate, atunci veți avea nevoie de un serviciu de hosting care suportă ASP.NET 2.0 și SQL Server 2005.

²⁸ numite și *ferestre* pentru că pot fi flotante

²⁹ **ASP.NET** este o tehnologie Microsoft pentru crearea de aplicații web și servicii web.

Observații:

Există variante de hosting gratuit, unde vă puteți instala propriul site ASP.NET fără să vă coste nimic (căutați pe www.live.com "asp.net free hosting").

II.6. Compilarea dinamică a site-ului

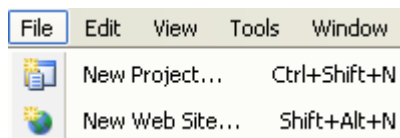
Începând cu ASP.NET 2.0, există posibilitatea de a crea site-uri web în care compilarea se face dinamic pentru o pagină, atunci când pagina respectivă este cerută de utilizator. Cealaltă opțiune ar fi de a avea un *assembly*³⁰. Pagina compilată ține cont de un *timestamp*³¹ asociat fișierului sursă, astfel încât după fiecare modificare a acestuia, pagina se recompilază la prima cerere. Avantajele acestei opțiuni de instalare a site-urilor ASP.NET sunt:

- Nu este nevoie să recompilăm întreaga aplicație atunci când se modifică doar o pagină a acesteia.
- Paginile care conțin erori de compilare nu opresc execuția întregului site cum s-ar întâmpla dacă în loc de pagini compilate dinamic am avea un *assembly*.

II.7. Web site/Web project

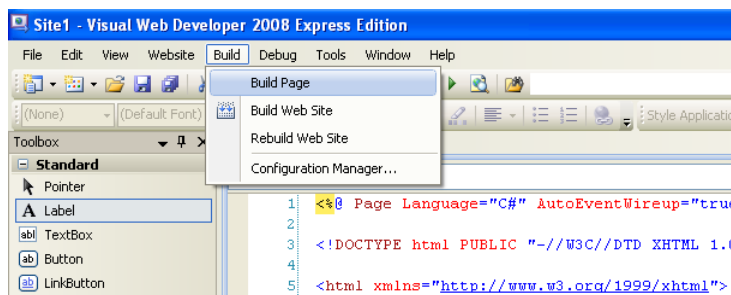
Visual Web Developer Express dispune de două variante diferite de a crea un site web:

- Web Site (File -> New Web Site)
- Web Application Project.



Câteva diferențe dintre cele două variante:

- Web Site-ul compilează dinamic paginile; Web Application Project crează un *assembly*. Atunci când creem un Web Site, avem și posibilitate de a face *build* pe o singură pagină și nu pe tot site-ul.

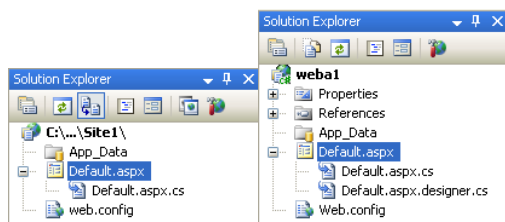


- Web site-ul generează dinamic fișierele care conțin cod generat automat de Visual Studio (cel cu definițiile controalelor); Web Application Project crează fișierul respectiv pe disc, cu extensia *.designer.cs*. Un beneficiu al variantei a doua este timpul mai mic necesar

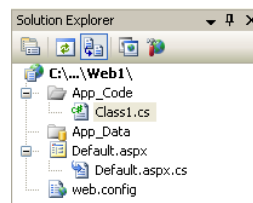
³⁰ un .dll

³¹ timp de creare.

unui *build*.



- Atunci când creem o clasă în aplicația noastră, dacă lucrăm cu varianta Web Site clasa respectivă va fi implicit mutată într-un folder special numit *App_Code*. Asta pentru că la build nu rezultă un assembly, ci se compilează dinamic paginile, iar codul care nu ține de pagini este căutat special în acel folder. Dacă lucrăm cu Web Application Project, putem să creem propria noastră structură de foldere.



Observații:

Dacă ne referim la "care dintre cele 2 variante este mai bună (Web Site sau Web Application Project)" recomandăm ca pentru aplicațiile cu mult cod și o durată de implementare mare să se folosească Web Application Project, iar pentru cele mici Web Site.

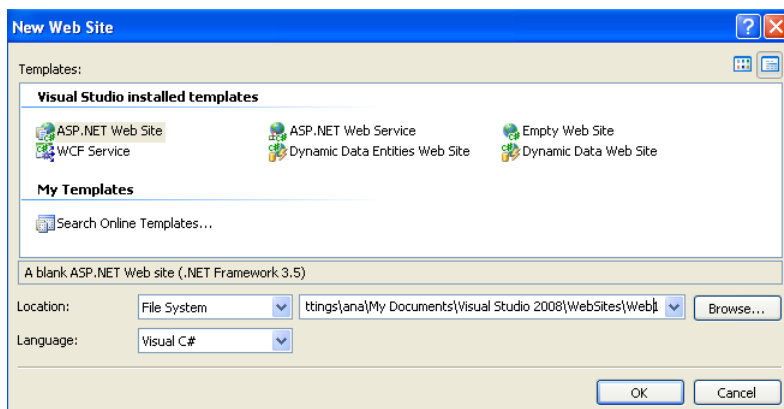
II.8. Crearea unui site web

Primul lucru pe care îl aveți de făcut în Visual Web Developer este să deschideți sau să creați un nou Web Site. Din punctul de vedere al Visual Web Developer, un site Web este un director care conține mai multe subdirectoare și fișiere. Spre deosebire de un un site Web real, cel pe care îl creați va fi stocat pe hard-disk-ul calculatorului dumneavoastră.

Pentru a crea un site Web nou care utilizează C# ca

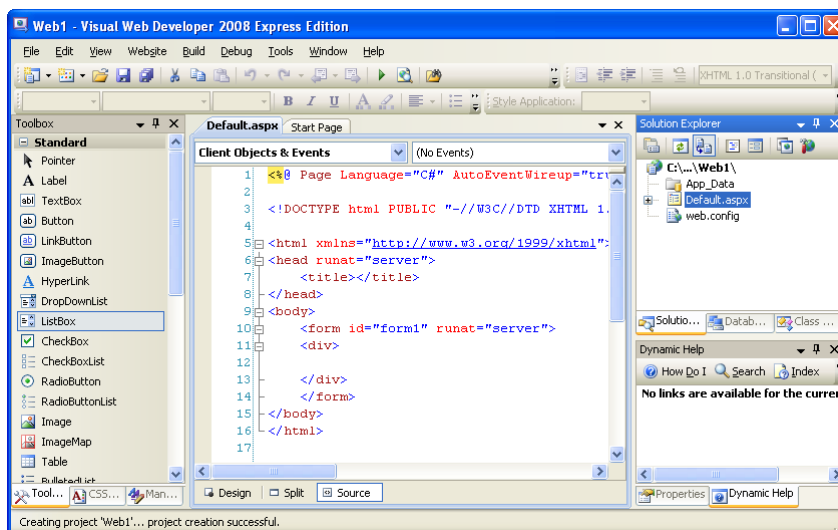
limbaj de programare implicit, se urmăresc pașii:

- De la bara de meniu se selectează *File*→*New Web Site*. Căsuța de dialog va apărea ca în figură.
- Se selectează *ASP.NET Web Site*.
- Din lista Location, se alege *File System*.



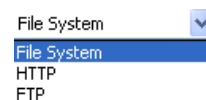
- Din lista Language, se alege limbajul C#.
- Opțional, se schimbă numele și locația site-ului. În exemplul din figură, noul site a fost denumit Web1.
- Se apasă OK.

Crearea site-ului Web va dura câteva secunde. Când site-ul s-a încărcat, va fi afișat un fișier *Default.aspx*, care este pagina principală a site-ului. Veți observa și un folder *App_Data*, în care va fi stocată o posibilă bază de date.





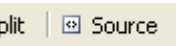
II.9. Despre opțiunea location(File System, HTTP, FTP)

Opțiunea *Location* din fereastra de creare a unui nou site Web vă oferă două alternative la File System pentru a vă stoca site-ul: **FTP** sau **HTTP**.



FTP este prescurtarea de la **Hypertext Transfer Protocol**³² și este un protocol standard de Internet pentru transferarea fișierelor. Veți alege această opțiune dacă doriți să creați site-ul pe un server Web la distanță, server care permite încărcare FTP. Dar nu este neapărat nevoie să creați site-ul pe un calculator la distanță. Este mai simplu să creați un site local (pe calculatorul personal), și apoi când este terminat să-l transferați pe un server.

HTTP este prescurtarea de la **File Transfer Protocol**³³ și este metoda standard pentru transferul datelor în World Wide Web. Puteți folosi această opțiune pentru a crea site-ul pe un server la distanță, chiar și pe calculatorul personal dacă aveți IIS³⁴ instalat, dar pentru că nu este necesar și este dificil de efectuat, vom rămâne cu opțiunea *File System* selectată.

Pagina afișată de Visual Studio ne prezintă codul care va fi trimis spre browser. Aceasta poate fi abordată și în mod *Designer*³⁵, selectând tab-ul  Design  Split  Source

³² Protocol de Transfer al Hypertext-ului

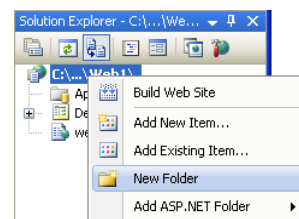
³³ Protocol de Transfer al Fișierelor

³⁴ **Internet Information Services**, în engleză

³⁵ în mod grafic

II.10. Crearea și utilizarea directoarelor³⁶

Puteți folosi directoare pentru a organiza paginile și alte componente din site-ul tău Web în aproape același fel în care folosiți directoarele în Windows pentru a organiza fișierele. Pentru a crea un director, urmați următorii pași:

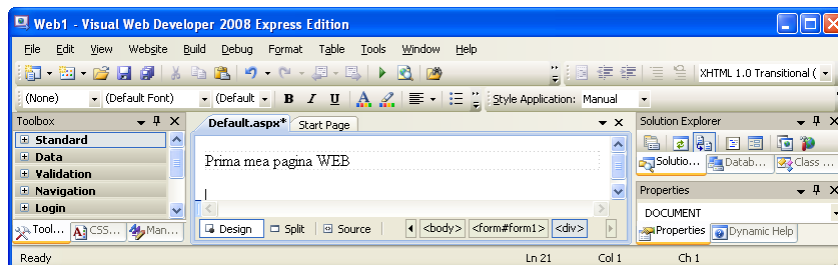


- selectați panoul Solution Explorer.
- faceți click-dreapta pe numele site-ului din partea de sus a ferestrei *Solution Explorer* și alegeți *New Folder*, ca în exemplu
- tastați un nume pentru folder, apoi apăsați Enter.

II.11. Editarea paginilor

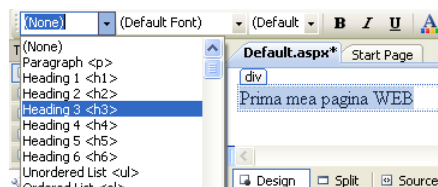
Pentru a edita o pagină existentă în Visual Web Developer, este nevoie mai întâi să deschideți pagina pentru a fi afișată pe suprafața de Design. Pentru a deschide o pagină, se face dublu-click pe pictograma ei din Solution Explorer.

Să încercăm să edităm pagina în mod Design. Vom tasta Prima mea pagină WEB.



Editarea textului în modul Design este foarte asemănătoare cu editarea în Microsoft Word, sau orice alt editor de text: selectați pagina, poziționați cursorul, apoi tastați textul. Toate instrumentele și metodele standard de editare a textului funcționează în suprafața de Design. De exemplu puteți șterge textul cu tastele *Backspace* și *Delete*, puteți selecta textul cu mouse-ul sau cu tastele de navigare, puteți copia și lipi text din/pe suprafața de Design.

Formatarea textului funcționează la fel în Visual Web Developer ca în orice alt program de editare a textului. În exemplul din dreapta am selectat textul și din bara de formatare am ales opțiunea Heading 3.

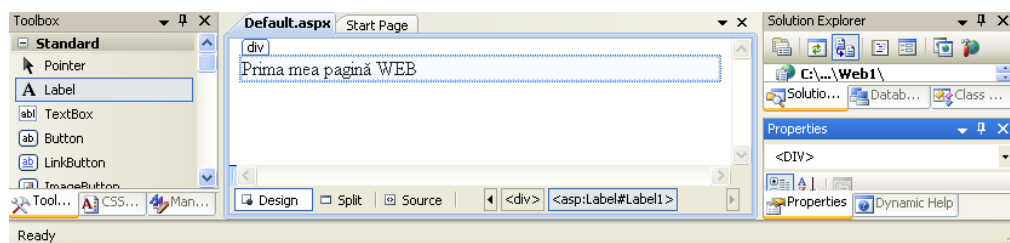


³⁶ *foldere-lor*, în engleză

II.12. Schimbarea proprietăților

Pagina Web lucrează cu *obiecte*. Pentru a vedea, și poate a schimba proprietățile unui obiect, selectați obiectul (click) și vizualizați panoul *Properties*.

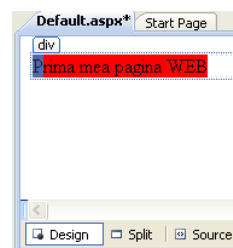
Vom adăuga din *Toolbox* un control *Label* (Label1) schimbându-i proprietatea *Text* în „Prima mea pagină WEB”.



Lista de proprietăți a unui obiect oferă câteva opțiuni pentru modificarea obiectului, dar nu pe toate. Orice opțiune care se leagă de felul în care arată obiectul pe în pagină, poate fi schimbată și cu ajutorul Style Builder sau CSS.

II.13. Salvarea modificărilor

Imediat după ce începeți să editați o pagină, veți observa că numele paginii este scris cu bold și se termină cu '**', ca în exemplul din figura alăturată. Caracterul '**' din dreapta titlului marchează faptul că ați modificat pagina de la ultima salvare. Pentru a salva pagina cu schimbările recente, puteți folosi oricare din metodele: butonul *Save* din bara de unelte, *Ctrl+S*, *File* → *Save numele paginii* din meniul.

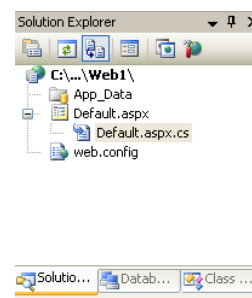


Observații:

Apăsând butonul *Save All*, poți salva toate paginile deschise.

II.14. Despre fișierele cod

Multe din paginile *.aspx* au un fișier cod în spate, cod de programare care definește comportamentul paginii. Codul din aceste fișiere va fi scris în limbajul de programare pe care îl alegeți atunci când creați site-ul Web, în cazul nostru *C#*. În *Solution Explorer*, orice pagină care are un fișier cod în spate prezintă un semn + lângă iconița sa, sau un semn minus cu o iconiță pentru fișierul cod din spatele ei.

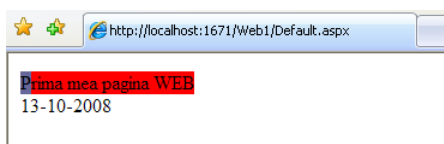


Numele paginii de cod este același cu numele paginii .aspx, dar cu o extensie .cs³⁷ adăugată, cum poate fi văzut în figură.

Mergem în zona de cod (*click dreapta* pe numele fișierului .aspx și alegem *View Code*), în care se implementează codul care dă caracterul dinamic al paginii și modificăm metoda *Page_Load()* ca mai jos:

```
protected void Page_Load(object sender, EventArgs e)
{
    int an = DateTime.Now.Year;
    int luna = DateTime.Now.Month;
    int zi = DateTime.Now.Day;
    Label1.Text = Label1.Text + " " + zi + "-" + luna + "-" + an;
}
```

Ce se vede în browser?




Observații:

Fișierul cod este locul unde se află *partea de programare* din spatele paginii Web.

II.15. Vizualizarea Paginilor într-un Browser Web

Modul de vizualizare Design vă oferă o idee despre cum va arăta pagina în browser. Dar nu prezintă întotdeauna o imagine exactă a site-ului. Pentru a testa pagina dumneavoastră, deschideți pagina într-un browser Web.

Pentru a vizualiza, într-un browser, pagina la care lucrați în modurile de vizualizare *Design* sau *Source* în Visual Web Developer, puteți folosi oricare dintre metodele următoare:

- Click dreapta pe orice spațiu gol din pagina și alegeți *View in Browser*.
- Click pe butonul View in Browser aflat în bara de instrumente 
- Tastați Ctrl+F5

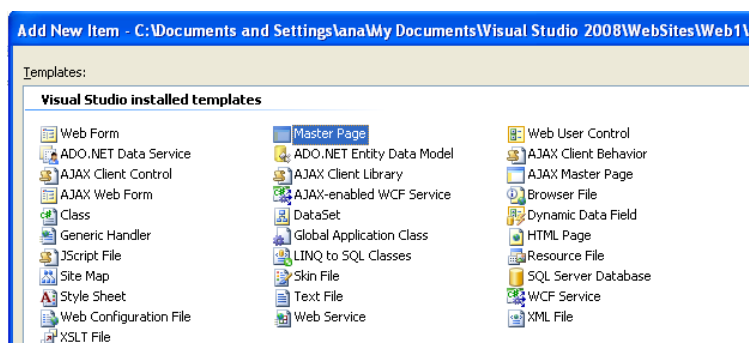
II.16. Master pages

Este foarte indicat pentru un site ca paginile sale să aibă același aspect vizual, adică să urmărească același tipar³⁸. ASP.NET, încă de la versiunea 2.0, ne pune la dispoziție paginile numite "Master Pages", care pot fi utilizate ca un template de către paginile propriu zise ale aplicației. "Master pages" nu sunt pagini care au conținut, în sensul că ele nu pot fi

³⁷ în cazul în care ați ales limbajul C#

³⁸ *template*, în engleză

vizualizate direct, ci doar "moștenite" din punct de vedere al interfeței de către paginile cu conținut.



II.17. Servere Web³⁹ în Visual Web Developer

Pentru a testa și a rula o aplicație Web ASP.NET, veți avea nevoie de un server Web. Serverul Web pentru sistemele de operare Windows este IIS, care include serverul FTP, serverul de e-mail SMTP⁴⁰ și alte facilități. Pentru a rula IIS, trebuie să lucrați pe o versiune Windows, care este capabilă să funcționeze ca un server de rețea.

În Windows 2000 Server și în versiuni anterioare, IIS este instalat ca o parte a sistemului de operare. În Windows XP și Windows Server 2003, IIS trebuie instalat separat; puteți face acest lucru folosind opțiunea *Add/Remove Windows Components* → *Add or Remove Programs* din Control Panel.

Observații:

Puteți avea probleme cu IIS din următoarele motive:

- Lucrați cu pagini ASP.NET pe un sistem de operare Windows XP Home Edition, care nu suportă IIS
- Nu vreți să aveți un server Web pe computerul dumneavoastră din motive de securitate. Rularea unui server Web precum IIS presupune câțiva pași în plus pentru a securiza serverul și a fi la zi cu ultimele update-uri de securitate.

II.18. ASP.NET Development Server


Dacă nu puteți folosi IIS ca server web, puteți testa paginile dumneavoastră ASP.NET folosind **ASP.NET Development Server**. ASP.NET Development Server este integrat în Visual Web Developer și este un server Web care rulează local pe sistemele de operare Windows, incluzând Windows XP Home Edition. Este construit special pentru a rula pagini ASP.NET doar pentru calculatorul pe care lucrați. ASP.NET Development Server vă pune la dispoziție o foarte bună și simplă metodă de a vă testa paginile local înainte de a le publica și a le pune pe un server IIS.

³⁹ Un calculator conectat la Internet care furnizează clienților, la cerere, diverse resurse web

⁴⁰ Simple Mail Transfer Protocol

ASP.NET Development Server funcționează numai cu pagini individuale și nu include toate facilitățile IIS. De exemplu, ASP.NET Development Server nu suportă un server de mail SMTP. Dacă aplicația dumneavoastră Web include trimiterea de mesaje e-mail, trebuie să aveți acces la serverul virtual IIS SMTP pentru a testa e-mail, deoarece ASP.NET Development Server nu poate trimite mesaje e-mail.

II.19. Rularea Serverului integrat

ASP.NET Development Server este instalat automat odată cu Visual Web Developer. Dacă lucrați la un site Web bazat pe un sistem de fișiere, Visual Web Developer folosește automat Visual Web Developer pentru a rula paginile. Implicit, serverul Web utilizează un port aleatoriu. De exemplu, dacă testați o pagină numită `ex1.aspx`, când o rulați pe ASP.NET Development Server, URL-ul poate arăta așa: 

Dacă vreți să rulați ASP.NET Development Server pe un port specific, puteți configura serverul pentru a face asta.

Observații:

Visual Web Developer nu poate garanta că portul pe care îl specificați va putea fi folosit atunci când veți rula site-ul dumneavoastră Web.

II.20. Securitate în ASP.NET Development Server

O diferență importantă între ASP.NET Development Server și IIS o reprezintă securitatea în fiecare dintre cele două servere.

Atunci când rulați o pagină folosind ASP.NET Development Server, pagina rulează în contextul de securitate a contului de utilizator pe care îl folosiți. De exemplu, atunci când rulați o pagină cu ajutorul ASP.NET Development Server folosind un cont de utilizator cu nivel de administrator, și pagina va avea privilegiile de administrator.

În IIS, ASP.NET rulează implicit în contextul userului special (ASPNET sau NETWORK SERVICES) care are privilegiile limitate, ceea ce restrânge accesul la resursele de pe alt calculator.

Dacă rulați codul dintr-o pagină simplă ASP.NET, diferența nu este foarte importantă. Diferențele apar atunci când:

- sunt accesate resurse cum ar fi: fișiere care nu sunt pagini Web, regiștri Windows etc.,
- este accesată o bază de date,
- sunt accesate resurse protejate.

Observații:

Chiar dacă utilizați ASP.NET Development Server pentru a testa paginile care sunt funcționale, ar trebui să le testați din nou după ce le-ați publicat pe un server Web care rulează IIS.

III. Limbajul de scripting server-side ASP.NET

III.1. Structura unei pagini ASP.NET

Orice pagină ASP.NET presupune două componente: **partea de interfață** (controalele care vor fi vizualizate de către utilizator) și **codul .NET** care va fi executat pe server atunci când se face o cerere către pagina respectivă (numit și *code behind*). Codul poate fi scris folosind limbajul C# sau Visual Basic .NET.

Script-urile ASP.NET sunt fișiere text cu extensia *.aspx*, care conțin controale HTML sau ASP. Codul C# sau VB.NET asociat paginii sau obiectelor din pagină poate fi plasat direct în fișierul cu extensia *.aspx*, sau într-un fișier separat *<nume_pagină>.aspx.cs* (pentru C#) sau *<nume_pagină>.aspx.vb* (pentru VB.NET). Pentru a evita așa-numitul „spaghetti code”, unde partea care descrie interfața este intercalată cu instrucțiunile care asigură funcționalitatea, este de preferat folosirea unui fișier separat pentru a reține codul.

Observație. În cadrul acestui manual, toate exemplele vor fi prezentate folosind limbajul C#.

În cazul în care se găsește direct în pagină, codul este cuprins între tag-urile `<script>` `</script>`, cu atributul **runat = "server"**. Iată o prima pagina web simplă:

Exemplul 3.1 first.aspx

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        output.InnerText = "Hello World!";
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>First page</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <p id = "output" runat="server"/>
            </div>
        </form>
    </body>
</html>
```

În exemplul de mai sus, la încărcarea paginii web, va fi afișat mesajul *Hello world*. Codul C# asociat, este scris în cadrul paginii .aspx.

Observații:

În Visual Web Developer, pentru a separa codul într-un fișier separat, la adăugarea unei pagini noi în proiectul curent, se bifează opțiunea „Place code in separate file”:

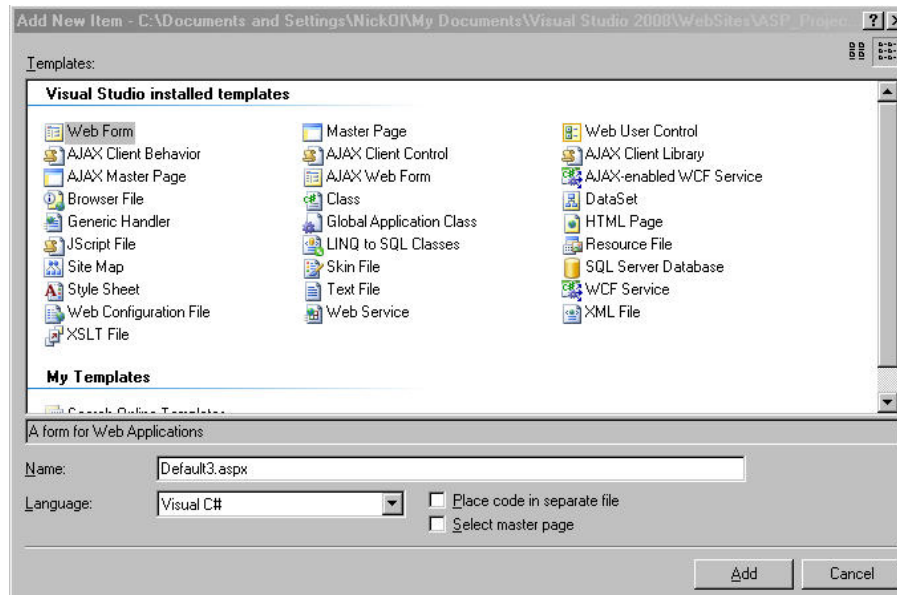


Figura 3.1 Adăugarea unei noi pagini web într-un proiect

Aceeași aplicație poate fi realizată scriind codul C# într-un fișier separat având extensia .aspx.cs:

first.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="first.aspx.cs" Inherits="first1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>First Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<p id = "output" runat="server"/>
</div>
</form>
</body>
</html>
```



```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class first : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        output.InnerText = "Hello World !";
    }
}
```

first.aspx.cs Exemplul 3.2

Observații:

Cu excepția liniei `output.InnerText = "Hello World !";`, codul de mai sus este generat automat de Visual Web Developer.

Orice pagină web .aspx conține o secțiune de directive, care descrie modul în care pagina este procesată de către server. Această secțiune este cuprinsă între tag-urile `<%@` și `%>` și precizează limbajul utilizat pentru scrierea codului și numele fișierului în care este reținut codul (atunci când este cazul):

```
<%@ Page Language="C#" CodeFile="first.aspx.cs" Inherits="_first"%>
```

III.1.1. Controale ASP.NET

Pentru a realiza layout-ul unei pagini web, ASP.NET pune la dispoziția programatorului o serie de controale predefinite, precum și posibilitatea definirii unor controale proprii. Controalele ASP.NET se mai numesc și *controale server* deoarece pot fi programate, prin intermediul unui cod server-side, să răspundă la anumite evenimente din pagină. Pentru a putea fi accesate prin intermediul codului, aceste controale trebuie să conțină atributele `id` și `runat="server"`.

Există două tipuri de controale server: Web și Html. Controalele server HTML corespund controalelor standard HTML, și pot fi create prin adăugarea atributului `runat="server"` unui tag HTML. Pentru a putea identifica un control Html în codul server-side, trebuie să-i atașăm atributul „id”: `<p id = "output" runat="server"/>`.

Când scrieți cod server-side pentru controalele server HTML de tip input (text, button, checkbox, radioButton) puteți folosi proprietatea *value* pentru a accesa valoarea introdusă de utilizator. Proprietatea *InnerText* reprezintă textul cuprins între tag-urile de deschidere respectiv închidere ale unui control Html, iar proprietatea *InnerHtml* reprezintă marcajele Html cuprinse între tag-urile de deschidere respectiv închidere ale controlului Html.

Observații:

În Visual Web Developer, puteți adăuga controale standard HTML din modul design, folosind fereastra ToolBox secțiunea HTML.

Controalele server Web oferă mai multe funcționalități programabile decât cele HTML. Aceste controale pot corespunde mai multor tag-uri HTML, și pot include cod javascript. Sunt cuprinse între tag-urile `<asp>` `</asp>`. De exemplu un control Web pentru introducerea datelor de către utilizator este textbox: `<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>`

III.1.2. Ciclul de viață al unei pagini web.

Paginile ASP.NET rulează pe server-ul web Microsoft IIS. În urma prelucrării pe server, rezultă o pagină web Html, care este trimisă către browser.

Ciclul de viață al unei pagini web are următorii pași:

1. User-ul scrie adresa unei pagini web. Acest lucru înseamnă o cerere a browser-ului web către server, prin intermediul metodei HTTP GET. Pe server, pagina rulează pentru prima dată, executându-se și codul C# existent.
2. Rezultatul este o pagină Html care este trimisă browser-ului.
3. User-ul poate introduce date sau apăsa un buton, și pagina este trimisă înapoi server-ului. Dacă utilizatorul accesează un link, se încarcă o altă pagină, și nici o prelucrare nu este efectuată de pagina inițială.
4. Pagina este trimisă către browser prin intermediul metodei HTTP POST. În ASP.NET această acțiune se numește *PostBack*.
5. Pe server-ul web, pagina rulează din nou, și sunt prelucrate informațiile introduse de către utilizator în browser.
6. Rezultatul este trimis browser-ului, și astfel se reia ciclul.

Procesul de prelucrare a paginii web de către server este împărțit în mai multe etape. Fiecare etapă corespunde unui eveniment asociat paginii. Printre cele mai importante evenimente asociate paginii sunt:

Aplicația 2. Folosind Visual Web Developer și controale server HTML, creați o pagină web care să conțină o casetă de text și un buton. La apăsarea butonului, mesajul introdus în caseta de text trebuie afișat în pagină pe centru, cu fontul arial, mărimea de 14, colorat cu albastru.

Rezolvare: În fișierul .aspx vom adăuga cele două controale Html, împreună cu atributul `runat="server"`. De asemenea, în codul html vom adăuga un tag html `<div>`. În fișierul care va conține codul C#, vom modifica proprietatea `InnerHTML` a tag-ului `<div>`.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>Input</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <input id="mesaj" type="text" runat="server"/>
        <input id="Button1" type="button" value="Trimite" runat="server"
onserverclick="afisare"/><br />
        <div id = "output" runat = "server"></div>
      </div>
    </form></body></html>
```

Fișierul .aspx

```
public partial class _Default : System.Web.UI.Page
{
    protected void afisare(object sender, EventArgs e)
    {
        output.InnerHtml = "<center><b><i><font face='arial' color = 'blue' size
= '14'>";
        output.InnerHtml += mesaj.Value;
        output.InnerHtml += "</font></i></b></center>";
    }
}
```

Fișierul .aspx.cs

Aplicația 3. Folosind Visual Web Developer și controale server HTML, creați o pagină web care să conțină un link și un buton. La apăsarea butonului, link-ul trebuie să-și modifice url-ul și să fie aliniat pe centrul paginii. De asemenea, culoarea de fundal a paginii trebuie să se modifice.

Rezolvare: Pentru a modifica atributele unui control server Html, se poate folosi în codul C# proprietatea `Attributes` asociată unui control Html. Această proprietate este un vector asociativ atribut/valoare. Vom asocia atributele `id` respectiv `runat` tag-urilor `body`, `div`,

a, input, iar în codul C# vom modifica atributele bgcolor, align respectiv href, prin intermediul proprietății *Attributes*.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
    Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server"> <title>Attribute</title>
  </head>
  <body id="mainBody" runat="server">
    <form id="form1" runat="server">
      <div id="mainDiv" runat="server">
        <a href = "http://www.microsoft.com" target="_blank" id="link"
runat="server">Microsoft</a> <br/>
        </div>
        <input type = "submit" id = "button1" runat = "server"
onserverclick="change"/>
      </form></body></html>
```

Fișierul .aspx

```
protected void change(object sender, EventArgs e)
{
    mainDiv.Attributes["align"] = "center";
    mainBody.Attributes["bgcolor"] = "#659EC7";
    link.Attributes["href"] = "http://www.microsoft.com/romania/";
    link.InnerText = "Microsoft Romania";
}
```

Fișierul .aspx.cs

III.2. Limbajul C#

Limbajul C# este un limbaj nou de programare, apărut odată cu platforma .NET. Ideea centrală a limbajului C# este programarea orientată pe obiecte (POO). O aplicație C# este formată din una sau mai multe clase, grupate în spații de nume (*namespaces*).

O clasă reprezintă un șablon care definește forma unui obiect. Este un tip de date abstract, care definește atât datele cât și codul care va prelucra acele date. Obiectele sunt instanțe, particularizări ale claselor. Datele din cadrul clasei se mai numesc proprietăți, iar funcțiile mai poartă numele de metode.

Simplificând mult lucrurile, puteți să vă imaginați o clasă ca o extindere a tipului structură din C sau înregistrare din Pascal. Într-o structură puteți defini date de tipuri diferite. Într-o clasă, pe lângă aceste date există și funcțiile care le prelucrează. La fel cum puteți defini variabile de tip structură, se pot defini variabile de tip clasă (numite obiecte). Accesul la câmpurile unei structuri se realizează prin *nume_variabilă_structură . nume_câmp*.

Analog, accesul la datele și funcțiile membre ale unei clase se realizează prin *obiect . proprietate* sau *obiect . metodă*.

Un spațiu de nume reprezintă o schemă logică pentru denumirea tipurilor de date înrudite. Vă puteți imagina spațiile de nume ca fiind asemănătoare cu structura de directoare și fișiere din Windows. La fel cum într-un folder salvați fișiere înrudite, într-un spațiu de nume se găsesc obiecte de tipuri asemănătoare. De exemplu clasele *HTMLInputTextBox* și *HTMLAnchor* se vor găsi în același spațiu de nume, deoarece ambele reprezintă controale html de interfață.

Rădăcina acestei ierarhii de spații de nume este *System*. Spațiile de nume pot conține la rândul lor alte spații de nume. De exemplu, *System* conține spațiile de nume *System.Data* (unde se găsesc clase pentru conectarea la diverse surse de date) și *System.Web* (unde se găsesc toate clasele pentru dezvoltarea unei aplicații web). Controalele server Html sunt incluse în spațiul de nume *System.Web.UI.HtmlControls*, iar controalele server Web în spațiul *System.Web.UI.WebControls*. Pentru a avea acces la clasele dintr-un anumit spațiu de nume, acesta trebuie inclus în aplicație folosind instrucțiunea *using*:

```
using System
using System.Web
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
```

III.2.1. Vocabularul limbajului

Ca și limbajul C++ cu care se înrudește, limbajul C# are un alfabet format din litere mari și mici ale alfabetului englez, cifre și alte semne. Vocabularul limbajului este format din identificatori, expresii, separatori, delimitatori și comentarii.

Comentariile pot fi scrise pe un singur rând prin folosirea *//*. Tot ce urmează după caracterele *//* sunt considerate, din acel loc, până la sfârșitul rândului drept comentariu. Comentariile pe mai multe rânduri pot fi scrise prin folosirea */** și **/*.

Un identificator reprezintă o succesiune de caractere care îndeplinește următoarele reguli:

- începe cu o literă sau cu unul dintre caracterele *"_"* și *"@"*;
- primul caracter poate fi urmat numai de litere, cifre sau un caracter de subliniere;
- identifiatorii care reprezintă cuvinte cheie nu pot fi folosiți în alt scop decât acela pentru care au fost definiți.

III.2.2. Tipuri de date

C# include două categorii generale de tipuri predefinite: tipuri simple (sau tipuri valoare) și tipuri referință. Tipurile simple cuprind tipurile numerice (întregi, reale și char), tipul boolean, tipul struct și tipul enumerare. În categoria tipurilor referință se încadrează tipurile tablou, șir de caractere și clasă.

Tipurile întregi

Tip	Descriere	Domeniul de valori
byte	Întreg pe 8 biți fără semn	0 .. 255
sbyte	Întreg pe 8 biți cu semn	-128 .. 127
short	Întreg pe 16 biți cu semn	-32 768 .. 32 767
ushort	Întreg pe 16 biți fără semn	0 .. 65 535
int	Întreg pe 32 biți cu semn	-2 147 483 648 .. 2 147 483 647
uint	Întreg pe 32 biți fără semn	0 .. 4 294 967 295
long	Întreg pe 64 biți cu semn	$-2^{63} .. 2^{63} - 1$
ulong	Întreg pe 64 biți fără semn	$0 .. 2^{64} - 1$

Tipuri în virgulă mobilă

Tip	Descriere	Domeniul de valori
float	tip cu virgulă mobilă, simplă precizie, pe 32 biți (8 pentru exponent, 24 pentru mantisă)	1.5 E-45 .. 3.4E+38
double	tip cu virgulă mobilă, dublă precizie, pe 64 biți (11 pentru exponent, 53 pentru mantisă)	5E-324 .. 1.7E+308
decimal	tip zecimal, pe 128 biți (96 pentru mantisă), 28 de cifre semnificative	1E-28 .. 7.9E+28

Valorile de tip decimal trebuie urmate de m sau M.

```
protected void Page_Load(object sender, EventArgs e)
{
    decimal balanta, dobanda;
    //calcul sold nou
    balanta = 2000.201m;
    dobanda = 0.1m;
    balanta = balanta * dobanda + balanta;
    mainDiv.InnerHtml = "<b>Noul sold este " + balanta + "
    RON</b>";}
```

Exemplul 3.3

Observații:

În fișierul .aspx am presupus definit un tag <div> cu atributele id = „mainDiv” și runat = “server”.

Tipul caracter

În C# caracterele nu sunt reprezentate pe 8 biți ca în C++, ci pe 16 biți, datorită folosirii modelului *Unicode*. Modelul Unicode definește un set de caractere care poate reprezenta caractere din mai multe limbi. Setul standard de caractere ASCII este un subset al modelului Unicode. Domeniul de valori al tipului char este [0..65535].

Exemplu:

```
char c = 'ș';
mainDiv.InnerHtml += c;
```

Tipul bool

Acest tip reprezintă valorile de adevăr și fals. În C# acestea sunt definite prin cuvintele rezervate **true** și **false**. Nu este definită nici o regulă de conversie între tipul **bool** și valori întregi (1 nu este convertit în true și nici 0 în false).

În următorul exemplu, în pagina web va fi afișată valoarea true:

```
bool c = true;
mainDiv.InnerText += c;
```

III.2.3. Operatori

Operatorii C# sunt cuprinși în următoarele categorii:

- Operatori aritmetici
- Operatori de incrementare/decrementare
- Operatori relaționali
- Operatori logici
- Operatori de atribuire
- Operatorul condițional

Operatorii aritmetici sunt +, -, *, /, %, și au aceeași semnificație ca și în C++. Toți operatorii aritmetici sunt operatori binari. În cazul în care + este folosit cu operanzi de tip șir de caractere, rezultatul este un șir obținut prin concatenarea celor doi operanzi. Modul de utilizare al operatorilor aritmetici este descris în exemplul următor:

Exemplul 3.4

```
protected void Page_Load(object sender, EventArgs e)
{
    int x = 7, y = 2;
    mainDiv.InnerHtml = "Suma dintre " + x + " si " + y + " este:" + (x+y) +
"<br/>";
    mainDiv.InnerHtml += "Diferenta dintre " + x + " si " + y + " este: " + (x-y) +
"<br/>";
    mainDiv.InnerHtml += "Produsul dintre " + x + " si " + y + " este: " + (x*y) +
"<br/>";
    mainDiv.InnerHtml += x + " / " + y + " = " + (x/y) + " rest " + (x%y); }
```


Rezultatul execuției este următorul:

Suma dintre 7 și 2 este: 9 Diferența dintre 7 și 2 este: 5 Produsul dintre 7 și 2 este: 14 $7 / 2 = 3$ rest 1
--

Operatorii de incrementare și decrementare sunt ++, --, ambii operatori având o formă prefixată respectiv postfixată:

Operator	Semnificație
++x	Se mărește cu o unitate valoarea variabilei x, și apoi x este utilizat în cadrul expresiei (preincrementare)
x++	Se mărește cu o unitate valoarea variabilei x după ce x este utilizat în cadrul expresiei (postincrementare)
--x	Se micșorează cu o unitate valoarea variabilei x, și apoi x este utilizat în cadrul expresiei (predecrementare)
x--	Se micșorează cu o unitate valoarea variabilei x după ce x este utilizat în cadrul expresiei (postdecrementare)

De exemplu, după executarea secvenței

```
int a = 7, b = 2, c;  
c = --a * b++;
```

variabila c va avea valoarea 12, (se micșorează a cu o unitate și se efectuează înmulțirea), a va avea valoarea 6, iar b va fi 3.

Operatorii relaționali sunt:

- == (egal cu)
- != (diferit)
- > (mai mare)
- < (mai mic)
- >= (mai mare egal)
- <= (mai mic egal)

Rezultatul evaluării unei expresii ce conține operatorii relaționali, este de tip bool.

Operatorii logici sunt:

- ! (negația logică)
- && (și logic)
- || (sau logic)

Tabelul operatorilor logici este:

p	q	p && q	p q	!p
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

În cadrul unei expresii logice ce conține operatorul &&, dacă primul operand are valoarea false rezultatul va fi false, indiferent de valoarea celui de al doilea operand. În mod

asemănător, într-o expresie logică cu operatorul |, dacă primul operand are valoarea true rezultatul va fi true, indiferent de valoarea celui de al doilea operand. În ambele cazuri, al doilea operand nu mai este evaluat, codul fiind mai eficient.

Operatorul de atribuire este =, și se folosește într-o construcție de forma *variabilă* = *expresie*. Se evaluează expresia, iar rezultatul evaluării este atribuit variabilei. Dacă este cazul se realizează și o conversie a valorii către tipul variabilei.

Pentru atribuire se pot folosi și operatorii +=, -=, *=, /=, %= . De exemplu atribuirea $x = x + 1;$, poate fi rescrisă $x += 1;$

Operatorul condițional are forma:

expresie1 ? expresie2: expresie3

Dacă expresia1 este adevărată, atunci se returnează valoarea expresiei2, altfel se returnează valoarea expresiei3. Următoarea secvență de instrucțiuni va returna maximumul dintre variabilele a și b.

$x > y ? x : y;$

```
protected void Page_Load(object sender, EventArgs e)
{
    int x = 7, y = 2;
    mainDiv.InnerHtml = "Maximumul este " + (x > y ? x : y);
}
```

În exemplul următor, pe ecran va fi afișată textul „Maximumul este 7”:

Precedența operatorilor este prezentată în tabelul următor:

Precedența operatorilor în C#
Prioritate maximă
() [] new sizeof
! ~ (cast) +(unar) ++ --
* / %
+ -
<< >>
< > <= >=
== !=
&&
?:
= op=
Prioritate minimă

III.2.4. Conversii

În C# există două tipuri de conversii numerice:

- implicite
- explicite.

Conversia implicită se efectuează doar dacă nu este afectată valoarea convertită.

Conversiile implicite sunt prezentate în tabelul următor:

Conversie din	în
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
char	ushort, int, uint, long, ulong, float, double, decimal
float	double
ulong	float, double, decimal

Conversia explicită se realizează prin intermediul operatorului *cast*, atunci când nu există posibilitatea unei conversii implicite. De exemplu, pentru a returna un număr real în urma împărțirii a două numere întregi, se folosește (float):

```
int a = 7, b = 2;
float r;
r = (float) a/b;           //r va avea valoarea 3.5
```

Conversii între numere și șiruri de caractere.

Conversia unui număr la un șir de caractere se realizează cu ajutorul metodei *ToString()*. De exemplu

```
int a = 7;
rezultat.InnerText = "Numărul este: " + a.ToString();
```

Conversia unui șir de caractere la un număr, se poate realiza prin apelarea metodelor din clasa *Convert*. De exemplu, pentru a citi de la tastatură un număr introdus prin intermediul unui control Html `<input type="text" id="nr1" runat="server">` se poate folosi următoarea linie de cod:

```
int a;
a = Convert.ToInt32(nr1.Value);
```

III.2.5. Funcții matematice

Funcțiile matematice puse la dispoziție de C# se găsesc în clasa *Math*. Printre cele mai importante amintim:

Funcția	Descriere
<i>Abs (x)</i>	Returnează modulul unei valori numerice
<i>Ceiling (x)</i>	Returnează cel mai apropiat întreg mai mare sau egal decât valoarea trimisă ca parametru.
<i>Exp (x)</i>	Returnează e^x
<i>Floor (x)</i>	Returnează cel mai apropiat întreg mai mic sau egal decât valoarea trimisă ca parametru.
<i>Log (x)</i>	Returnează logaritmul natural.
<i>Log10(x)</i>	Returnează logaritmul în baza 10.
<i>Max (x, y)</i>	Returnează maximumul dintre 2 numere.
<i>Min (x, y)</i>	Returnează minimumul dintre 2 numere.
<i>Pow (x, y)</i>	Returnează x^y
<i>Round (x, n)</i>	Returnează cel mai apropiat întreg sau un număr rotunjit cu un anumit număr de zecimale.
<i>Sqrt (x)</i>	Returnează radicalul numărului trimis ca parametru.
<i>Truncate(x)</i>	Returnează partea întreagă a numărului trimis ca parametru.

Exemplul 3.5

```
protected void Page_Load(object sender, EventArgs e)
{
    mainDiv.InnerHtml = "Floor(3.75) = " + Math.Floor(3.75) + "<br/>";
    mainDiv.InnerHtml += "Ceiling(3.25) = " + Math.Ceiling(3.25) + "<br/>";
    mainDiv.InnerHtml += "Round(3.25) = " + Math.Round(3.25) + "<br/>";
    mainDiv.InnerHtml += "Round(3.257) = " + Math.Round(3.257, 2) + "<br/>";
    mainDiv.InnerHtml += "Round(3.249) = " + Math.Round(3.249, 2) + "<br/>";
    mainDiv.InnerHtml += "Truncate(3.75) = " + Math.Truncate(3.75) + "<br/>";
    mainDiv.InnerHtml += "Truncate(3.25) = " + Math.Truncate(3.25) + "<br/>";
}
```

La executarea secvenței de instrucțiuni:

se va obține următorul rezultat:

```
Floor(3.75) = 3
Ceiling(3.25) = 4
Round(3.25) = 3
Round(3.257) = 3,26
Round(3.249) = 3,25
Truncate(3.75) = 3
Truncate(3.25) = 3
```

Dacă funcția `Round` este apelată fără cel de al doilea parametru, va returna cel mai apropiat întreg față de valoarea trimisă ca parametru. Dacă se precizează și al doilea parametru, atunci primul parametru va fi rotunjit la o valoare cu un număr de zecimale specificat de al doilea parametru.

III.2.6. Instrucțiuni C#

Instrucțiuni decizionale.

În limbajul C# există două instrucțiuni decizionale: simplă (*if*) și multiplă (*switch*).

Rezultatul execuției este următorul:

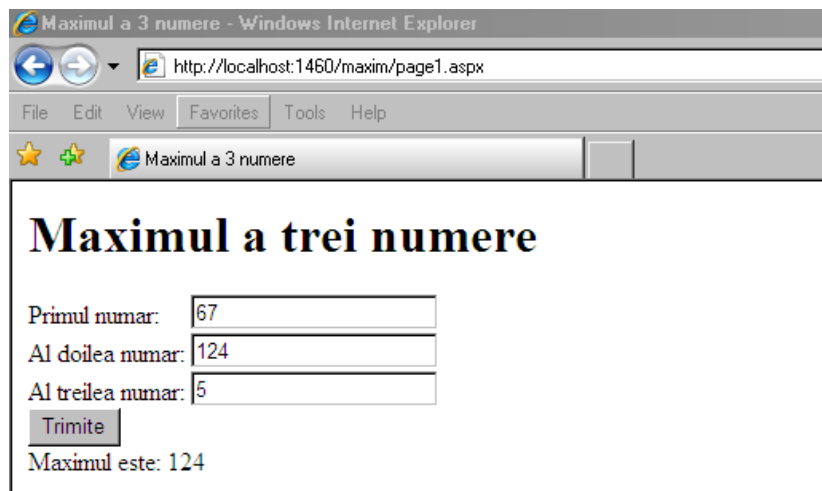


Figura 3.2

Exemplul 3.7 În următorul exemplu sunt calculate rădăcinile ecuației de gradul II:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Ecuatie gradul II</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      a = <input id="nr1" type="text" runat = "server"/><br />
      b = <input id="nr2" type="text" runat = "server"/><br />
      c = <input id="nr3" type="text" runat = "server"/><br />
      <span id = "rezultat" runat = "server" /><br /><br />
      <input id="Button1" type="button" value="Calculeaza" runat =
"server" onclick = "ecuatie"/>
    </div>
  </form></body></html>
```

Fișierul .aspx

```
public partial class _Default : System.Web.UI.Page
{
  protected void ecuatie(object sender, EventArgs e)
  { int a, b, c, d; float x1, x2;
    a = Convert.ToInt32(nr1.Value);
    b = Convert.ToInt32(nr2.Value);
    c = Convert.ToInt32(nr3.Value);
    if (a == 0)
    { rezultat.InnerText = "Ecuatie de gradul I"; }
    else
    { d = b * b - 4 * a * c;
      if (d < 0)
      { rezultat.InnerText = "Numere complexe"; }
      else
      { x1 = (float)(-b + Math.Sqrt(d)) / (2 * a);
        x2 = (float)(-b - Math.Sqrt(d)) / (2 * a);
        rezultat.InnerText = "x1 = " + x1 + " x2 = " + x2;
      }
    }
  }
}
```

Fișierul .aspx.cs

2. Instrucțiunea decizională multiplă *switch* are următoarea sintaxă:

```
switch (expresie)
{
    case val1: instrucțiuni1; break;
    case val2: instrucțiuni2; break;
    case val3: instrucțiuni3; break;
    .....
    case valN: instrucțiuniN; break;

    default: instrucțiuni; break;
}
```

Modul de execuție este următorul: se evaluează expresia, și dacă rezultatul evaluării este egal cu una din valorile de la 1 la N, atunci se execută instrucțiunile de pe ramura respectivă. Dacă rezultatul evaluării expresiei nu este egal cu nici una din valori, atunci se execută instrucțiunile de pe ramura default (daca aceasta există).

Spre deosebire de C++, în C# este obligatorie existența instrucțiunii break pe fiecare ramură, în caz contrar se generează o eroare la compilare.

Exemplul 3.8 Următorul exemplu este o pagină web în care se introduc de la tastatură două numere, și în funcție de alegerea utilizatorului se poate realiza una din operațiile: adunare, scădere, înmulțire, împărțire. Selecția este realizată de utilizator prin intermediul unui tag Html <select>. Rezultatul operației este afișat într-un control html de tip input (<input type="text">) cu atributul *readonly*.

```
public partial class _Default : System.Web.UI.Page
{
    protected void calcul(object sender, EventArgs e)
    {
        int x, y, op; float r;
        x = Convert.ToInt32(nr1.Value);
        y = Convert.ToInt32(nr2.Value);
        op = Convert.ToInt32(operatie.Value);
        switch (op)
        {
            case 1: r = x + y; rezultat.Value = r.ToString(); break;
            case 2: r = x - y; rezultat.Value = r.ToString(); break;
            case 3: r = x * y; rezultat.Value = r.ToString(); break;
            case 4:
                if (y != 0)
                { r = (float) x / y; rezultat.Value = r.ToString(); }
                else
                { rezultat.Value = "Impartire la 0 !"; }
                break;
        }
    }
}
```

Fișierul .aspx

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Switch</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <input id="nr1" type="text" runat="server"/>&nbsp;
      <select id="operatie" name="operator" runat="server">
        <option value="1">+</option>
        <option value="2">-</option>
        <option value="3">*</option>
        <option value="4">/</option>
      </select>&nbsp;
      <input id="nr2" type="text" runat="server"/> =
      <input id="rezultat" readonly="readonly" type="text" runat="server" /> <br />
      <input id="Button1" type="button" value="Calculeaza" runat="server" onserverclick="calcul"/><br />
      <div id="tst" runat="server" />
    </div>
  </form></body></html>

```

Fișierul .aspx.cs

Rezultatul execuției este următorul:

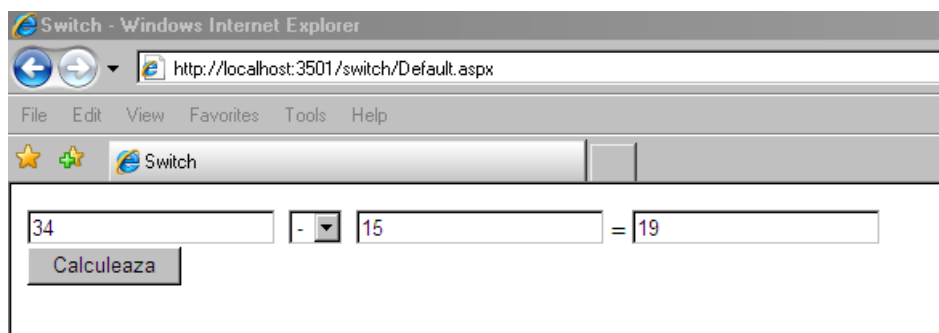


Figura 3.3

Instrucțiuni repetitive

Instrucțiunile repetitive în C# sunt:

- Cu test inițial (while)
- Cu test final (do while)
- Cu număr cunoscut de pași (for)

1. Instrucțiunea repetitivă *while* are sintaxa:

```

while (condiție)
    instrucțiuni

```

Modul de execuție este următorul: cât timp este adevărată condiția, se execută instrucțiunile.

Exemplul 3.8 Aplicația următoare calculează cmmdc a două numere introduse de utilizator.


```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Expresie</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <input id="nr1" type="text" runat = "server"/>
      <input id="Button1" type="button" value="Calculeaza" runat = "server"
        onclick = "calcul"/><br />
      <p id = "rezultat" runat = "server" />
    </div>
  </form>
</body>
</html>
```

Fișierul .aspx

```
public partial class _Default : System.Web.UI.Page
{
    protected void calcul(object sender, EventArgs e)
    {
        int n = Convert.ToInt32(nr1.Value), s = 0, p = 1;
        for (int i = 1; i <= n; i++)
        {
            p = p * i;
            s = s + p;
        }
        rezultat.InnerText = s.ToString();
    }
}
```

Fișierul .aspx.cs

III.2.7. Tablouri în C#

Pentru a declara un tablou unidimensional în C#, se folosește sintaxa:

tip_elemente [] variabilă;

Alocarea de spațiu în memorie pentru noua variabilă se realizează printr-o instanțiere de forma:

variabilă = **new** tip_elemente[număr_elemente];

De exemplu:

```
int[] v;
v = new int[20];
```

Aceași declarație poate fi scrisă printr-o singură instrucțiune: `int[] v = new int[10];`

La declararea tabloului se poate realiza și inițializarea elementelor sale: `int[] v = {1,2,3,4 };`

Pentru a afișa elementele tabloului, se pot folosi atât instrucțiunile repetitive clasice, cât și o instrucțiune pentru parcurgerea colecțiilor de date, numită *foreach*.

Exemplul 3.10. Afișarea unui tablou folosind instrucțiunea *for*.

```
int[] v = { 1, 2, 3, 4 };
for (int i = 0; i < v.Length; i++)
    mainDiv.InnerHtml += v[i] + " ";
```

Observații:

Proprietatea *Length* returnează numărul de elemente din tablou.

Exemplul 3.11. Afișarea unui tablou folosind instrucțiunea *foreach*.

```
int[] v = { 1, 2, 3, 4 };
foreach (int i in v)
    mainDiv.InnerHtml += i + " ";
```

În cazul instrucțiunii *foreach*, variabila contor nu are ca semnificație poziția elementului în vector (ca la *for*), ci reprezintă chiar elementul parcurs. De aceea, această variabilă trebuie să aibă același tip cu cel al elementelor din tablou.

Pentru orice variabilă de tip tablou se pot apela următoarele metode:

- **Sum()** – returnează suma elementelor din tablou.

Exemplu:

```
mainDiv.InnerHtml += "Suma elementelor din vector este: " +
    v.Sum();
```

- **Average()** – returnează media aritmetică din tablou

Exemplu:

```
mainDiv.InnerHtml += "Suma elementelor din vector este: " + v.Average();
```

- **Contains(valoare)** – returnează *true* dacă valoarea trimisă ca parametru se găsește în tablou, și *false* în caz contrar

Exemplul 3.12

```
if (v.Contains(6) == true)
    mainDiv.InnerHtml += "Elementul se gaseste in vector";
else
    mainDiv.InnerHtml += "Elementul nu se gaseste in
vector";
```

- **CopyTo(tablou_destinație, indice)** – copiază toate elementele din tabloul inițial în tabloul destinație începând cu poziția dată de cel de al doilea parametru

Exemplul 3.13

```
int[] v = { 1, 2, 3, 4 };
int[] w = new int[10];
v.CopyTo(w, 0); // w = 1 2 3 4 0 0 0 0 0 0
v.CopyTo(w, 3); // w = 0 0 1 2 3 4 0 0 0 0
```

- **Equals(variabilă_tablou)** – returnează true dacă cele două tablouri conțin aceleași elemente și false în caz contrar.

Exemplul 3.14

```
if (v.Equals(w) == true)
    mainDiv.InnerHtml += "Cei doi vectori au aceleasi elemente";
else
    mainDiv.InnerHtml += "Cei doi vectori nu au aceleasi elemente";
```

Clasa **Array** pune la dispoziția programatorului alte câteva metode pentru prelucrarea vectorilor:

- **Sort(tablou)** – sortează crescător elementele din vectorul trimis ca parametru
Exemplu: `Array.Sort(w);`
- **Reverse(tablou)** – inversează elementele vectorului trimis ca parametru
Exemplu: `Array.Reverse(w);`
- **IndexOf(tablou, valoare)** – caută o valoare în vector, și returnează -1 dacă nu se găsește, respectiv poziția pe care se găsește valoarea în vector.

Exemplul 3.15

```
if ((poz = Array.IndexOf(v, 3)) != -1)
    mainDiv.InnerHtml += "Elementul cautat se gaseste in vector pe pozitia "+poz;
else
    mainDiv.InnerHtml += "Elementul cautat nu se gaseste in vector";
```

Exemplul 3.16 Exemplul următor este o pagină web, care prezintă modul de utilizare a vectorilor.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>Tablouri</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div id = "mainDiv" runat = "server">
      </div>
    </form>
  </body>
</html>
```

Fișierul .aspx

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        int[] v = { 1, 2, 3, 4 }; //initializarea elementelor din vector
        int[] w = new int[10];
        int poz;
        //Afisarea elementelor
        mainDiv.InnerHtml += "Elementelor vectorului sunt: ";
        for (int i = 0; i < v.Length; i++) mainDiv.InnerHtml += v[i] + " ";
    }
}
```

Fișierul .aspx.cs

```
mainDiv.InnerHtml += "<br/>";
    //Suma si media aritmetica a elementelor
mainDiv.InnerHtml += "Suma elementelor din vector este: " + v.Sum() + "<br/>";
mainDiv.InnerHtml += "Media elementelor din vector este: " + v.Average() + "<br/>";
    //Copierea elementelor unui vector intr-un alt vector
v.CopyTo(w, 0);
mainDiv.InnerHtml += "Elementele vectorului sunt: ";
for (int i = 0; i < w.Length; i++)mainDiv.InnerHtml += w[i] + " ";
mainDiv.InnerHtml += "<br/>";
    //Compararea a doi vectori
if (v.Equals(w) == true)
{
    mainDiv.InnerHtml += "Cei doi vectori au aceleasi elemente <br/>";
}
else
{
    mainDiv.InnerHtml += "Cei doi vectori nu au aceleasi elemente <br/>";
}
    //Sortarea elementelor vectorului
w[3] = 5;
Array.Sort(w);
mainDiv.InnerHtml += "Vectorul sortat: ";
foreach (int i in w) mainDiv.InnerHtml += i + " ";
mainDiv.InnerHtml += "<br/>";

    //Inversarea elementelor
Array.Reverse(w);
mainDiv.InnerHtml += "Vectorul inversat: ";
for (int i = 0; i < w.Length; i++) mainDiv.InnerHtml += w[i] + " ";
mainDiv.InnerHtml += "<br/>";

    //Cautarea unui element in vector
if (v.Contains(6) == true)
{
    mainDiv.InnerHtml += "Elementul se gaseste in vector<br/>";
}
else
{
    mainDiv.InnerHtml += "Elementul nu se gaseste in vector<br/>";
}

if ((poz = Array.IndexOf(v, 3)) != -1)
{
    mainDiv.InnerHtml += "Elementul cautat se gaseste in vector pe pozitia " + poz;
}
else
{
    mainDiv.InnerHtml += "Elementul cautat nu se gaseste in vector";
}
}
```

Rezultatul execuției este următorul:

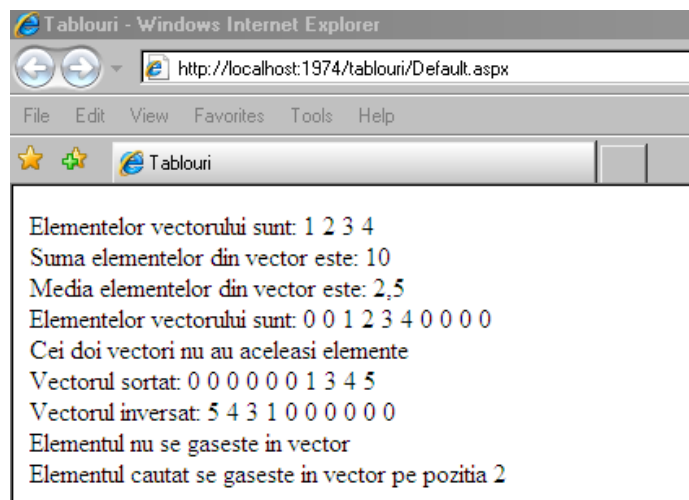


Figura 3.4

III.2.8. Șiruri de caractere

O variabilă de tip șir de caractere poate fi declarată folosind sintaxa:

```
string nume_variabilă
```

De exemplu: `string s;`

La declarare, o variabilă poate fi inițializată:

```
string s = "Hello World";
```

Operații cu șiruri de caractere

1. Determinarea lungimii unui șir de caractere se realizează cu ajutorul proprietății

Length:

Exemplul 3.17

```
mainDiv.InnerHtml = "Sirul " + s + " are " + s.Length + " caractere" + "<br/>";
```

2. Conversia șirurilor la majuscule sau minuscule

Conversiile sunt realizate de metodele *ToUpper()*, respectiv *ToLower()*.

Exemplul 3.18

```
mainDiv.InnerHtml += s.ToUpper() + "<br/>";
mainDiv.InnerHtml += s.ToLower() + "<br/>";
```

3. Eliminarea spațiilor dintr-un șir poate fi realizată cu metodele:

- *Trim()* – Elimină spațiile de la ambele extremități ale șirului
- *TrimEnd()* – Elimină spațiile de la sfârșitul șirului
- *TrimStart()* – Elimină spațiile de la începutul șirului

Exemplul 3.19

```
mainDiv.InnerHtml += s.Trim() + "<br/>";
mainDiv.InnerHtml += s.TrimEnd() + "<br/>";
mainDiv.InnerHtml += s.TrimStart() + "<br/>";
```

4. Compararea șirurilor de caractere

Implicit, compararea a două șiruri de caractere este *case sensitive*. Metodele pentru compararea șirurilor acceptă un parametru pentru a seta modul de comparare. De exemplu, a realiza o comparare în care nu contează dacă literele sunt majuscule sau minuscule, se folosește obiectul *StringComparison* cu proprietatea *OrdinalIgnoreCase*.

Observații:

Pentru a folosi obiectul *StringComparison*, în pagina web trebuie inclus namespace-ul *System.Globalization*:

```
using System.Globalization;
```

Metodele pentru compararea a două șiruri de caractere sunt:

- *Equals()* - returnează valoarea *true* dacă două șiruri sunt egale, și *false* în caz contrar.

Exemplul 3.20

```
if (s.Equals(s1) == true)
    {mainDiv.InnerHtml += "Sirurile sunt egale <br/>";}
else
    {mainDiv.InnerHtml += "Sirurile nu sunt egale <br/>";};
```

Pentru a nu face distincție între minuscule și majuscule, se apelează metoda *Equals* cu al doilea parametru :

Exemplul 3.21

```
if (s1.Equals(s2, StringComparison.OrdinalIgnoreCase) == true)
    {mainDiv.InnerHtml += "Sirurile sunt egale <br/>";}
else
    {mainDiv.InnerHtml += "Sirurile nu sunt egale <br/>";};
```

- *CompareTo()* - returnează valoarea 0 dacă șirurile sunt egale, o valoare mai mică decât 0 dacă șirul 1 mai mic decât șirul2, și o valoare mai mare ca 0 dacă șirul 1 este mai mare decât șirul 2.

Exemplul 3.22

```
int value = s1.CompareTo(s2);
if ( value == 0)
    {mainDiv.InnerHtml += "Sirurile sunt egale <br/>";}
else
    if (value < 0)
        {mainDiv.InnerHtml += s1 + " mai mic decat " + s2 + "<br/>";}
    else
        {mainDiv.InnerHtml += s1 + " mai mare decat " + s2 + "<br/>";};
```

5. Operații cu subșiruri.

- Extragerea unui subșir dintr-un șir este realizată de metoda *substring(poziție, lungime)*

```
string s = "Hello World";
mainDiv.InnerHtml += s.Substring(6) + "<br/>";
```

Va afișa pe ecran șirul „World”.

```
mainDiv.InnerHtml += s.Substring(6,3) + "<br/>";
```

Va afișa pe ecran șirul „Wor”.

- Căutarea unui subșir într-un șir este realizată de metoda *Contains(subșir)*. Returnează valoarea *true* dacă subșirul este găsit, și *false* în caz contrar.

Exemplul 3.23

```
string s = "Hello World", s1 = "He";
if (s.Contains(s1) == true)
    {mainDiv.InnerHtml += "Subsirul " + s1 + " se gaseste in sirul " + s + "<br/>";}
else
    {mainDiv.InnerHtml += "Subsirul " + s1 + " nu se gaseste in sirul " + s + "<br/>";}
```

- Căutarea primei apariții a unui subșir într-un șir se realizează cu ajutorul metodei *IndexOf(subșir, poziție, lungime, mod_comparare)*. Dacă subșirul nu se găsește în șir, se returnează valoarea *-1*, altfel se returnează poziția pe care se găsește subșirul în șir.

```
string s = "Hello World", s1 = "ello";
int poz = s.IndexOf(s1, 0, 10, StringComparison.OrdinalIgnoreCase);
if (poz == -1)
    {mainDiv.InnerHtml += "Subsirul " + s1 + " nu se gaseste in sirul " + s + "<br/>";}
else
    {
    mainDiv.InnerHtml += "Subsirul " + s1 + " se gaseste pe pozitia " + poz + " in sirul " + s + "<br/>";
    }
```

Exemplul 3.24

Instrucțiunea *s.IndexOf(s1, 0, 10, StringComparison.OrdinalIgnoreCase)* caută prima apariție a subșirului *s1* în primele 10 caractere din șirul *s*, începând cu poziția 10. Căutarea nu este *case sensitive*.

III.2.9. Date calendaristice

O variabilă de tip dată calendaristică este un obiect care poate reține valori pentru data și timp. Declararea unei variabile de acest tip este:

```
DateTime d1;
```

Exemplul 3.24 Variabilele de tip dată calendaristică pot fi inițializate la declarare:

```
DateTime d1 = DateTime.Now; // returneaza data si ora curenta
DateTime d2 = new DateTime(2008, 10, 14); // 14-10-2008
DateTime d3 = new DateTime(2008, 10, 14, 10, 50, 23); // 14-10-2008 10:50:23
mainDiv.InnerHtml = d1.ToString() + "</br>";
mainDiv.InnerHtml += d2.ToString() + "</br>";
mainDiv.InnerHtml += d3.ToString() + "</br>";
```

Rezultatul va fi următorul:

```
13.10.2008 22:30:00
14.10.2008 00:00:00
14.10.2008 10:50:23
```

Metodele și proprietățile care se aplică unei variabile de tip dată calendaristică sunt:

- *ToShortDateString()* – returnează o dată calendaristică în forma scurtă
mainDiv.InnerHtml += d1.ToShortDateString() + "</br>"; Afășează: 14.10.2008
- *ToLongDateString()* – returnează o dată calendaristică în forma desfășurată
mainDiv.InnerHtml += d1.ToLongDateString() + "</br>"; Afășează: 14 octombrie 2008
- *ToShortTimeString()* – returnează ora, minutele dintr-o variabilă de tip dată calendaristică
mainDiv.InnerHtml += d1.ToShortTimeString() + "</br>"; Afășează: 22:30
- *ToLongTimeString()* – returnează ora, minutele și secunde dintr-o variabilă de tip dată calendaristică
mainDiv.InnerHtml += d1.ToLongTimeString() + "</br>"; Afășează: 22:30:00
- *Day* – proprietate care returnează ziua dintr-o dată calendaristică
mainDiv.InnerHtml += d1.Day + "</br>"; Afășează: 14
- *Month* – proprietate care returnează luna dintr-o dată calendaristică
mainDiv.InnerHtml += d1.Month + "</br>"; Afășează: 10
- *Year* – proprietate care returnează ziua dintr-o dată calendaristică
mainDiv.InnerHtml += d1.Year + "</br>"; Afășează: 2008
- *DayOfYear* – proprietate care returnează numărul zilei din cadrul anului
mainDiv.InnerHtml += d1.DayOfYear + "</br>"; Afășează: 288
- *DayOfMonth* – proprietate care returnează ziua din cadrul săptămânii
mainDiv.InnerHtml += d1.DayOfWeek + "</br>"; Afășează: Tuesday

Operații cu date calendaristice

1. Prin adăugarea unui număr la o dată calendaristică, rezultă o altă dată calendaristică. Astfel, se pot folosi metodele *AddSeconds(valoare)*, *AddMinutes(valoare)*, *AddHours(valoare)*, *AddDays(valoare)*, *AddMonth(valoare)*, *AddYears(valoare)*.

Exemplul 3.25

```
mainDiv.InnerHtml += d2.AddDays(-50).ToShortDateString() + "</br>";
mainDiv.InnerHtml += d2.AddMonths(10).ToShortDateString() + "</br>";
mainDiv.InnerHtml += "Ziua de " + d2.ToLongDateString() + " va fi peste 1
an in ziua de " + d2.AddYears(1).DayOfWeek + "</br>";
```

Rezultatul va fi următorul:

```
04.09.2008
24.08.2009
Ziua de 24 octombrie 2008 va fi peste 1 an in ziua de Saturday
```

2. Timpul scurs între două date calendaristice poate fi aflat prin aplicarea următoarelor proprietăți diferenței dintre două date calendaristice: *TotalSeconds*, *TotalMinutes*, *TotalHours*, *TotalDays*:

```
d3 = new DateTime(2008, 10, 28);
d2 = new DateTime(2008, 10, 24);
mainDiv.InnerHtml += Math.Floor((d3 - d2).TotalDays) + "</br>";
```

ezultatul va fi: 4

IV. Modelul client-server

IV.1. Controale server web

Controalele server web oferă facilități superioare controalelor server Html. Astfel, controalele web sunt puternic orientate obiect, permit legarea la diferite surse de date, detectează automat tipul browserului și afișarea pe client va fi optimizată în funcție de capacitățile acestuia.

Un control web poate fi creat fie în modul design (din fereastra *Toolbox*), fie în pagina .aspx folosind tag-urile `<asp>` `</asp>`.

```
<asp:Label ID="Label1" runat="server" Text="Hello world!"></asp:Label>
```

Controalele web pot răspunde la diferite evenimente. De exemplu, un obiect de tip *button* generează evenimentul *Click* atunci când utilizatorul apasă butonul. În momentul apariției unui eveniment, se realizează o cerere către server, iar datele din pagina web sunt trimise serverului. Aici se execută codul C# asociat evenimentului, iar rezultatul este trimis către client.

Valorile proprietăților controalelor web pot fi setate prin trei moduri:

- în modul design din fereastra *Properties*
- în sursă paginii (fișierul .aspx)
- prin intermediul codului C# executat pe server

Toate controalele server web au o serie de proprietăți comune:

Proprietate	Descriere
BackColor	Culoarea de fundal a controlului
BorderColor	Culoarea chenarului controlului
BorderWidth	Grosimea chenarului controlului, exprimată în pixeli
BorderStyle	Tipul de chenar. Valorile posibile sunt: NotSet, None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, Outset
CssClass	Clasa CSS asociată controlului
Enabled	Dacă are valoarea False, controlul este inactiv
Font	Acest atribut conține subproprietăți pentru stabilirea caracteristicilor fontului
ForeColor	Culoarea textului
Height	Lungimea controlului
TabIndex	Ordinea în care controlul este accesat la apăsarea tastei Tab de către utilizator în interfață
ToolTip	Textul care apare atunci când utilizatorul poziționează mouse-ul peste control
Width	Lățimea controlului. Unitatea de măsură implicită este pixel-ul. Alte unități posibile sunt: Point, Pica, Inch, Mm, Cm, Percentage, Em.

În continuare sunt prezentate câteva dintre cele mai folosite controale web pentru realizarea interfeței.

IV.1.1. Label

Permite afișarea unui șir de caractere în pagina web. Textul este reținut în proprietatea *Text*:

```
<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="Large"
  ForeColor="#0066FF" Text="Evenimente butoane">
</asp:Label>
```

IV.1.2. Button, LinkButton, ImageButton

Aceste controale creează butoane de *submit* pentru pagina web. Proprietatea *Text* reține șirul de caractere afișat pe buton. În exemplul următor, la apăsarea butoanelor se execută codul C# asociat evenimentului *click*, și este modificată proprietatea text a unui obiect **Label**.

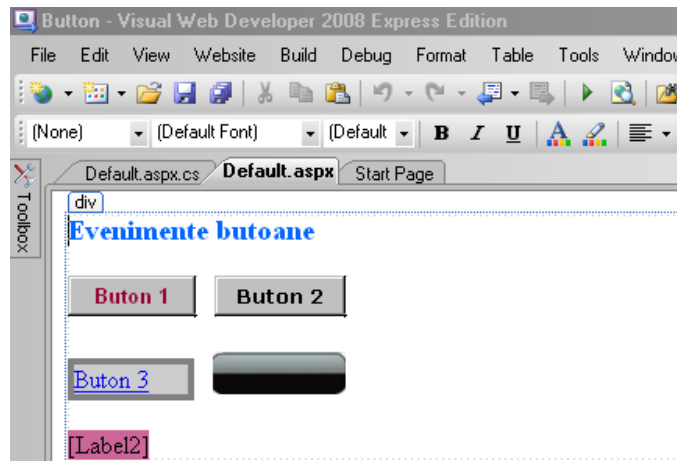


Figura 4.1 Adăugarea de controale button, în modul design

Controlul **LinkButton** afișează un hyperlink în locul unui text, iar controlul **ImageButton** este reprezentat sub forma unei imagini. Proprietatea *ImageUrl* definește calea către fișierul care conține imaginea. În acest exemplu, controlul de tip **ImageButton** are asociat fișierul *button.jpg*.

Observație. Pentru a adăuga o imagine în proiect, puteți crea un director **images** din fereastra Solution Explorer (click dreapta, *new folder*). Aici salvați fișierul care conține imaginea. (dacă fișierul imagine nu apare în folderul *images*, alegeți opțiunea *refresh folder*).

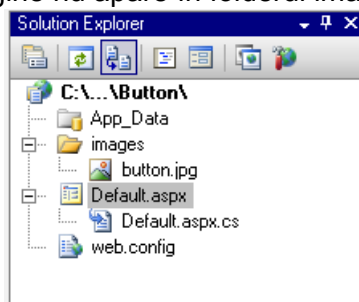


Figura 4.2 Adăugarea unui director Images în cadrul proiectului

Modificarea valorii proprietății ImageUrl se poate realiza fie în modul design, fie direct în codul aspx:

```
<asp:ImageButton ID="ImageButton1" runat="server" Height="27px"
ImageUrl="~/images/button.jpg" onclick="ImageButton1_Click" Width="85px" />
```

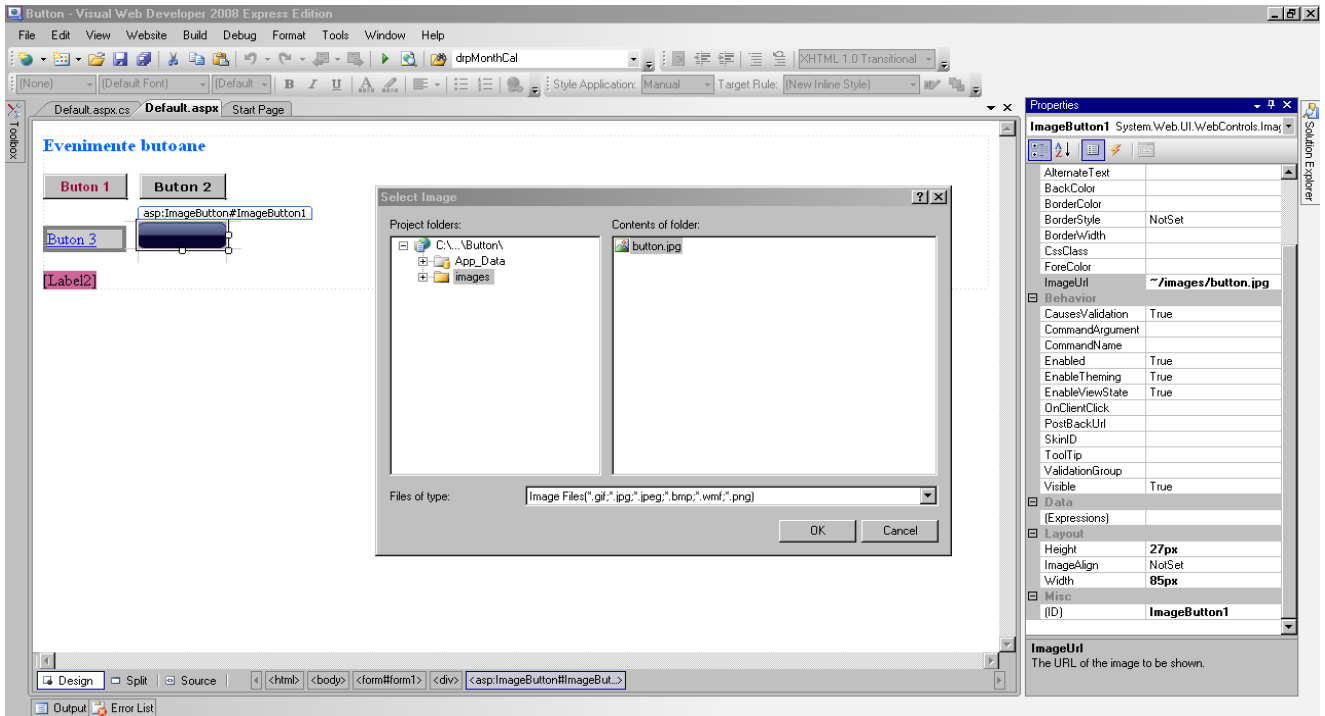


Figura 4.3 Modificarea proprietății ImageUrl în modul design

Exemplul 4.1 La apăsarea unui dublu click pe controlul de tip button se generează antetul metodei care va conține codul C# care va fi executat la apariția evenimentului *Click*:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label2.Text = "Ati apasat butonul 1 !";
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        Label2.Text = "Ati apasat butonul 2! ";
    }
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        Label2.Text = "Ati apasat butonul 3! ";
    }
    protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        Label2.Text = "Ati apasat butonul 4! ";
    }
}
```

IV.1.3. TextBox

Afișează o casetă de dialog în care utilizatorul poate introduce date. Șirul de caractere introdus, este reținut de proprietatea *Text*. Proprietatea *TextMode* definește modul în care utilizatorul poate introduce datele: *SingleLine*, *MultiLine*, *Password*.

Exemplul 4.2 Aplicația următoare permite introducerea unui text, care la apăsarea butonului va fi afișat pe ecran de către controlul Label2.

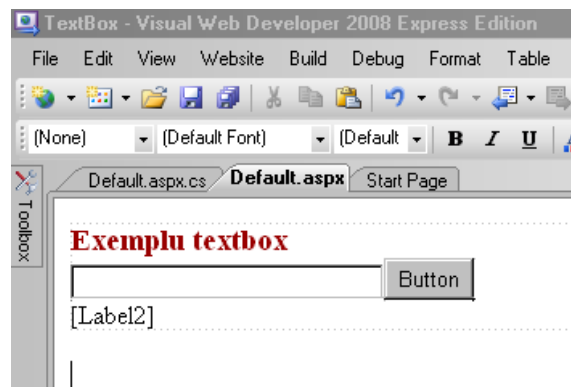


Figura 4.4 Adăugarea unui control TextBox, în modul design

Fișierul .aspx este:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>TextBox</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
        Size="14pt"
          ForeColor="#990000" Text="Exemplu textbox" TabIndex="1">
      </asp:Label><br />
      <asp:TextBox ID="TextBox1" runat="server" ForeColor="#009900"
        ToolTip="Introduceti un text si apasati apoi butonul Submit" Width="195px">
      </asp:TextBox>
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
        Text="Button" TabIndex="2" /><br />
      <asp:Label ID="Label2" runat="server" TabIndex="3"></asp:Label>
    </div>
  </form>
</body>
</html>
```

Codul asociat evenimentului Click pentru butonul din pagină, este:

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label2.Text = TextBox1.Text;
}
```

IV.1.4. CheckBox, CheckBoxList

Controlul CheckBox permite crearea unei casete de marcare. Pentru a lucra cu mai multe casete la nivel unitar, se folosește controlul CheckBoxList.

Controlul CheckBox are proprietatea *Checked*, care are valoarea *True* dacă s-a bifat caseta sau *False* în caz contrar. Șirul de caractere afișat în dreptul casetei, este reținut de proprietatea *Text*.

Exemplul 4.3 În exemplul următor, dacă este bifată prima casetă și apoi se apasă butonul Ok, se modifică proprietatea *Text* a controlului Label1. De asemenea, la apăsarea butonului Submit se modifică proprietatea *Text* a obiectului Label2, fiind afișate orașele selectate prin intermediul casetelor de bifare.

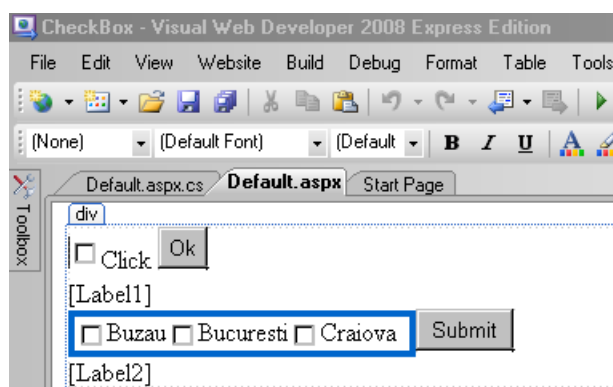


Figura 4.5 Adăugarea controalelor CheckBox, CheckBoxList, în modul design

Verificarea dacă s-a bifat sau nu caseta, se realizează prin intermediul proprietății *Checked*:

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (CheckBox1.Checked == true)
    {
        Label1.Text = "Ati selectat checkbox-ul";
    }
    else
    {
        Label1.Text = "Nu ati selectat checkbox-ul";
    }
}
```

Pentru un control de tip CheckBoxList, există proprietățile *RepeatDirection* și *RepeatLayout*, care permit formatarea controlului. *RepeatDirection* poate lua valorile *Vertical*, respectiv *Horizontal*. Pentru *RepeatLayout* puteți alege între valorile *Flow*, respectiv *Table*. Prin intermediul valorii *Flow*, pot fi poziționate și alte controale pe același rând cu lista de casete.

Exemplul 4.4 Adăugarea de valori pentru un control de tip CheckBoxList se poate realiza atât în modul design, cât și prin intermediul codului din pagina .aspx:

1. Prin cod:

```

<asp:CheckBoxList ID="CheckBoxList1" runat="server"
  BorderColor="#0066CC"
  BorderStyle="Solid" RepeatDirection="Horizontal"
  RepeatLayout="Flow">
  <asp:ListItem>Buzau</asp:ListItem>
  <asp:ListItem>Bucuresti</asp:ListItem>
  <asp:ListItem>Craiova</asp:ListItem>
</asp:CheckBoxList>

```

2. În modul design:

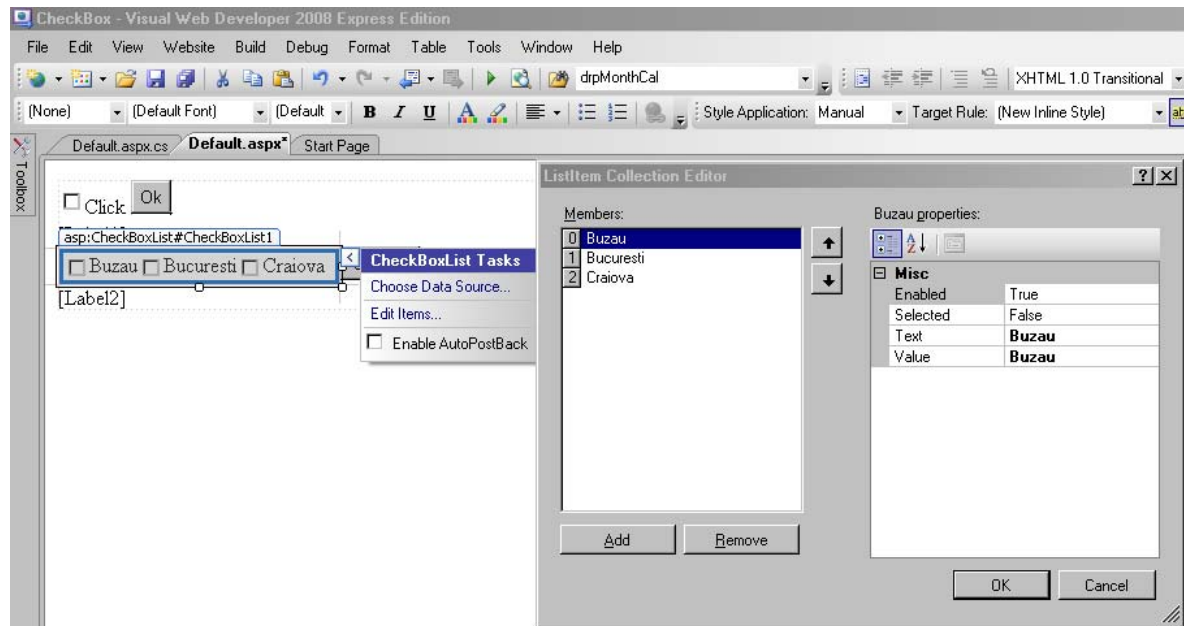


Figura 4.6 Adăugarea de valori pentru controlul CheckBoxList, în modul design

Controlul CheckBoxList are asociat o proprietate numită *Items*. *Items* este un vector care conține valorile din controlul CheckBoxList. La rândul său are proprietatea *Count* ce returnează numărul de valori din listă. Fiecare element poate fi accesat prin intermediul unui indice. În exemplul de mai sus, `CheckBoxList.Items[1].Text` este șirul de caractere „București”. Pentru a verifica dacă a fost bifată o anumită casetă, se folosește proprietatea *Selected*, care poate avea valorile *True* sau *False*.

Efectul execuției este următorul:

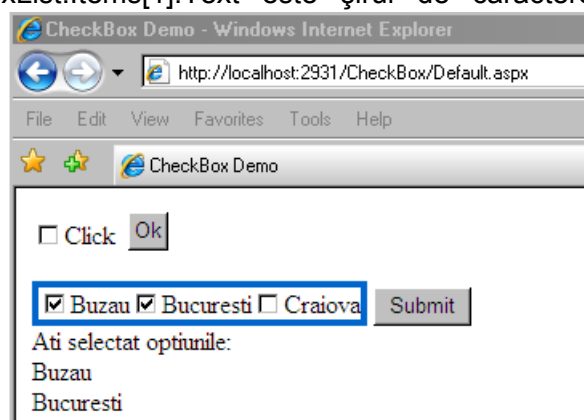


Figura 4.7 Rezultatul prelucrării paginii în urma apăsării butonului Submit

Codul C# asociat evenimentului *Click* pentru butonul Submit este:

```
protected void Button2_Click(object sender, EventArgs e)
{
    String s = "";
    for (int i = 0; i < CheckBoxList1.Items.Count; i++)
    {
        if (CheckBoxList1.Items[i].Selected)
        {
            s = s + CheckBoxList1.Items[i].Text;
            s = s + "<br>";
        }
    }
    if (s.Length > 0)
    {
        Label2.Text = "Ati selectat optiunile:<br>" + s;
    }
    else
    {
        Label2.Text = "Nu ati selectat nici o optiune" ;
    }
}
```

IV.1.5. RadioButton

Afișează un buton radio. Dintr-un grup de butoane radio, doar unul poate fi selectat la un moment dat.

Proprietatea *Text* a acestui control, definește șirul de caractere ce apare în dreptul controlului. Pentru a grupa mai multe butoane radio, se folosește proprietatea *GroupName*.

Exemplu:

```
<asp:RadioButton ID="RadioButton1" runat="server" GroupName="stareCivila"
    Text="Casatorit" /><br />
<asp:RadioButton ID="RadioButton2" runat="server" GroupName="stareCivila"
    Text="Necasatorit" />
```

Exemplul 4.5 Ca și în cazul controlului *CheckBox*, verificarea dacă s-a bifat sau nu butonul radio se realizează prin intermediul proprietății *Checked*.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (RadioButton1.Checked == true)
    {
        Label1.Text = "Sunteti casatorit";
    }
    else if (RadioButton2.Checked == true)
    {
        Label1.Text = "Nu sunteti casatorit ";
    }
}
```

IV.1.6. RadioButtonList

Afișează un grup de butoane radio. Ca și CheckBoxList, acest control are proprietățile *RepeatDirection*, *RepeatLayout* și *Items*. Pentru fiecare opțiune din listă se poate asocia o valoare de tip șir de caractere, prin intermediul atributului *value*.

Exemplul 4.6 Adăugarea de valori în listă poate fi realizată în modul design sau prin intermediul codului din fișierul .aspx.

1. Prin cod:

```
<asp:RadioButtonList ID="RadioButtonList1"
runat="server"
RepeatDirection="Horizontal" RepeatLayout="Flow">
  <asp:ListItem Value="1">Bucuresti</asp:ListItem>
  <asp:ListItem Value="2">Buzau</asp:ListItem>
  <asp:ListItem Value="3">Craiova</asp:ListItem>
</asp:RadioButtonList>
```

2. În design mode:

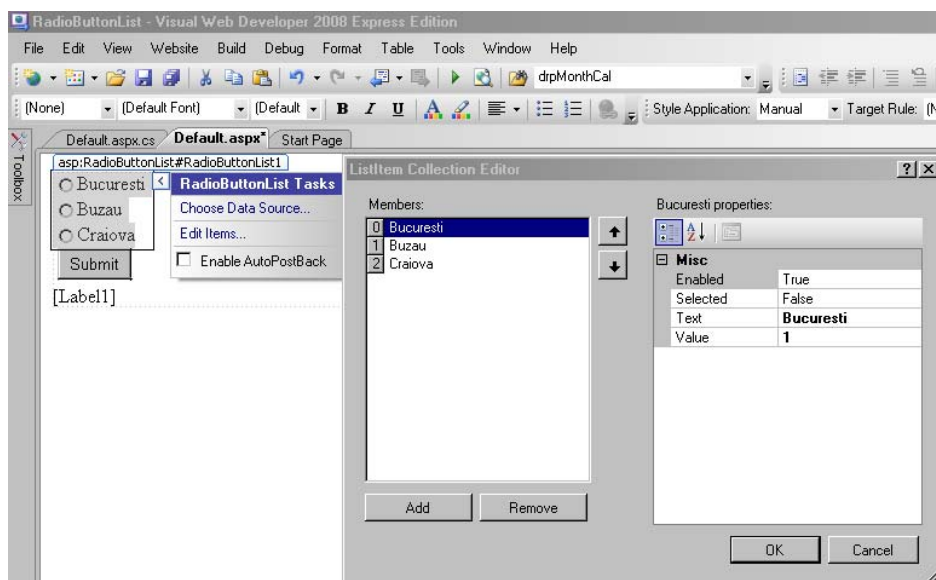


Figura 4.8 Adăugarea de valori pentru controlul RadioButtonList, în modul design

Proprietatea *SelectedValue* returnează valoarea butonului radio selectat. Astfel, pentru a verifica dacă un buton radio a fost selectat sau nu, se poate folosi următorul cod C#:

```
protected void Button1_Click(object sender, EventArgs e)
{
    switch (RadioButtonList1.SelectedValue)
    {
        case "1": Label1.Text = "Ati selectat orasul Bucuresti "; break;
        case "2": Label1.Text = "Ati selectat orasul Buzau "; break;
        case "3": Label1.Text = "Ati selectat orasul Craiova "; break;
    }
}
```

IV.1.7. BulletList

Afișează o listă numerotată. Proprietatea *BulletStyle* definește tipul de marcator, și poate avea valorile: *NotSet*, *Numbered*, *LowerAlpha*, *UpperAlpha*, *LowerRoman*, *UpperRoman*, *Disc*, *Circle*, *Square*.

```
<asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="Disc">
  <asp:ListItem>while</asp:ListItem>
  <asp:ListItem>for</asp:ListItem>
  <asp:ListItem>do while</asp:ListItem>
</asp:BulletedList>
```

Proprietatea *DisplayMode* definește modul în care sunt afișate valorile din listă: *Text*, *HyperLink*, *LinkButton*. În cazul selectării tipului de afișare ca *HyperLink*, atributul *value* reține adresa http:

```
<asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="Square"
  DisplayMode="HyperLink" Target="_blank">
  <asp:ListItem
  Value="http://msdn.microsoft.com/">MSDN</asp:ListItem>
  <asp:ListItem
  Value="http://www.microsoft.com">Microsoft</asp:ListItem>
  <asp:ListItem Value="http://www.microsoft.com/romania/">Microsoft
  Romania</asp:ListItem>
</asp:BulletedList>
```

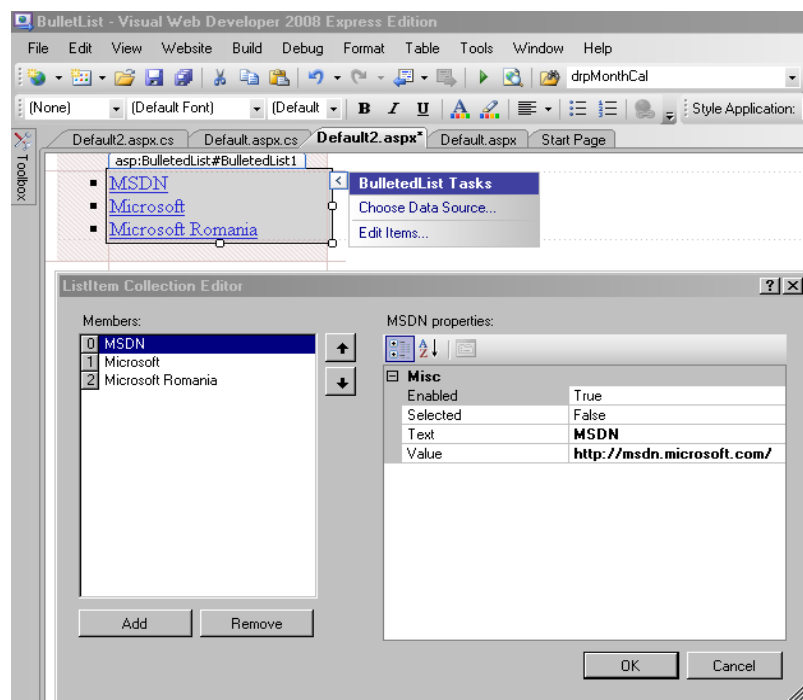


Figura 4.9 Adăugarea de valori pentru controlul BulletList, în modul design

IV.1.8. Image

Afișează o imagine în pagina web. Proprietatea *ImageUrl* definește calea către fișierul care conține imaginea. Textul afișat de browser atunci când user-ul poziționează mouse-ul peste imagine, este reținut de proprietatea *AlternateText*.

```
<asp:Image ID="Image1" runat="server" ImageUrl="~/images/movie1.jpeg"
    Height="406px" Width="265px" AlternateText="Jaws" />
```

IV.1.9. DropDownList

Permite selectarea unei opțiuni dintr-o listă derulantă. Ca și celelalte controale de tip list, DropDownList are proprietățile *Item* și *SelectedValue*. De asemenea, proprietatea *SelectedItem* returnează obiectul selectat din lista derulantă.

Exemplul 4.7 Pagina web următoare permite selectarea unui film dintr-o listă derulantă, iar la apăsarea butonului Submit, în controlul de tip Image este afișat posterul filmului selectat. Posterele sunt fișiere cu extensia .jpg, care au fost salvate în directorul images din cadrul proiectului.

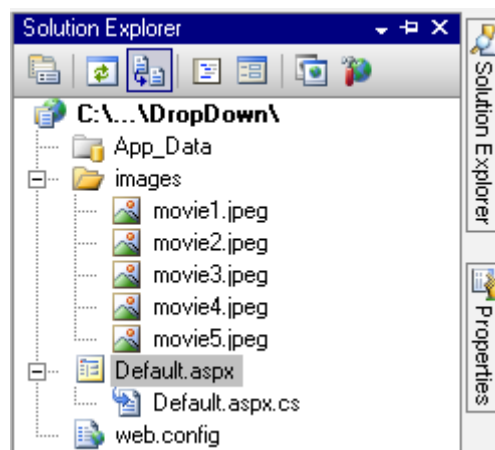


Figura 4.10 Fișierele cu imagini adăugate în directorul images din cadrul proiectului

Atributul *value* pentru controlul DropDownList reține numele fișierului care conține imaginea, iar atributul *text* reține numele filmului, care va fi folosit pentru proprietatea *AlternateText*:

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem Value="movie1.jpeg">Jaws</asp:ListItem>
    <asp:ListItem Value="movie2.jpeg">Casablanca</asp:ListItem>
    <asp:ListItem Value="movie3.jpeg">Breakfast at
Tiffany&#39;s</asp:ListItem>
    <asp:ListItem Value="movie4.jpeg">Butch & amp; The Kid are
back</asp:ListItem>
    <asp:ListItem Value="movie5.jpeg">Forrest Gump</asp:ListItem>
</asp:DropDownList>
```

Codul C# care schimbă imaginea în funcție de filmul selectat de utilizator este:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Image1.ImageUrl = "images/" + DropDownList1.SelectedItem.Value;
        Image1.AlternateText = DropDownList1.SelectedItem.Text;
    }
}
```

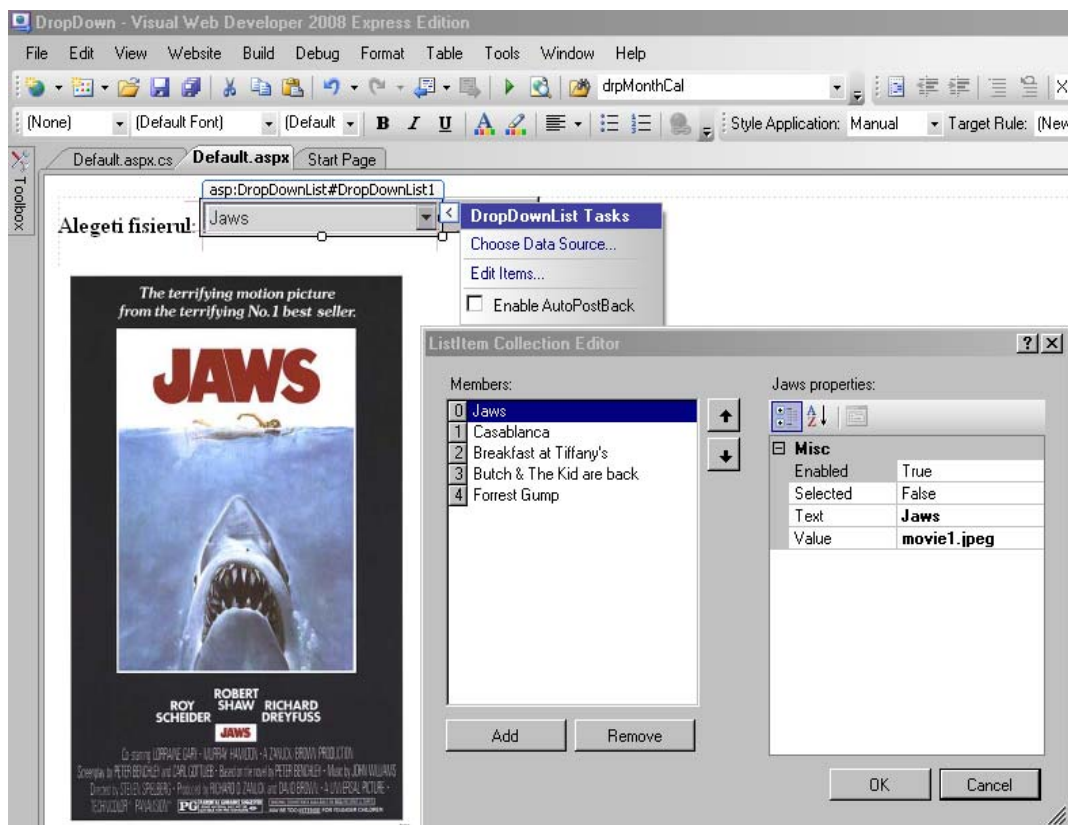


Figura 4.11 Adăugarea de valori pentru controlul DropDownList, în modul design

IV.1.10. HyperLink

Este un control folosit pentru afișarea unui hyperlink în pagina web. Are proprietățile *Text* pentru șirul de caractere ce va fi afișat în pagină, respectiv *NavigateUrl* pentru adresa http. Modul de deschidere a link-ului în browser, poate fi specificat prin intermediul proprietății *Target*, care are valorile: *_blank*, *_parent*, *_search*, *_self*, *_top*.

```
<asp:HyperLink ID="HyperLink1" runat="server"
    NavigateUrl="http://www.microsoft.com" Target="_blank">www.microsoft.com
</asp:HyperLink>
```

IV.1.11. Table, TableRow, TableCell

Aceste controale permit construirea unui tabel Html. Controlul Table are proprietatea *Rows*, care definește liniile tabelului (controale TableRow). La rândul său, Rows are proprietatea *Cells* (controale TableCell), care definește celulele liniei. Șirul de caractere care va fi afișat în celulă este definit de proprietatea *Text* a controlului *TableCell*.

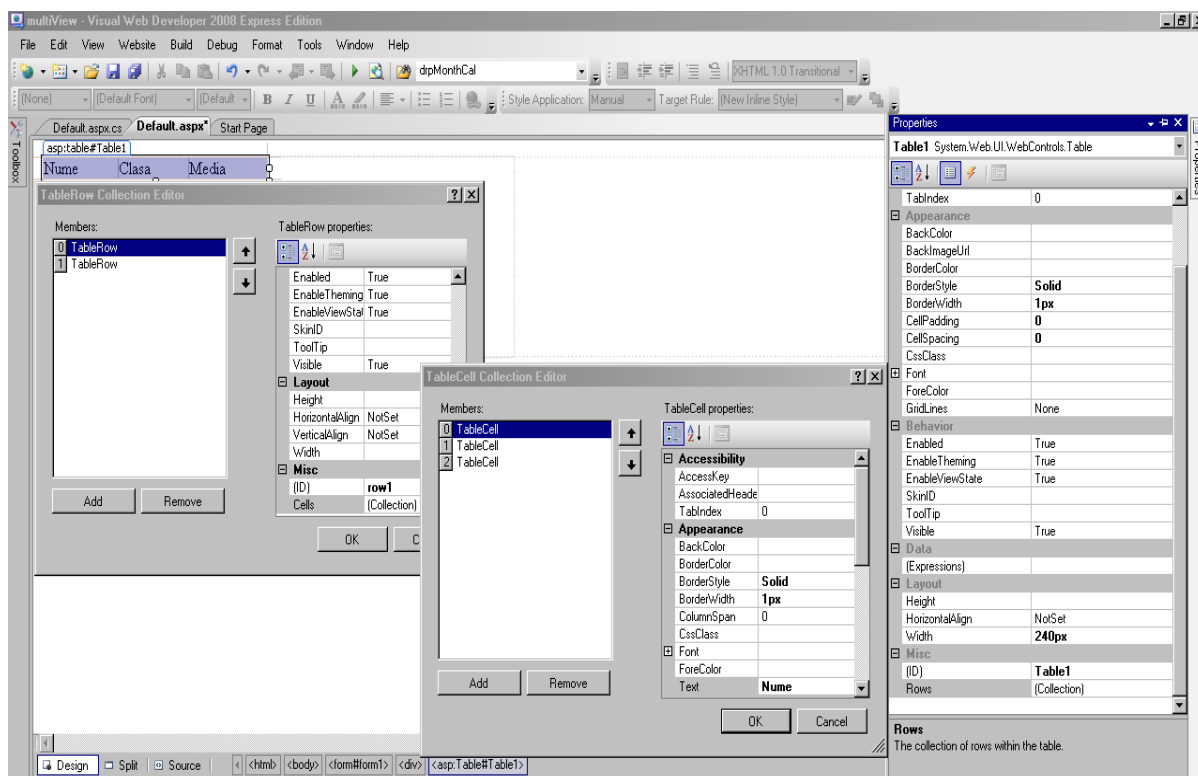


Figura 4.12 Adăugarea controalelor Table, TableRow, TableCell, în modul design

Codul ASP care va fi generat în cazul exemplului de mai sus este:

```
<asp:Table ID="Table1" runat="server" BorderStyle="Solid" BorderWidth="1px"
    CellPadding="0" CellSpacing="0" Width="240px">
  <asp:TableRow ID="row1" runat="server" BorderStyle="Solid" BorderWidth="1px">
    <asp:TableCell runat="server" BorderStyle="Solid" BorderWidth="1px">
      Nume
    </asp:TableCell>
    <asp:TableCell runat="server" BorderStyle="Solid" BorderWidth="1px">
      Clasa
    </asp:TableCell>
    <asp:TableCell runat="server" BorderStyle="Solid" BorderWidth="1px">
      Media
    </asp:TableCell>
  </asp:TableRow>
</asp:Table>
```

IV.1.12. MultiView, View

Controlul MultiView acționează ca un container pentru controale de tip View. Controalele View sunt la rândul lor părinții altor controale web. În funcție de anumite criterii, un control View poate fi afișat la un moment dat în pagina web.

Exemplul 4.8 În următorul exemplu este prezentat un control MultiView care conține trei controale de tip View. Primul control View conține o casetă de text și un buton, al doilea View un are asociat un control hyperlink, iar al treilea un tabel. În acest exemplu, pentru afișarea succesivă a controalelor dintr-un View, se folosește un control BulletList, în care proprietatea *DisplayMode* are valoarea *LinkButton*.

Proprietatea *ActiveViewIndex* definește indexul controlului View care este afișat în pagina web. (primul control View are indicele 0).

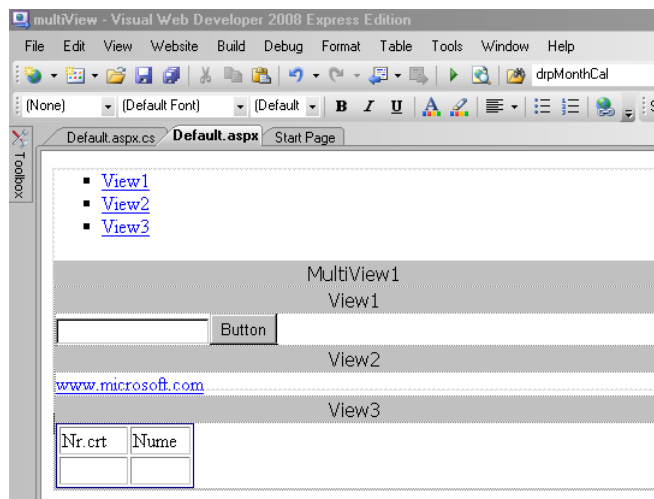


Figura 4.13 Adăugarea controalelor MultiView, View, în modul design

Controlul BulletList are asociat evenimentul *Click*. Funcția care se execută la apariția acestui eveniment, are ca parametru un obiect de tipul *BulletedListEventArgs* cu proprietatea *Index*. Valoarea acestei proprietăți este indexul opțiunii din lista de marcatori care a generat evenimentul click.

```
public partial class _Default : System.Web.UI.Page
{
    protected void BulletedList1_Click1(object sender, BulletedListEventArgs e)
    {
        MultiView1.ActiveViewIndex = Convert.ToInt32(e.Index);
    }
}
```

IV.1.13. FileUpload

Permite upload-ul unui fișier pe server-ul web. Afișează o casetă de text împreună cu un buton Browse, prin intermediul căruia se poate selecta fișierul dorit. Pentru a realiza

efectiv upload-ul, trebuie utilizat un buton (sau alt control) care să conțină codul C# pentru upload.

Controlul FileUpload are următoarele metode și proprietăți:

Nume	Descriere
<i>HasFile</i>	Metodă care returnează true sau false dacă utilizatorul a selectat un fișier pentru upload
<i>SaveAs</i>	Metodă care salvează fișierul selectat pe server. Are ca parametru calea destinație
<i>FileName</i>	Proprietate care reține numele fișierului
<i>PostedFile</i>	Proprietate care conține informații despre fișierul upload-at
<i>ContentType</i>	Proprietate care returnează tipul fișierului upload-at
<i>ContentLength</i>	Proprietate care returnează dimensiunea fișierului upload-at.

Exemplul 4.9 În următorul exemplu, fișierul va fi upload-at într-un folder numit **upload**, construit în cadrul proiectului (din fereastra Solution Explorer).

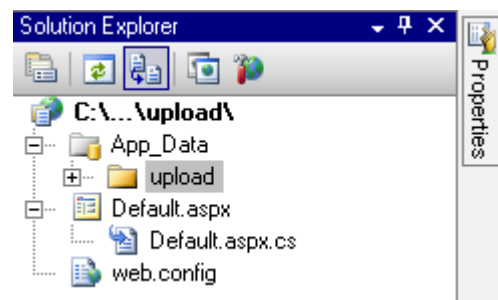


Figura 4.14 Adăugarea directorului upload în cadrul proiectului

Informațiile despre fișierul upload-at vor fi afișate folosind un control de tip Label.

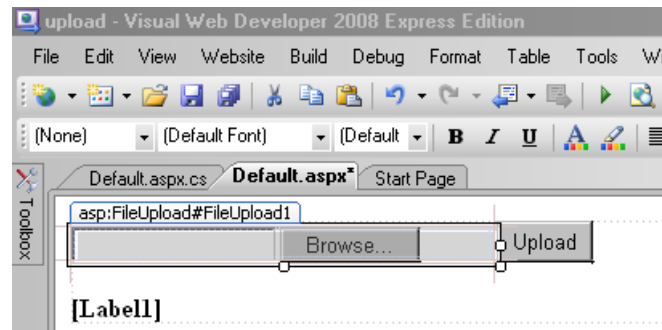


Figura 4.15 Adăugarea controlului FileUpload, în modul design

Fișierul .aspx este:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Untitled Page</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:FileUpload ID="FileUpload1" runat="server" Width="265px" />
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
        PostBackUrl="~/Default.aspx" Text="Upload" /><br />
      </div></form>
      <asp:Label ID="Label1" runat="server" Font-Bold="True"></asp:Label>
    </div></body></html>
```


Codul C# care se execută la apăsarea butonului Upload este cel care realizează efectiv upload-ul fișierului selectat:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile)
        {
            FileUpload1.SaveAs(@"C:\Documents and Settings\User\My Documents\Visual Studio
2008\WebSites\ASP_Projects\FileUpload\App_Data\upload\"+FileUpload1.FileName);
            Label1.Text = "Ati upload-at fisierul: " + FileUpload1.FileName + "<br/>Tip: " +
            FileUpload1.PostedFile.ContentType + " <br/> Dimensiune: " +
            FileUpload1.PostedFile.ContentLength;
        }
        else
        { Label1.Text = "Nu ati selectat nici un fisier";}
    }
}
```



IV.1.14.Evaluare

1. Realizați o pagină web care să conțină un test grilă cu 5 întrebări. Întrebările pot fi cu un singur răspuns sau cu răspunsuri multiple. La apăsarea unui buton numit **Punctaj**, va fi afișat în pagină rezultatul testului (la câte întrebări s-a răspuns corect). La apăsarea unui buton numit **Reset**, se vor reinițializa variantele de răspuns, și se va șterge rezultatul testului precedent.

Indicație: Pentru variantele de răspuns corecte se poate folosi proprietatea *value* a unui element din CheckBoxList sau RadioButtonList.

2. Modificați pagina web de mai sus, astfel încât după selectarea unei variante, aceasta să devină inactivă.

Indicație: Se va modifica proprietatea *enabled*.

3. Realizați o pagină web care conține un test grilă cu 3 întrebări și care îndeplinește următoarele condiții:

- Întrebările sunt afișate prin intermediul unui control BulletList , de tip LinkButton.
- Variantele de răspuns pentru fiecare întrebare sunt poziționate într-un control de tip view al unui control multiview.
- Răspunsurile sunt cu o singură variantă corectă, aceasta având valoarea 1.
- La selectarea unei întrebări, se afișează variantele de răspuns corespunzătoare.
- La apăsarea unui buton **Punctaj** se calculează punctajul final, care va fi afișat prin intermediul unui control Label.

4. Realizați un site web care conține 5 pagini web, fiecare având câte o întrebare a unui test grilă. Testul grilă va avea întrebări cu o singură variantă de răspuns corectă. În fiecare pagină va exista un hyperlink către pagina următoare (cu excepția ultimei).

5. Realizați o pagină web care să afișeze un formular de introducere a informațiilor despre un produs: denumire, preț, unitatea de măsură, data expirării, dacă i se aplică adaos comercial sau nu. Pentru data expirării se va folosi câte un control de tip DropDownList pentru zi, lună an. Data curentă va fi selectată implicit. Valorile pentru DropDownList-ul ce conține anul, vor fi în intervalul [an_curent, an_curent + 10].

6. Realizați o pagină web care permite utilizatorului să facă upload pentru un fișier de tip imagine. Fișierul upload-at va fi afișat printr-un control de tip Image.

7. Realizați o pagină web care va afișa un calculator capabil să efectueze adunări, scăderi, împărțiri, înmulțiri, operația de ridicare la putere, calculul $n!$. Fiecare tastă a calculatorului va fi un control de tip Button.

IV.2. Post Back

Post Back este fenomenul prin care la generarea unui eveniment de către utilizator (client), pagina web este transmisă server-ului, unde se poate executa o secvență de cod care să trateze evenimentul respectiv.

Unele controale generează automat un post back către server. De exemplu controlul Button generează un post back la apăsarea butonului. Alte controale nu generează un post back automat. De exemplu, controlul TextBox are un eveniment numit *TextChanged*. Acesta este generat de fiecare dată când valoarea din caseta de text este modificată. Implicit acest eveniment nu generează un post back. Însă, în cazul în care un alt control generează un post back (de exemplu un buton), în momentul executării codului pe server, este tratat prima dată evenimentul *TextChanged* și apoi evenimentul *Click* al butonului.

Dacă proprietatea *AutoPostBack* are valoarea **true**, atunci controlul respectiv va genera automat un post back.

Exemplul 4.11 Să considerăm în continuare o pagină web în care avem un control de tip CheckBox și un control de tip TextBox care are proprietatea *visible = false*. În momentul în care este bifat checkbox-ul, vrem ca în pagină să apară controlul TextBox. La selectarea controlului CheckBox, se generează evenimentul *CheckedChanged*. Codul C# este:

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    if (CheckBox1.Checked == true)
    {
        TextBox3.Visible = true;
        TextBox3.Focus();
    }
    else
    {
        TextBox3.Visible = false;
    }
}
```

Când vom rula pagina, vom constata totuși că nu este afișat controlul TextBox. Pentru a se executa metoda `CheckBox1_CheckedChanged`, pagina trebuie retrimisă serverului în momentul bifării checkbox-ului. Serverul trebuie să execute codul și apoi să retrimită către browser pagina în care textbox-ul este vizibil. De aceea controlul `CheckBox` trebuie să genereze acțiunea de post back. Pentru aceasta trebuie setată proprietatea `AutoPostBack` cu valoarea `true`.

Exemplul 4.12 O altă situație în care dorim generarea automată a acțiunii de post back este cazul controlului `DropDownList`. În subcapitolul anterior am prezentat o aplicație în care se selecta denumirea unui film dintr-un `DropDownList`, apoi la apăsarea unui buton era afișat posterul filmului respectiv (într-un control de tip `Image`). Vrem să modificăm aplicația, astfel încât afișarea imaginii să se realizeze automat la selectarea filmului în `DropDownList`.

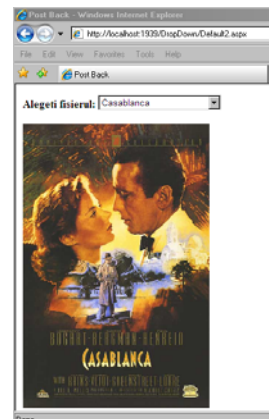


Figura 4.16 Afișarea imaginii se face la selectarea unei opțiuni din controlul `DropDownList`

Pentru aceasta, vom stabili valoarea `true` proprietății `AutoPostBack` pentru controlul `DropDownList`. Apoi, codul care se execută la apăsarea butonului va fi executat la generarea evenimentului `SelectedIndexChanged` pentru controlul `DropDownList`.

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    Image1.ImageUrl = "images/" +
        DropDownList1.SelectedItem.Value;
    Image1.AlternateText = DropDownList1.SelectedItem.Text;
}
```

Obiectul `Page` asociat fiecărei pagini web poate fi folosit pentru a verifica dacă a avut loc o acțiune de post back, sau este prima încărcare a paginii. Astfel, proprietatea `IsPostBack` are valoarea `true` dacă un control a generat o acțiune de post back, și `false` în caz contrar.

Exemplul 4.13 Iată un exemplu în care avem nevoie de această proprietate. Considerăm o pagină web în care avem un control TextBox inițializat cu valoarea 0. La apăsarea unui buton dorim incrementarea valorii din TextBox.

Codul asp este următorul:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>PostBack</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:TextBox ID="TextBox1" runat="server" ReadOnly="True">
      </asp:TextBox>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click1" Text="+"/>
    </div></form></body></html>
```

Exemplul 4.14 Pentru a ști când să realizăm inițializarea valorii din TextBox cu 0, folosim proprietatea *IsPostBack*:

```
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
    if (Page.IsPostBack == false)
    {
      TextBox1.Text = "0";
    }
  }
  protected void Button1_Click1(object sender, EventArgs e)
  {
    TextBox1.Text = Convert.ToString(Convert.ToInt32(TextBox1.Text) + 1);
  }
}
```



IV.2.1. Evaluare

1. Să se realizeze o pagină web care conține un test grilă, în care întrebările sunt replici celebre din filme, iar variantele de răspuns reprezintă un nume de film. La selectarea răspunsului corect, va fi afișata o imagine care conține poster-ul filmului respectiv. (Inițial controlul Image are proprietatea *visible = false*).

2. Să se realizeze o pagină web care conține un test grilă în care întrebările au o singură variantă de răspuns corectă. Rezultatul testului va fi afișat prin intermediul unui control TextBox, pentru care utilizatorul nu poate modifica valoarea. La prima încărcare a paginii, rezultatul va avea valoarea 0. La fiecare selectare a unei variante de răspuns, se va reactualiza valoarea rezultatului.

IV.3. Controale pentru validarea datelor

În toate aplicațiile web se pune problema validării datelor introduse de utilizator. Cu alte cuvinte, trebuie să ne asigurăm că utilizatorul site-ului nostru introduce numai date corecte în casetele de text care îi sunt puse la dispoziție. De exemplu, dacă într-o pagină web se cere utilizatorului introducerea vârstei, și pentru aceasta îi punem la dispoziție o casetă de text, va fi obligatoriu să ne asigurăm că în acea casetă se pot introduce numai cifre, și că numărul rezultat este încadrat într-un anumit interval. Un alt exemplu, este introducerea unei adrese de email validă din punct de vedere al formatului.

ASP.Net pune la dispoziție controale predefinite pentru validarea datelor introduse în pagina web. Controalele de validare au în comun proprietățile:

- *ErrorMessage* – definește mesajul de eroare care va fi afișat dacă datele nu sunt introduse corect
- *ControlToValidate* – definește controlul pentru care se aplică validarea.

IV.3.1. RequiredFieldValidator

Verifică dacă în caseta de text asociată s-au introdus date de către utilizator. Se folosește pentru formularele de date în care câmpurile sunt obligatorii.

IV.3.2. RangeValidator

Verifică dacă datele introduse în caseta de text asociată fac parte dintr-un anumit interval. Proprietățile *MinimumValue* și *MaximumValue* permit introducerea limitelor intervalului. Tipul valorilor din interval poate fi specificat prin intermediul proprietății *Type*.

IV.3.3. RegularExpressionValidator

Se folosește pentru validarea datelor care trebuie să respecte un anumit format, cum ar fi numere de telefon, adrese de email, etc. Proprietatea *ValidationExpression* permite alegerea unei expresii predefinite.

IV.3.4. CompareValidator

Compară valoarea introdusă în controlul asociat, cu o valoare predefinită. Proprietatea *ValueToCompare* reține această valoare. Tipul de comparare este definit de proprietatea *Operator*, valorile sale posibile fiind: *Equal*, *NotEqual*, *GreaterThan*, *GreaterThanEqual*, *LessThan*, *LessThanEqual*, *DataTypeCheck*. Tipul valorilor comparate este specificat de proprietatea *Type*.

IV.3.5. CustomValidator

Pe lângă aceste controale predefinite, programatorul poate defini reguli proprii de validare, prin intermediul controlului CustomValidator. Validarea se poate realiza prin intermediul unei funcții client-side (folosind `JavaScript`), sau la nivel de server.

Exemplul 4.15 Următoarea aplicație folosește controalele prezentate mai sus pentru validarea datelor introduse de utilizator:

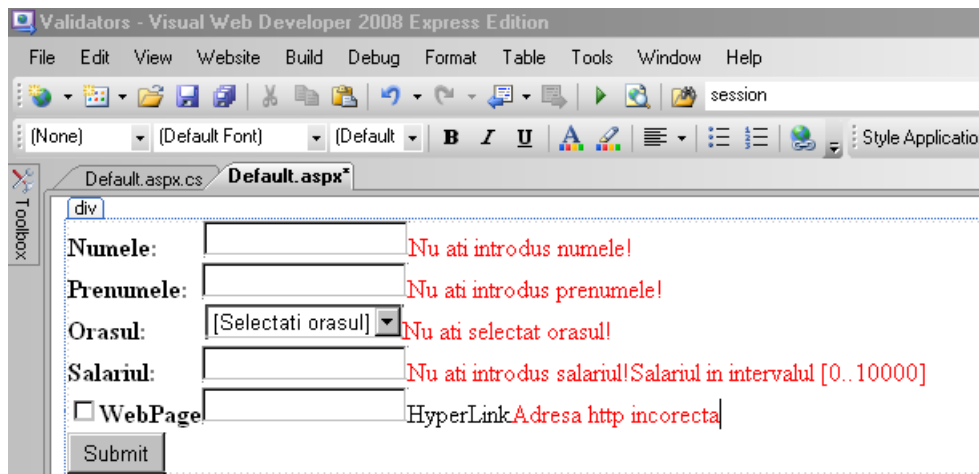


Figura 4.17 Adăugarea controalelor de validare, în modul design

Pentru validarea numelui și prenumelui s-au folosit două controale `RequiredFieldValidator`:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
    ControlToValidate="TextBox1" ErrorMessage="Nu ati introdus numele:">
</asp:RequiredFieldValidator>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
    ControlToValidate="TextBox2" ErrorMessage="Nu ati introdus prenumele!">
</asp:RequiredFieldValidator>
```

Pentru selectarea orașului s-a folosit un control de tip `DropDownList`, pentru care proprietatea `AutoPostBack` are valoarea `true`. Astfel, la selectarea unui element din listă se va realiza automat validarea opțiunii (fără să se aștepte apăsarea butonului de submit). Fiecare element are o valoare asociată prin intermediul proprietății `value`. (prima opțiune are valoarea 0). Validarea se realizează prin intermediul unui control `CompareValidator`:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToValidate="DropDownList1" ErrorMessage="Nu ati selectat orasul!"
    Operator="GreaterThan" Type="Integer" ValueToCompare="0">
</asp:CompareValidator>
```

Dacă opțiunea selectată are valoarea 0, se va genera mesajul de eroare.

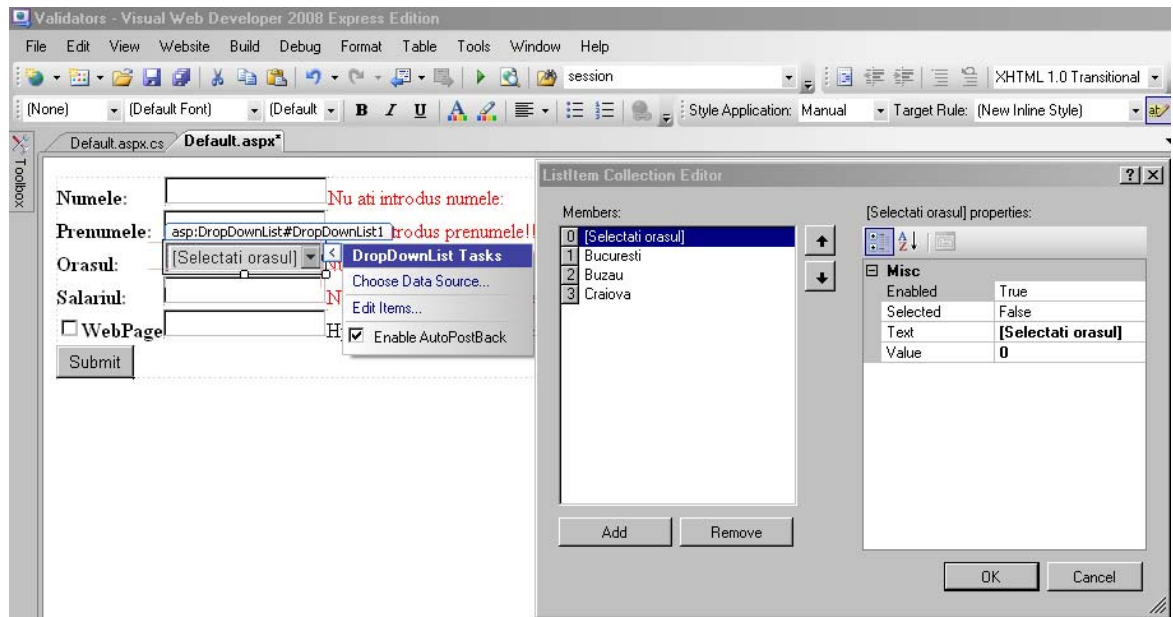


Figura 4.18 Modificarea valorilor returnate de elementele din controlul DropDownList

Salariul este validat prin intermediul a două controale:

- RequiredFieldValidator pentru a verifica dacă s-a introdus o valoare:

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
    ControlToValidate="TextBox4" ErrorMessage="Nu ati introdus salariul!">
</asp:RequiredFieldValidator>
```

- RangeValidator pentru a verifica dacă valoarea introdusă este în intervalul [0..10000]:

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="TextBox4" ErrorMessage="Salariul in intervalul [0..10000]"
    MaximumValue="10000" MinimumValue="0" Type="Integer">
</asp:RangeValidator>
```

La prima încărcare a paginii, controlul TextBox care permite introducerea unei adrese web este inactiv. Pentru aceasta, proprietatea *Enabled* va avea valoarea false. De asemenea, controlul de tip hyperlink nu va fi vizibil inițial pe ecran, proprietatea sa *Visible* având valoarea false.

Exemplul 4.16 Controlul de tip CheckBox are proprietatea *AutoPostBack* = true. Astfel, la bifarea sau debifarea sa, se va realiza o acțiune de post back către server, unde se va executa codul C# corespunzător evenimentului *CheckedChanged*:

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    if (CheckBox1.Checked == true)
    {
        TextBox3.Enabled = true; //controlul TextBox devine activ
        TextBox3.Focus(); //mouse-ul este positionat in textbox
    }
    else
    {
        TextBox3.Text = ""; //se sterge valoarea din TextBox
        TextBox3.Enabled = false; //controlul TextBox devine inactiv
    }
}
```

La introducerea unui șir de caractere în TextBox se generează evenimentul *TextChanged*. La apăsarea butonului Submit se va realiza un post back către server, unde se va realiza codul asociat evenimentului. Aici se va afișa pe ecran hyperlink-ul, iar acesta va trimite către URL-ul introdus de utilizator.

```
protected void TextBox3_TextChanged(object sender, EventArgs e)
{
    if (TextBox3.Text.Trim().Length > 0)
    {
        HyperLink1.Visible = true;
        HyperLink1.Text = TextBox3.Text;
        HyperLink1.NavigateUrl = TextBox3.Text;
    }
    else
    {HyperLink1.Visible = false;}
}
```

Exemplul 4.17 Pentru a verifica dacă valoarea introdusă pentru URL poate fi o adresă validă, se folosește controlul *RegularExpressionValidator*.

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
    ControlToValidate="TextBox3" ErrorMessage="Adresa http incorecta"
    ValidationExpression="http(s)?://([\w-]+\.)+[\w-]+(/[\w-
    ./?%&=]*)?">
</asp:RegularExpressionValidator>
```

Valoarea proprietății *ValidationExpression* a fost aleasă dintr-o listă de formate predefinite.



Figura 4.19 Alegerea unei expresii regulate pentru controlul *RegularExpressionValidator*.

La apăsarea butonului Submit, dacă toate validările sunt îndeplinite, se va realiza o acțiune de redirect către adresa web introdusă de utilizator (dacă aceasta există). Dacă utilizatorul nu a introdus nici o adresă web în controlul TextBox, se va realiza o acțiune de redirect către a doua pagină a proiectului (Default2.aspx), unde se afișează prin intermediul unui control Label mesajul „Datele s-au introdus cu succes!”. În practică, dacă datele sunt corecte, se va apela o interogare SQL pentru inserarea acestor date într-o tabelă, după care se poate realiza un redirect către o altă pagină web.

Pentru a verifica dacă toate condițiile sunt îndeplinite, se apelează metoda *IsValid* a obiectului *Page*, care returnează valorile true sau false.

Realizarea unui redirect către o altă pagină web, se realizează prin intermediul metodei *Redirect* a obiectului *Response*, care primește ca parametru un șir de caractere ce reprezintă adresa paginii.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (Page.IsValid == true)
    {
        if (CheckBox1.Checked == true && HyperLink1.NavigateUrl != null)
        {
            Response.Redirect(HyperLink1.NavigateUrl);
        }
        else
        {
            Response.Redirect("Default2.aspx");
        }
    }
}
```



IV.3.6. Evaluare

1. Realizați un formular pentru introducerea datelor unui angajat: nume, prenume, adresa, telefonul, salariul, data nașterii, funcția, vechimea în muncă, adresa de e-mail. Toate câmpurile sunt obligatorii, cu excepția adresei de email. Se vor realiza următoarele validări:

- Salariul mai mare ca 0, și mai mic ca 20000.
- Vechimea în muncă mai mare ca 0, și mai mică decât 50.
- Telefonul și adresa de e-mail trebuie să fie introduse corect.
- Pentru funcție se va folosi un control DropDownList, în care prima opțiune este mesajul: „Selectați funcția”.

IV.4. Controale server web avansate.

IV.4.1. ImageMap

Este un control care permite afișarea unei imagini care conține zone speciale numite *hot spots*. Când utilizatorul apasă un click cu mouse-ul într-o astfel de regiune se poate realiza una din următoarele acțiuni:

- controlul trimite pagina înapoi către server unde se poate executa un anumit cod (se generează un **postBack**)
- utilizatorul este redirectat către o altă pagină (hyperlink).

Proprietatea *ImageUrl* definește calea către fișierul care conține imaginea. Proprietatea *HotSpots* este o colecție de obiecte de tip *HotSpot*. Un obiect *HotSpot* poate fi de formă circulară, rectangulară sau poligonală. Orice *HotSpot* are proprietățile:

Nume	Descriere
AlternateText	Textul care apare când user-ul poziționează mouse-ul peste hot spot
HotSpotMode	Poate avea una din valorile: NotSet, None,PostBack, Navigate. Dacă se alege PostBack, la apăsarea hotSpot-ului se realizează un post back către server. În cazul opțiunii Navigate, utilizatorul este redirectat către o altă pagină.
NavigateUrl	Reține URL-ul paginii către care se realizează redirectarea
PostBackValue	Definește o valoare care va returnată server-ului în momentul când se realizează un post back
Target	Definește modul în care este deschisă pagina web către care se realizează redirectarea

În funcție de tipul de tip, un *HotSpot* poate avea și alte proprietăți.

Tip HotSpot	Proprietăți
Circle	X, Y, Radius – coordonatele centrului cercului, și raza acestuia
Rectangle	Bottom, Left, Right, Top – coordonatele colțurilor dreptunghiului din stanga jos respectiv dreapta sus
Polygon	Coordinates – listă de perechi de coordonate pentru vârfurile poligonului

Exemplul 4.18 Județul Buzău de pe harta din fișierul [romania.gif](#), va avea asociat un hotSpot de tip *circle*, județul Dolj un hotSpot de tip *rectangle*, iar județul Timiș un hotSpot de tip *polygon*.

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="PostBack"
    ImageUrl="~/Images/romania.gif" onclick="ImageMap1_Click1">
  <asp:CircleHotSpot HotSpotMode="PostBack" AlternateText="Buzau"
    PostBackValue="1" Radius="30" Target="_blank" X="545" Y="370" />
  <asp:RectangleHotSpot AlternateText="Dolj" Bottom="535"
    HotSpotMode="PostBack" Left="236" PostBackValue="2" Right="314"
    Target="_blank" Top="480" />
  <asp:PolygonHotSpot Coordinates="14,261, 65,339, 100,336, 112,319, 173,315,
    165,293" HotSpotMode="PostBack" PostBackValue="3" Target="_blank"
    AlternateText="Timis" />
</asp:ImageMap>
```

În acest exemplu, fiecare HotSpot generează un post back către server cu o anumită valoare. La un click pe hotSpot, se generează evenimentul *Click* pentru controlul ImageMap. Metoda care se va executa pe server, are un parametru de tip *ImageMapEventArgs*. Acest obiect are proprietatea *PostBackValue* care reține valoarea returnată de hotSpot.

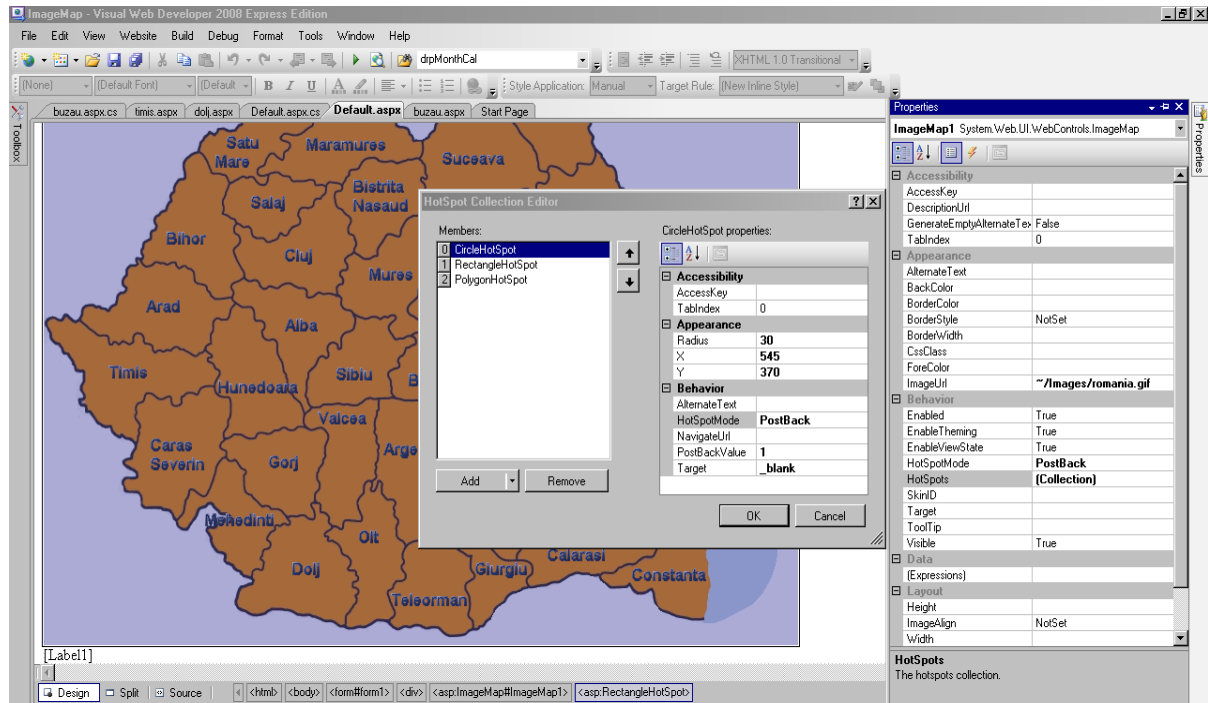


Figura 4.20 Modificarea proprietăților unui obiect HotSpot pentru controlul ImageMap

Codul C# care se execută la apăsarea unui hotSpot este următorul:

```
public partial class _Default : System.Web.UI.Page
{
    protected void ImageMap1_Click1(object sender, ImageMapEventArgs e)
    {
        String judet = "";
        switch (e.PostBackValue)
        {
            case "1": judet = "Ati selectat judetul Buzau"; break;
            case "2": judet = "Ati selectat judetul Dolj"; break;
            case "3": judet = "Ati selectat judetul Timis"; break;
        }
        Label1.Text = judet;
    }
}
```

Exemplul 4.19 A doua posibilitate de utilizare a controlului ImageMap este prin intermediul unui HotSpot care redirectează către o altă pagină web. În următorul exemplu, în cadrul proiectului s-a adăugat un director Images care conține imagini cu județele Buzău, Dolj, Timis. De asemenea, s-au adăugat încă trei pagini web, fiecare având un control de tip Image, cu o hartă a județului respectiv. De exemplu buzau.aspx are codul:

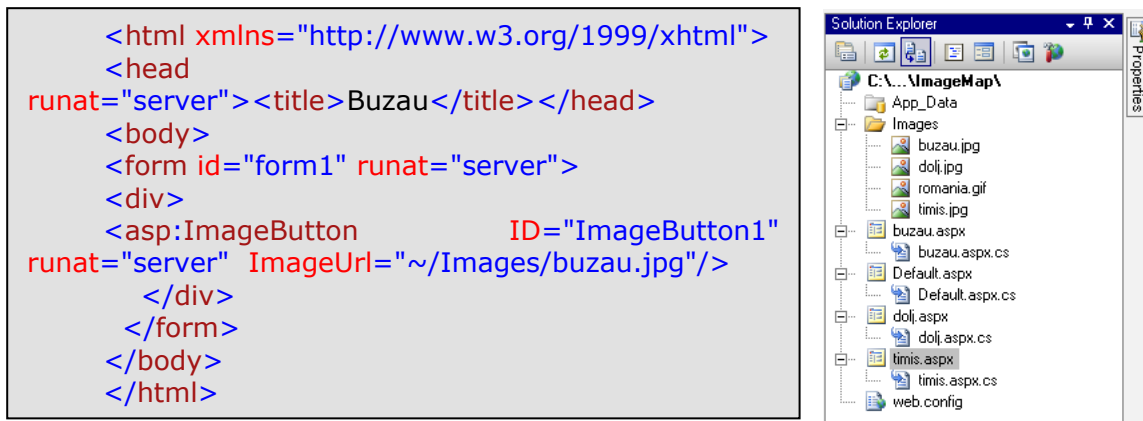


Figura 4.21 Imaginile și paginile .aspx corespunzătoare fiecărui județ

La apăsarea unui hotSpot, se va deschide pagina web care conține imaginea cu județul respectiv.

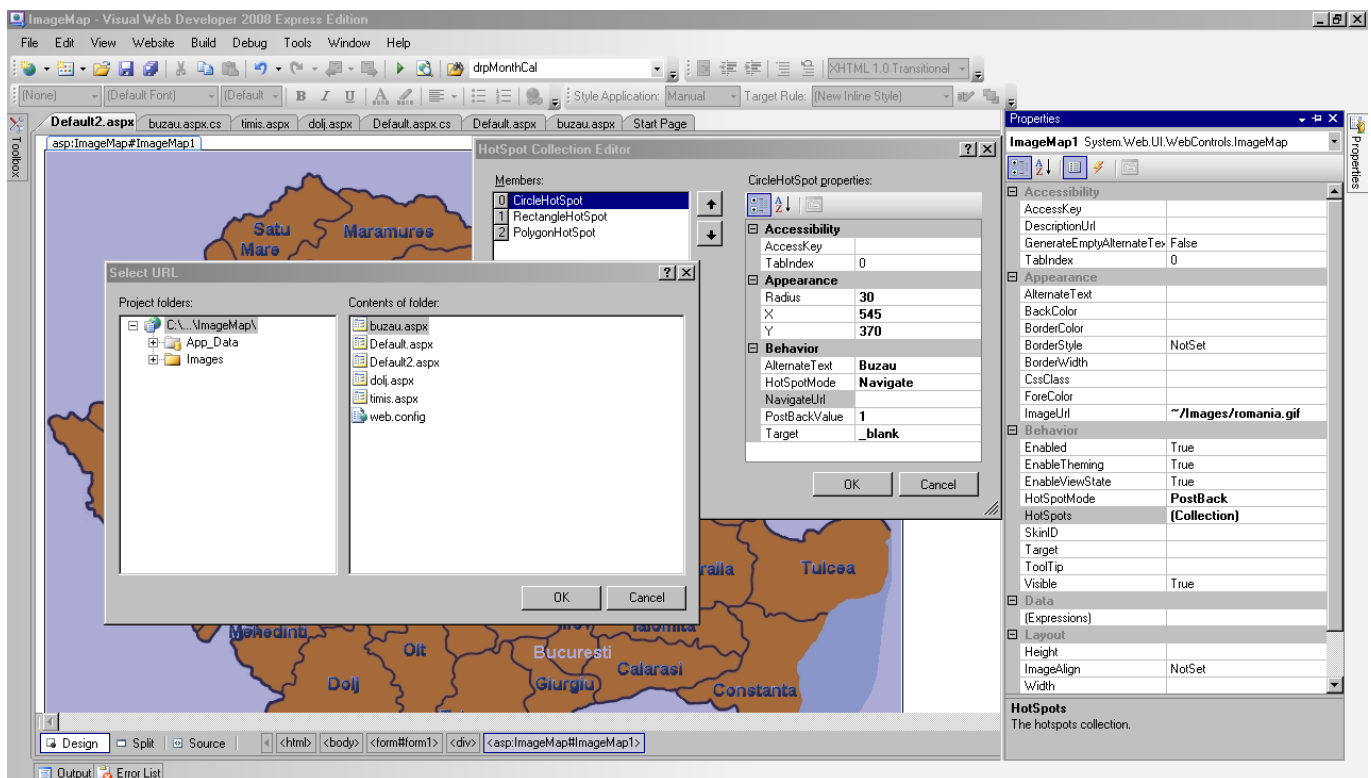


Figura 4.22 Modificarea proprietății NavigateUrl pentru un HotSpot

Fișierul .aspx al paginii principale va conține codul:

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="Navigate"
ImageUrl="~/Images/romania.gif">
<asp:CircleHotSpot HotSpotMode="Navigate"
PostBackValue="1" Radius="30" Target="_blank" X="545" Y="370"
AlternateText="Buzau" NavigateUrl="~/buzau.aspx" />
<asp:RectangleHotSpot AlternateText="Dolj" Bottom="535"
HotSpotMode="Navigate" Left="236" PostBackValue="2" Right="314"
Target="_blank" Top="480" NavigateUrl="~/dolj.aspx" />
<asp:PolygonHotSpot Coordinates="14,261, 65,339, 100,336, 112,319, 173,315,
165,293" HotSpotMode="Navigate" PostBackValue="3" Target="_blank"
AlternateText="Timis" NavigateUrl="~/timis.aspx" />
</asp:ImageMap>
```

IV.4.2. Ad Rotator

Este un control care afișează în pagina web o imagine (de obicei un banner de reclamă), pe care user-ul poate da click, fiind redirectat către o altă pagină web. La fiecare încărcare a paginii web se va fișa o altă imagine aleasă aleator. Fișierele cu imagini împreună cu link-urile de redirect se găsesc într-o sursă de date, de exemplu un fișier XML.

XML este un standard pentru descrierea de marcaje. Un fișier XML are extensia .xml, și conține elemente definite de utilizator, într-o structură ierarhică. Pentru a adăuga un fișier XML în cadrul proiectului, în fereastra **Solution Explorer**, se alege opțiunea **Add New Item**, și apoi **XML File**.

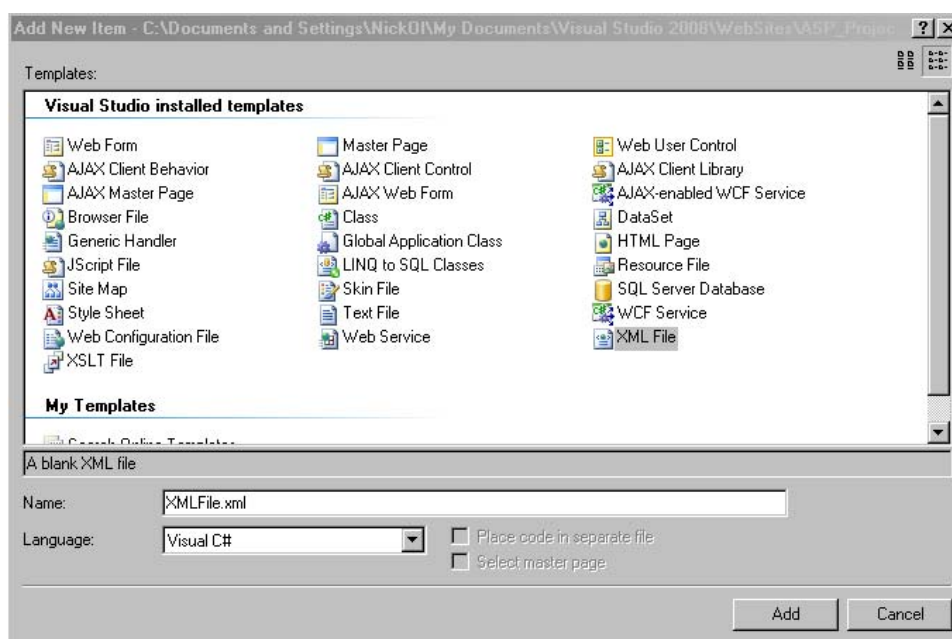


Figura 4.23 Adăugarea unui fișier XML în cadrul proiectului

Pentru a putea fi utilizat ca sursă de date pentru controlul adRotator, fișierul XML trebuie să conțină valori pentru următoarele proprietăți:

Nume	Descriere
ImageUrl	Calea către fișierul care conține imaginea
NavigateUrl	URL-ul către care se realizează redirecarea la un click pe imagine
AlternateText	Textul care apare la poziționarea mouse-ului peste imagine
Keyword	O categorie pentru reclama, după care se pot realiza filtrări
Impressions	Un număr care reprezintă „importanța” reclamei respective. Cu cât este mai mare, cu atât imaginea va fi afișată mai des.

Singura proprietate obligatorie este ImageUrl.

Exemplul 4.20 În următorul exemplu, va fi folosit un fișier ads.xml care conține trei reclame. Proprietățile pentru fiecare reclamă trebuie cuprinse între tag-urile <Ad> și </Ad>. Tag-ul rădăcină pentru fișierul XML este <Advertisements>.

```

<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>~/images/programming.gif</ImageUrl>
    <NavigateUrl>http://www.programmersheaven.com/</NavigateUrl>
    <AlternateText>Limbaje de programare</AlternateText>
    <Keyword>Computers</Keyword>
    <Impressions>80</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>~/images/startrek.gif</ImageUrl>
    <NavigateUrl>http://www.startrek.com/startrek/view/index.html</NavigateUrl>
    <AlternateText>Star Trek Home</AlternateText>
    <Keyword>SF</Keyword>
    <Impressions>20</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>~/images/wallstreet.gif</ImageUrl>
    <NavigateUrl>http://online.wsj.com/</NavigateUrl>
    <AlternateText>Wall Street Journal</AlternateText>
    <Keyword>Finance</Keyword>
    <Impressions>70</Impressions>
  </Ad>
</Advertisements>

```

În proiect au fost adăugate trei fișiere cu imagini, în directorul images:

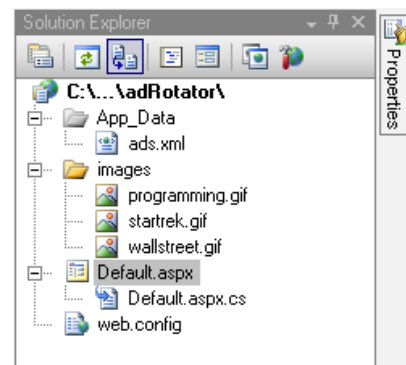


Figura 4.24 Fișierele cu imaginile corespunzătoare bannere-lor.

Exemplul 4.21 După adăugarea controlului în pagina web, trebuie aleasă sursa de date:

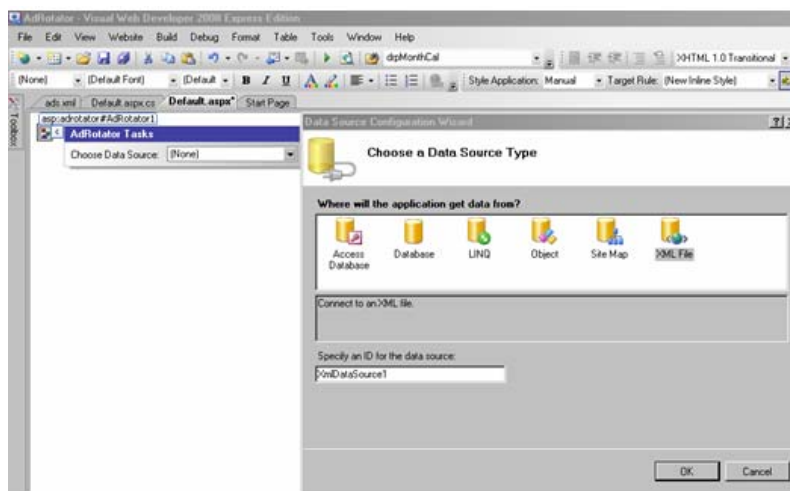


Figura 4.25 Selectarea tipului sursei de date pentru controlul adRotator

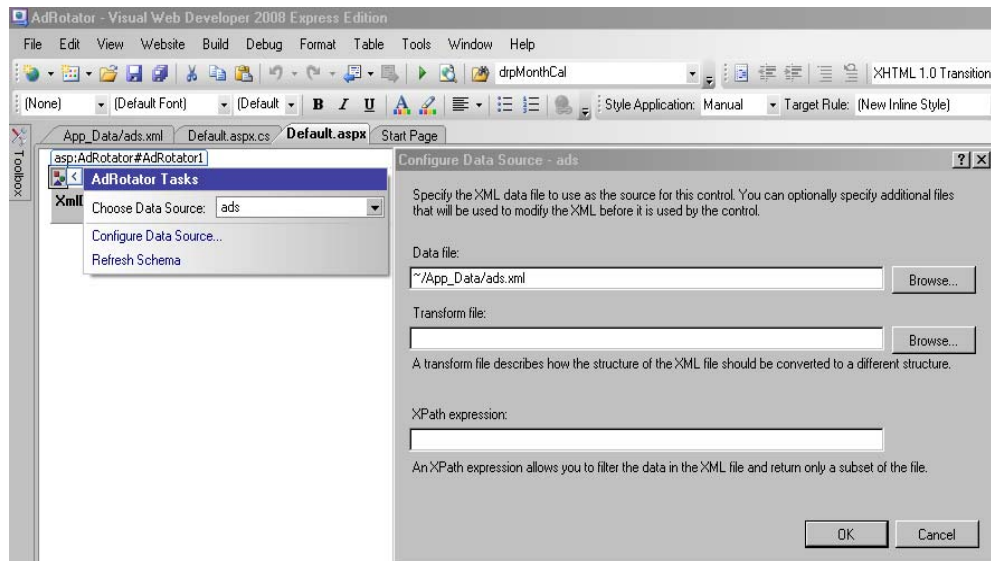


Figura 4.26 Selectarea fișierului XML ca sursă de date

Codul asp generat este următorul:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"> <title>Ad rotator</title> </head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:AdRotator ID="AdRotator1" runat="server" DataSourceID="ads"
        Target="_blank" />
      <asp:XmlDataSource ID="ads" runat="server"
        DataFile="~/App_Data/ads.xml"> </asp:XmlDataSource>
    </div>
  </form></body></html>
```

În urma execuției paginii web, va fi afișată random la reîncărcarea paginii, una din imaginile:

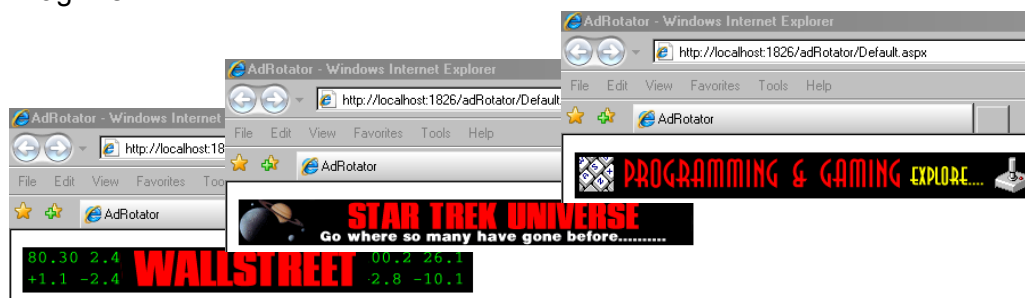


Figura 4.27 Rezultatul execuției pentru o pagină cu un control AdRotator

IV.4.3. Calendar

Este un control care afișează în pagina web un calendar, din care utilizatorul poate alege o dată oarecare. Modul în care utilizatorul poate selecta o dată calendaristică este definit de proprietatea *SelectionMode*, care poate avea valorile:

Valoare	Descriere
Day	Utilizatorul poate selecta doar o zi
DayWeek	Utilizatorul poate selecta o săptămână
DayWeekMonth	Utilizatorul poate selecta o lună
None	Selecția unei date este dezactivată

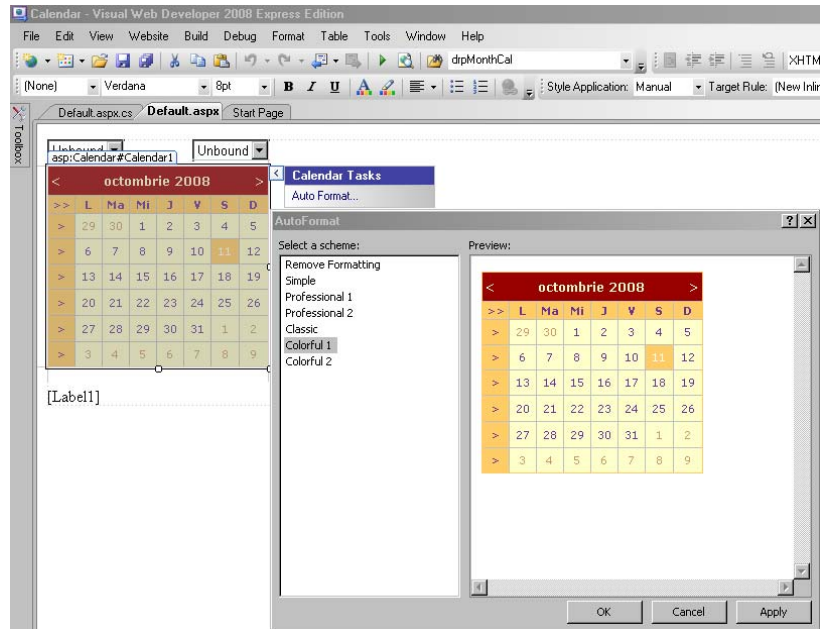


Figura 4.28 Adăugarea unui control Calendar

Controlul de tip calendar are următoarele proprietăți:

Nume	Descriere
TodayDate	Reține data curentă a calendarului
SelectedDate	Data selectată de către utilizator (dacă SelectionMode = „Day”)
SelectedDates	Vector care conține datele selectate de utilizator (dacă SelectionMode are valorile DayWeek sau DayWeekMonth). Vectorul are proprietatea Count care returnează numărul de zile selectate.

La selectarea unei date de către utilizator, se generează evenimentul *SelectionChanged*. La selectarea unei date calendaristice, se realizează automat un post back către server.

Exemplul 4.22 În următorul exemplu, pagina web conține un control Calendar, două controale DropDownList, respectiv un control Label. Cele două controale DropDownList au proprietatea *Id* modificată la valorile *monthCmb* pentru luni, respectiv *yearCmb* pentru ani.

La încărcarea paginii, se realizează următoarele acțiuni:

- primul control DropDownList este inițializat cu lunile anului.
- al doilea control DropDownList este inițializat cu anii din intervalul [an_curent – 10, an_curent + 10].
- în cele două dropDown-uri se selectează luna și anul ce corespund datei curente.
- se afișează data curentă folosind controlul Label.

Popularea celor două DropDown-uri este realizată prin intermediul codului C#. Metodele `monthInit()`, respectiv `yearInit()` folosesc metoda `Add` aplicată vectorului `Items` pentru adăugarea de valori în listă.

După popularea DropDown-urilor, selectarea lunii și anului ce corespund datei curente se realizează prin modificarea proprietății `selected` a unui element din listă. Pentru a determina elementul curent, se apează metoda `FindByValue` pentru vectorul de itemi din DropDown:

```
yearCmb.Items.FindByValue(DateTime.Now.Year.ToString()).Selected = true;
monthCmb.Items[DateTime.Now.Month - 1].Selected = true;
```

Codul C# este următorul:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        monthInit();
        yearInit();
    }
    Label1.Text="Data curenta este: "+Calendar1.TodaysDate.ToShortDateString();
}
private void yearInit()
{
    int firstYear = Convert.ToInt32(DateTime.Now.Year) - 10;
    int lastYear = Convert.ToInt32(DateTime.Now.Year) + 10;
    for (int i = firstYear; i <= lastYear; i++)
    {
        yearCmb.Items.Add(Convert.ToString(i));
    }
    yearCmb.Items.FindByValue(DateTime.Now.Year.ToString()).Selected = true;
}
private void monthInit()
{
    monthCmb.Items.Add("Ianuarie");
    monthCmb.Items.Add("Februarie");
    monthCmb.Items.Add("Martie");
    monthCmb.Items.Add("Aprilie");
    monthCmb.Items.Add("Mai");
    monthCmb.Items.Add("Iunie");
    monthCmb.Items.Add("Iulie");
    monthCmb.Items.Add("August");
    monthCmb.Items.Add("Septembrie");
    monthCmb.Items.Add("Octombrie");
    monthCmb.Items.Add("Noiembrie");
    monthCmb.Items.Add("Decembrie");
    monthCmb.Items[DateTime.Now.Month - 1].Selected = true;
}
```

Exemplul 4.23

Cele două controale DropDownList au setată valoarea `true` pentru proprietatea `AutoPostBack`. Astfel la selecția unui element din listă, se generează un post back către

server, și se execută codul asociat evenimentului *SelectedIndexChanged*. Acest cod modifică data curentă a calendarului, conform lunii și anului selectate de utilizator.

```
protected void monthCmb_SelectedIndexChanged(object sender, EventArgs e)
{ Calendar1.TodaysDate = Convert.ToDateTime(monthCmb.SelectedItem.Value+" 1, "
+ yearCmb.SelectedItem.Value);
}
protected void yearCmb_SelectedIndexChanged(object sender, EventArgs e)
{
Calendar1.TodaysDate =Convert.ToDateTime(monthCmb.SelectedItem.Value+" 1, " +
yearCmb.SelectedItem.Value);
}
```

Când utilizatorul selectează o dată calendaristică, se generează evenimentul *SelectionChanged*. Metoda care tratează acest eveniment, modifică proprietatea *Text* a controlului *Label*, afișând fie ziua selectată, fie prima și ultima zi a perioadei selectate.

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
switch (Calendar1.SelectedDates.Count)
{
case (0): Label1.Text = "Nu ati selectat nici o data calendaristica";break;
case (1): Label1.Text="Ziua selectata:" + Calendar1.SelectedDate.ToShortDateString();break;
case (7): Label1.Text = "Ati selectat o saptamana incepand cu ziua: " +
Calendar1.SelectedDates[0].ToLongDateString()+" pana la " +
Calendar1.SelectedDates[Calendar1.SelectedDates.Count-1].ToLongDateString();break;
default: Label1.Text = "Ati selectat o luna incepand cu ziua: " +
Calendar1.SelectedDate.ToShortDateString(); break;
}
}
```

Exemplul 4.24 Efectul execuției paginii web este următorul:

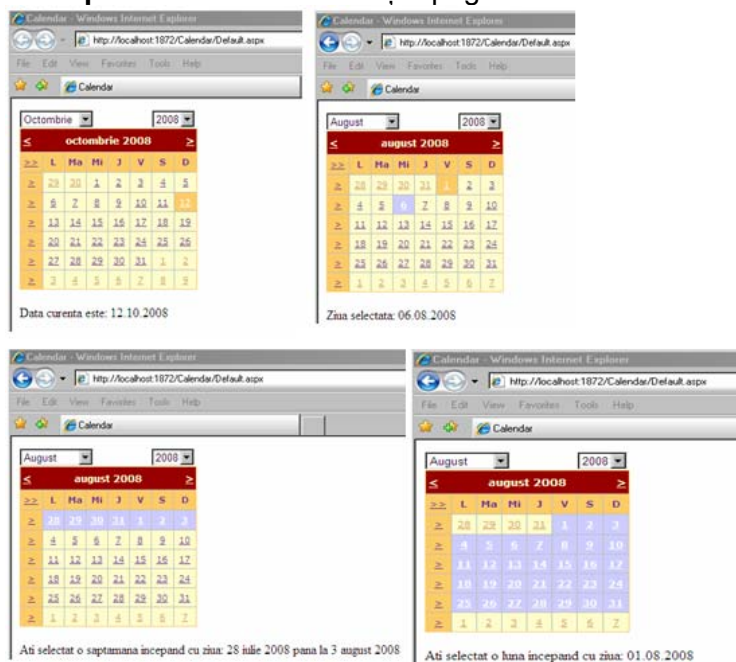


Figura 4.29 Rezultatul execuției, cu diferitele moduri de selectare



IV.4.4. Evaluare

1. Realizați un site web care să conțină informații despre statele uniunii Europene. Pagina de index va conține o imagine cu harta Europei, iar la un click pe o anumită țară se va deschide o pagină nouă, în care sunt afișate informațiile despre țara selectată. De asemenea, în pagina principală trebuie să existe o zonă de header unde va fi afișat un banner cu steagul unei țări la fiecare încărcare a paginii. La un click pe banner se va deschide pagina de internet oficială a țării respective.
2. Realizați o pagină web care conține un formular de introducere a datelor unui angajat. Data nașterii și data angajării trebuie să fie alese prin intermediul unui control calendar.

IV.5. Conectarea la o sursă de date a controalelor

Unele controale prezentate până acum, permit ca datele care le populează să poată fi preluate din diverse surse de date : tablouri, fișiere XML, baze de date. De exemplu controalele de tip „list”: CheckBoxList, RadioButtonList, DropDownList, ListBox, etc.

Proprietățile și metodele care permit conectarea unui control la o sursă de date sunt:

Nume	Descriere
DataSource	Numele sursei de date
DataTextField	Câmpul din sursa de date care va popula proprietatea text a controlului
DataValueField	Câmpul din sursa de date care va popula proprietatea value a controlului
DataBind	Metodă care populează controlul cu datele din sursa de date

Exemplul 4.25

Aplicația următoare folosește un control ListBox pentru care se poate modifica sursa de date prin intermediul unor butoane radio.

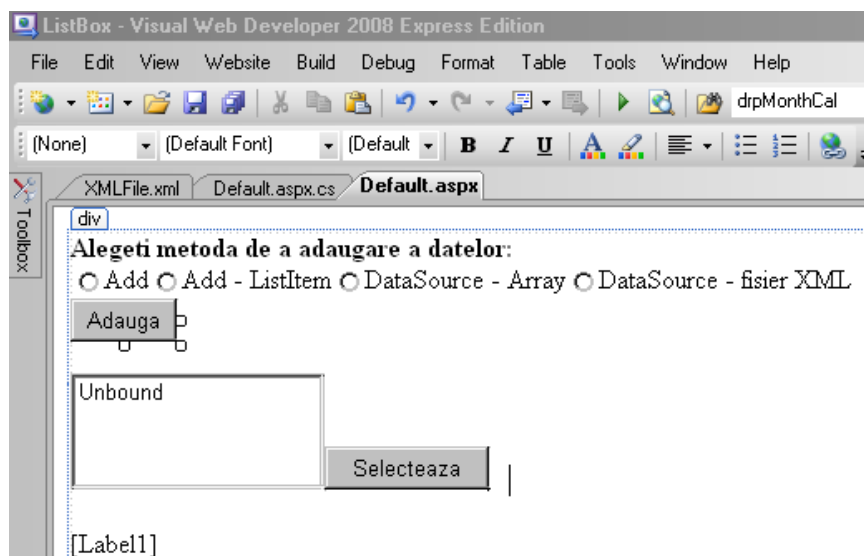


Figura 4.30 Interfața aplicației pentru popularea datelor din controlul ListBox

La apăsarea butonului Adaugă, se folosesc următoarele posibilități pentru a popula obiectul ListBox:

- metoda Add a vectorului *Items* pentru modificarea proprietății *text* a controlului
- metoda Add a vectorului *Item* pentru adăugarea de perechi text / valoare
- folosirea unui vector ca sursă de date
- folosirea unui fișier XML ca sursă de date

Fișierul XML are structura:

```
<?xml version="1.0" encoding="utf-8" ?>
<elevi>
  <elev>
    <nume>Georgescu</nume><media>7</media>
  </elev>
  <elev>
    <nume>Marinescu</nume><media>10</media>
  </elev>
</elevi>
```

Numele este folosit pentru popularea proprietății Text, iar media pentru popularea proprietății Value pentru controlul ListBox.

Inițial butonul Selectează nu este activ (proprietatea Enabled are valoarea false) deoarece nu este nici un element în listă. De fiecare dată când se apasă butonul Adaugă, vechile elemente sunt șterse din listă, și butonul Selectează devine activ. La apăsarea acestui buton, pentru elementul selectat din listă se afișează perechea text / value.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (ListBox1.SelectedIndex > -1)
        {
            Label1.Text = "Elevul " + ListBox1.Items[ListBox1.SelectedIndex].Text +
                " are media " + ListBox1.Items[ListBox1.SelectedIndex].Value;
        }
        else
        {Label1.Text = "Nu ati selectat nici o valoare !";}
    }

    private void add()
    {
        ListBox1.Items.Add("Ionescu");
        ListBox1.Items[0].Value = "10";
        ListBox1.Items.Add("Popescu");
        ListBox1.Items[1].Value = "8";
    }
}
```

```
private void listItem()
{
    ListBox1.Items.Add(new ListItem("IONESCU", "10"));
    ListBox1.Items.Add(new ListItem("POPESCU", "9"));
}

private void vector()
{
    String [] elevi = {"Sorescu", "Dobrinescu"};

    ListBox1.DataSource = elevi;
    ListBox1.DataBind();
    ListBox1.Items[0].Value = "9";
    ListBox1.Items[1].Value = "8";
}

private void xml()
{
    DataSet setDateXml = new DataSet();
    setDateXml.ReadXml(Server.MapPath("XMLFile.xml"));
    ListBox1.DataSource = setDateXml;
    ListBox1.DataTextField = "nume";
    ListBox1.DataValueField = "media";
    ListBox1.DataBind();
}

protected void Button2_Click(object sender, EventArgs e)
{
    Button1.Enabled = true;
    ListBox1.Items.Clear();
    Label1.Text = "";

    switch (RadioButtonList1.SelectedValue)
    {
        case "1": add(); break;
        case "2": listItem(); break;
        case "3": vector(); break;
        case "4": xml(); break;
    }
}
}
```

IV.6.Păstrarea informațiilor între paginile Web

Există o deosebire fundamentală între aplicațiile Windows și cele Web. Anume, în aplicațiile Windows odată creat un obiect, acesta rămâne în memorie până la terminarea aplicației și va putea fi utilizat și din alte ferestre decât cele în care a fost creat, atât timp cât este public. Pe de altă parte, în aplicațiile web paginile nu se păstrează în memorie pe calculatorul utilizatorului (clientului) și astfel apare problema păstrării informațiilor la navigarea între pagini.

Când browserul cere o anumită pagină, ea este încărcată de serverul web, se execută codul asociat pe baza datelor trimise de user, rezultând un răspuns în format html trimis browserului. După ce este prelucrată pagina de către server, obiectele din pagină sunt șterse din memorie, pierzând astfel valorile. De aceea apare întrebarea: cum se salvează/transmit informațiile între paginile unui site web sau chiar în cadrul aceleiași pagini, între două cereri succesive către server?

În funcție de locul în care vor fi salvate datele, există două abordări:

- salvarea datelor pe client
- salvarea datelor pe server-ul web

Salvarea datelor pe client se poate realiza cu ajutorul

- controlului de tip hidden
- obiectului ViewState
- Cookies
- apelului paginilor folosind parametri de tip Get (Query String)

Pentru salvarea datelor pe server, se folosesc obiectele Session respectiv Application.

IV.6.1. Controlul HiddenField

Se folosește pentru a salva date la nivelul paginii web. După cum îi spune și numele, acest control nu este vizibil în pagina web. Valorile câmpurilor hidden sunt trimise server-ului odată cu pagina web, putând stoca date de dimensiuni mici. Nu sunt indicate pentru a reține date importante.

Pentru a reține date, controlul HiddenField folosește proprietatea Value:

```
Hidden1.Value="10";
```

Exemplul 4.26

În următorul exemplu se incrementează cu 1 valoarea reținută în controlul de tip HiddenField. La apăsarea unui buton, se generează o acțiune de post back, iar în evenimentul Load al paginii web se incrementează valoarea. La prima încărcare a paginii, se inițializează valoarea controlului hidden cu 0.

Fișierul .aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server"><title>Hidden</title></head>  
<body>  
  <form id="form1" runat="server">  
    <div>  
      <asp:Label ID="Label1" runat="server"></asp:Label>  
      <asp:Button ID="Button1" runat="server" Text="+" />  
      <asp:HiddenField ID="nr" runat="server" />  
    </div>  
  </form></body></html>
```

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Page.IsPostBack == false)
        {nr.Value = "0";}
        else
        { nr.Value = Convert.ToString(Convert.ToInt32(nr.Value) + 1);}
        Label1.Text = nr.Value;
    }
}
```

Fișierul .aspx.cs

IV.6.2. ViewState

În ASP.Net, toate controalele server web își păstrează starea între două acțiuni de post back. Acest lucru se realizează prin intermediul proprietății *EnableViewState*, care are implicit valoarea true.

Exemplul 4.27 De exemplu, dacă într-o pagină web există un control TextBox și un Buton, iar utilizatorul introduce date în caseta de text și apasă butonul, se generează o acțiune de post back către server. În momentul în care server-ul întoarce rezultatul prelucrării către browser, controlul TextBox își păstrează valoarea.

În momentul generării codului Html de către server, se generează un control html de tip `<input type="hidden"...>`, a cărui valoare este un șir de caractere ce codifică starea controalelor din pagină:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTE1OTg1NDYyNDZkZFCFstl/DwSGv81TuCB397Tk5+CJ" />
```

Datele sunt codificate Base64, ceea ce asigură securitatea, dar crește timpul de încărcare al paginii.

Programatorul poate adăuga valori în obiectul ViewState folosind metoda `Add(cheie, valoare)`:

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (ViewState["dataNasterii"] == null)
    {
        ViewState.Add("dataNasterii", "10/10/2008");
    }
    Label2.Text = "Data de nastere a elevului " + TextBox1.Text + " este: " +
        ViewState["dataNasterii"];
}
```

În exemplul de mai sus, la prima încărcare a paginii în obiectul ViewState este adăugată valoarea "10/10/2008". La orice altă încărcare a paginii, datele sunt preluate din obiectul ViewState prin intermediul cheii `dataNasterii`.

Exemplul 4.28 Incrementarea unei valori la apăsarea unui buton, poate fi realizată folosind obiectul ViewState astfel:

Fișierul.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>View State</title></head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Label ID="Label1" runat="server" Text="__"></asp:Label>
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="+" />
    </div>
  </form></body></html>
```

Fișierul .aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (ViewState["n"] == null)
    {
        ViewState["n"] = "0";
    }
    else
    {
        ViewState["n"] = Convert.ToInt32(ViewState["n"]) + 1;
    }

    if (ViewState["n"].ToString().Length == 1)
    { //dacă valoarea este o cifră, se mai adauga un „0”
        Label1.Text = "0" + ViewState["n"].ToString();
    }
    else
    {
        Label1.Text = ViewState["n"].ToString();
    }
}
```


IV.6.3. Cookies

Un cookies este o secvență de text care este salvată pe calculatorul user-ului. Valorile din cookie sunt perechi de forma cheie/valoare. De obicei, cookie-urile sunt folosite pentru a reține informații despre un user, la vizitarea unei pagini web. Când user-ul revine pe un site, informațiile din cookie sunt citite, ajutând la identificarea rapidă a utilizatorului. Pe lângă valori, pentru un cookie se poate stabili durata de viață în calculatorul clientului.

Pentru a adăuga un cookie, se folosește proprietatea *Cookies* a obiectului *Response*. Această proprietate este un vector asociativ, de tip cheie/valoare:

```
Response.Cookies["cheie"].Value = valoare
```

Timpul de viață al cookie-ului se stabilește prin intermediul proprietății *Expires*.

Avantajul folosirii cookie-urilor este acela că informația este disponibilă în orice pagină la revenirea userului în site, după o anumită perioadă. Printre dezavantaje amintim:

- nu sunt potrivite pentru a reține date importante
- cookie-urile pot fi dezactivate din browser
- sunt încărcate pentru fiecare cerere, ducând la creșterea traficului

Exemplul 4.29

În exemplul următor, datele introduse de utilizator într-o casetă text, sunt salvate într-un cookie. Într-o a doua pagină web, datele sunt extrase din cookie și afișate prin intermediul unui control Label.

Pagina Default.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Cookie</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Submit" />
    </div>
  </form>
</body>
</html>
```

```
public partial class _Default : System.Web.UI.Page
{
  protected void Button1_Click(object sender, EventArgs e)
  {
    if (TextBox1.Text.Trim().Length > 0)
    {
      Response.Cookies["userName"].Value = TextBox1.Text;
      Response.Cookies["userName"].Expires.AddMinutes(10);
      //timpul de viata al cookie-ului este de 10 minute
    }
    Response.Redirect("Default2.aspx");
  }
}
```

L
a
apăsare
a

butonului se salvează datele în obiectul Cookies, și se redirectează către pagina Default2.aspx

Exemplul 4.30

În pagina Default.aspx.cs, datele sunt citite din Cookie și afișate.

```
public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Request.Cookies["userName"] != null)
        {
            //daca exista valori in cookie
            Label1.Text += " " + Request.Cookies["userName"].Value;
        }
        else
        { Label1.Text += " Guest ";}
    }
}
```

IV.6.4. Query String

Se folosește de obicei pentru transmiterea de informații de la o pagină la alta. Un query string e un șir de caractere, care conține adresa unei pagini web, urmată de o serie de parametri sub formă de perechi *nume=valoare*. Perechile de parametri sunt separate de semnul „&”, iar prima pereche este despărțită de adresa paginii web printr-un „?”.

De exemplu: <http://localhost:2881/queryString/Default2.aspx?nume=ion&media=10>

Pentru a extrage datele dintr-un query string, se pot folosi proprietățile *QueryString*, respectiv *Params* ale obiectului *Request*. Ca și în cazul utilizării cookies, aceste proprietăți sunt de fapt vectori asociativi.

În următorul exemplu, datele introduse într-o pagină web sunt folosite pentru construirea unui query string. Acest șir de caractere este de fapt o adresă a unei pagini web, către care se va face o redirectare. Datele sunt introduse prin intermediul a două controale de tip TextBox, iar la apăsarea unui buton se va realiza acțiunea de redirect.

În cea de a doua pagină, valorile sunt extrase din query string și afișate prin intermediul unui control Label.

Codul C# asociat butonului este:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        string redirectUrl = "Default2.aspx?";

        redirectUrl += "nume=" + TextBox1.Text;
        redirectUrl += "&media=" + TextBox2.Text;
        Response.Redirect(redirectUrl);
    }
}
```

Exem

plul 4.32

În evenimentul Load asociat celei de a doua pagini, se realizează prelucrarea valorilor:

```
public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Request.QueryString["nume"] != null)
        {
            Label1.Text = "Nume: " + Request.QueryString["nume"].ToString();
        }

        if (Request.QueryString["media"].Trim() != null)
        {
            Label2.Text = "Media:" + Request.Params["media"].ToString();
        }
    }
}
```

Exemplul 4.33

Pe lângă `Response.Redirect`, pentru a redirecta user-ul către o altă pagină, se poate folosi metoda `Transfer` a obiectului `Server`. De exemplu `Server.Transfer("Default2.aspx");`

Această metodă poate realiza un redirect doar către paginile web de pe server, dar nu și către pagini web externe.

Metoda *Transfer* are un al doilea parametru numit *preserveForm*, care poate avea valorile true sau false. În cazul în care parametrul are valoarea true, query string-ul și valorile controalelor din formularul paginii inițiale, vor putea fi accesate în pagina către care se realizează redirect-ul.

Exemplul de mai sus poate fi rescris folosind metoda *Server.Transfer* astfel:

- la apăsarea butonului din prima pagină, se apelează *Server.Transfer* pentru a realiza redirect-ul către a doua pagină.

```
public partial class Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Server.Transfer("Default2.aspx", true);
    }
}
```

Exemplul 4.34

- în pagina a doua, se folosește proprietatea *Form* a obiectului *Request*, pentru a prelucra informațiile:

```
public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "Nume: " + Request.Form["TextBox1"].Trim().ToString();
        Label2.Text = "Media: " + Request.Form["TextBox2"].Trim().ToString();
    }
}
```

Exemplul 4.35

Form este un vector asociativ, în care id-ul unui control este cheia, iar valoarea este reprezentată de datele introduse în controlul respectiv. *TextBox1* și *TextBox2*, reprezintă id-urile controalelor de tip *TextBox* din prima pagină.

IV.6.5. Session

Obiectul *Session* este creat pe serverul web la prima accesare a sitului de către un utilizator și rămâne în memorie atât timp cât utilizatorul rămâne conectat la site. Pentru a adăuga un obiect în sesiune, trebuie doar să scriem un cod de genul următor:

```
protected void Button1_Click(object sender, EventArgs e)
{
    Session["nume"] = "Hello World!";
}
```

Exemplul 4.36

Session este de fapt un dicționar (listă de perechi cheie – valoare), în care valorile sunt de tip object. Ceea ce înseamnă că la citirea unor valori din sesiune va trebui să realizăm o conversie de tip.

```
protected void Button2_Click(object sender, EventArgs e)
{
    Label1.Text = Session["nume"].ToString();
}
```

Exemplul 4.37

Odată introdus un obiect în Session, el poate fi accesat din toate paginile aplicației, atât timp cât el există acolo. Programatorul poate elimina obiectul din sesiune, folosind metoda *Remove*:

```
Session.Remove("nume");
```

În exemplul următor, sunt create două pagini web. Datele introduse în prima pagină, sunt salvate în sesiune la apăsarea unui buton, după care se realizează un redirect către pagina a doua.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Session.Add("nume", TextBox1.Text);
        Session.Add("media", TextBox2.Text);
        Response.Redirect("Default2.aspx");
    }
}
```

Exemplul 4.38

În a doua pagină, datele sunt extrase din sesiune, și afișate prin intermediul unor controale de tip Label.

```
public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["nume"].ToString().Trim() != null)
        { Label1.Text = "Numele: " + Session["nume"].ToString().Trim(); }

        if (Session["media"].ToString().Trim() != null)
        { Label2.Text = "Media " + Session["media"].ToString().Trim(); }
    }
}
```

Exemplul 4.39

În următoarea aplicație se folosește un control Menu pentru navigarea între paginile web. Există două pagini, prima numită index.aspx, iar cea de a doua info.aspx. În pagina de index se solicită user-ului datele de login. Dacă utilizatorul introduce corect numele și parola, în sesiune se salvează username-ul, și apoi se realizează un redirect către pagina a doua.

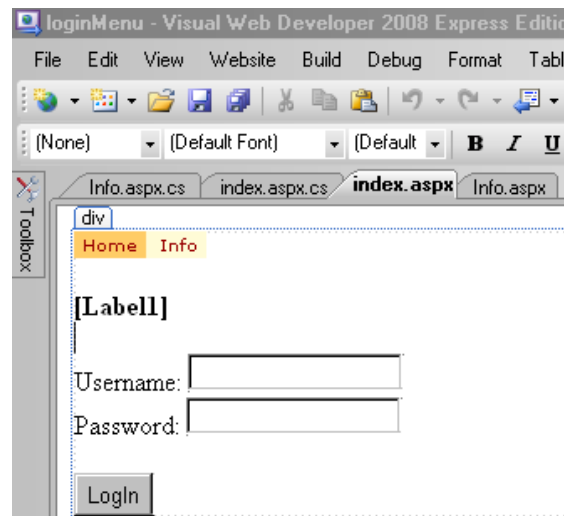


Figura 4.31 Interfața paginii de index

La încărcarea fiecărei pagini, dacă în sesiune se găsește username-ul, se afișează un mesaj de întâmpinare prin intermediul unui control Label.

```

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["user"] != null)
        {
            Label1.Text = "Bine ai venit " + Session["user"];
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (TextBox1.Text.Trim() == "username" && TextBox2.Text.Trim() == "password")
        {
            Session.Add("user", TextBox1.Text.Trim());
            Server.Transfer("Info.aspx");
        }
        else
        {
            Label1.Text = "Username gresit sau parola incorecta !";
        }
    }
}

```

Exemplul 4.40

La accesarea paginii doi prin intermediul meniului, dacă utilizatorul nu este logat (în sesiune nu există cheia „username”), se va afișa un mesaj de eroare.

```
public partial class Info : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["user"] != null)
        {
            Label1.Text = "Bine ai venit " + Session["user"];
        }
        else
        {
            Label1.Text = "Trebuie sa va logati pentru a putea accesa aceasta pagina !";
        }
    }
}
```

IV.6.6. Application

Obiectul Application se comportă în mod identic cu Session, diferența fiind că el este specific întregii aplicații (tuturor utilizatorilor care accesează un site web la un moment dat), și nu unei anumite sesiuni. Cu alte cuvinte odată introdus un obiect în Application, va putea fi accesat din orice loc al site-ului și de către toți utilizatorii acestuia.

Pentru a reține o valoare în obiectul Application, se folosește metoda Set(cheie, valoare), iar pentru extragerea unei valori metoda Get(cheie):

```
Application.Set("counter", 4);
```

```
int x = Convert.ToInt32(Application.Get("counter").ToString());
```

În exemplul următor, ne propunem să realizăm o aplicație care numără câți utilizatori accesează site-ul web la un moment dat, și să afișăm această valoare în fiecare pagină. Pentru aceasta vom adăuga în proiect un fișier numit Global.asax.

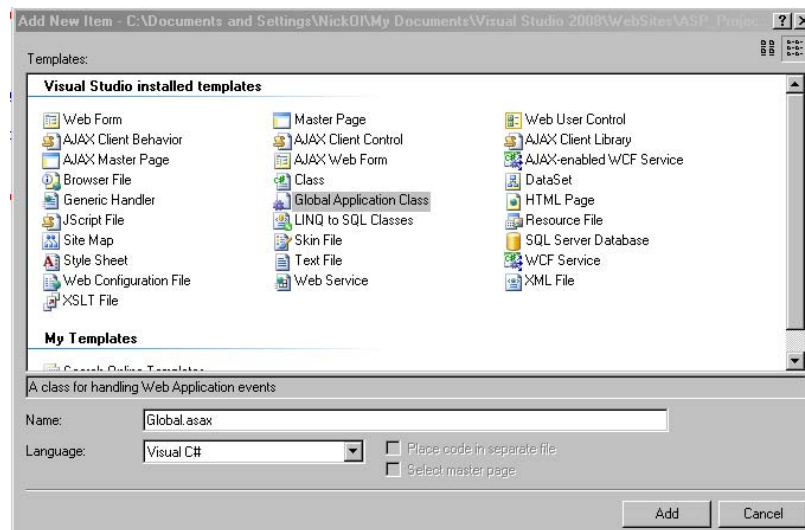


Figura 4.32 Adăugarea fișierului global.asax în cadrul proiectului

Fișierul Global.asax are structura:

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    { // Code that runs on application startup
    }
    void Application_End(object sender, EventArgs e)
    { // Code that runs on application shutdown
    }
    void Application_Error(object sender, EventArgs e)
    { // Code that runs when an unhandled error occurs
    }
    void Session_Start(object sender, EventArgs e)
    { // Code that runs when a new session is started
    }
    void Session_End(object sender, EventArgs e)
    { // Code that runs when a session ends.
    } </script>
```

Exemplul 4.42

Aici se poate scrie cod C# care se va executa la crearea sau distrugerea obiectelor Application, respectiv Session.

În cazul aplicației noastre, la crearea obiectului Application se va reține valoarea 0 într-un element numit „counter”:

```
void Application_Start(object sender, EventArgs e)
{
    // Code that runs on application startup
    Application.Add("counter", 0);
}
```

Exemplul 4.43

Obiectul Session este creat de fiecare dată când un utilizator accesează prima dată site-ul. De aceea, vom incrementa valoarea elementului „counter” la crearea obiectului Session. Decrementarea valorii „counter” se va realiza la distrugerea obiectului Session (când user-ul va închide pagina).

```
void Session_Start(object sender, EventArgs e)
{
    // Code that runs when a new session is started
    int counter = Convert.ToInt32(Application.Get("counter").ToString());
    counter++;
    Application.Set("counter", counter);
}
void Session_End(object sender, EventArgs e)
{
    // Code that runs when a session ends.
    int counter = Convert.ToInt32(Application.Get("counter").ToString());
    counter--;
    Application.Set("counter", counter);
}
```


Exemplul 4.44

Afișarea numărului de utilizatori conectați la un moment dat, se realizează în evenimentul Load al fiecărei pagini:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Page.IsPostBack == false)
        {
            Label1.Text = "Numar de utilizatori conectati:" +
                Application.Get("counter").ToString();
        }
    }
}
```

Exemplul 4.45

La încărcarea unei noi pagini, valoarea din elementul „counter” va fi incrementată, iar la închiderea unei pagini, decrementată.



IV.6.7. Evaluare

1. Să se realizeze un site web care conține un test grilă cu 5 întrebări. Fiecare întrebare va fi afișată într-o pagină separată. Fiecare pagină va avea un hyperlink către pagina precedentă (cu excepția primei pagini) respectiv către pagina următoare. La navigarea între pagini se va păstra răspunsul ales de utilizator la afișarea precedentă. Ultima pagină va avea un link către o pagină de rezultat, în care se va afișa rezultatul testului.

2. Să se modifice aplicația anterioară astfel încât să se folosească o singură pagină. Enunțul întrebărilor și variantele de răspuns vor fi reținute prin intermediul vectorilor. Navigarea între pagini se va realiza în două moduri:

- Cu ajutorul controalelor de tip Button. Indicele paginii curente va fi reținut în sesiune
- Cu ajutorul controalelor de tip hyperlink. Indicele paginii următoare, respectiv precedente va trimis printr-un query string. De exemplu pentru pagina 2 link-urile vor fi de forma: Default.aspx?pagina=1, respectiv Default.aspx?pagina=3

3. Realizați o pagină web care pentru fiecare user afișează prin intermediul unui cookie următoarele informații:

- data ultimei accesări a paginii.
- tipul de browser folosit la ultima accesare.
- ip-ul calculatorului de la care s-a accesat ultima dată pagina.

Indicație. Pentru a afla ip-ul respectiv tipul de browser, se folosește obiectul Request :

- Request.ServerVariables("REMOTE_ADDR") – pentru ip
- Request.ServerVariables("HTTP_USER_AGENT") – pentru tipul de browser

V. INTERACȚIUNEA CU BAZE DE DATE WEB

ADO.NET (ActiveX Data Objects) reprezintă o parte componentă a nucleului .NET Framework ce permite conectarea la surse de date diverse, extragerea, manipularea și actualizarea datelor.

De obicei, sursa de date este o bază de date, dar ar putea de asemenea să fie un fișier text, o foaie Excel, un fișier Access sau un fișier XML.

V.1. ROLUL BAZELOR DE DATE

Astăzi, cele mai multe dintre activitățile noastre zilnice necesită accesarea și actualizarea informației dintr-o bază de date: consultarea unui catalog de produse, lansarea unei comenzi, cumpărarea unui bilet de avion sau verificarea evidenței plăților.

De cele mai multe ori, informațiile pe care le dorim sunt obținute prin prelucrarea unor seturi de date relaționate. De exemplu, o bază de date tipică pentru un magazin on-line conține o listă de clienți, o listă de produse și o listă de vânzări bazată pe informațiile din primele două liste.

Aceste informații sunt descrise cel mai bine utilizând un **model relațional**, model ce presupune divizarea informațiilor în seturi consistente și definirea relațiilor dintre aceste seturi. Modelul relațional stă la baza tuturor produselor moderne axate pe baze de date, incluzând aici SQL Server, Oracle, MySQL și chiar Microsoft Access.

Din punct de vedere tehnic este posibil să organizăm datele în tabele și să păstrăm aceste tabele pe harddisc în unul sau mai multe fișiere (eventual folosind un standard XML). Această abordare nu este însă foarte flexibilă.

O aplicație web are nevoie de un sistem complet de gestionare a bazelor de date (RDBMS⁴¹) cum ar SQL Server, care să asigure gestionarea infrastructurii, performanță și fiabilitate. De exemplu, un astfel de sistem poate furniza date pentru mai mulți utilizatori simultan, elimină datele incorecte și poate executa la un moment dat un grup de comenzi care sunt văzute ca o singură unitate⁴².

Aplicațiile ASP.NET care utilizează baze de date pot fi grupate după tipul operațiilor efectuate. De exemplu:

- site-uri comerciale - păstrează cataloagele cu produse, gestionează comenzile, clienții și tranzacțiile și inventariază informațiile aflate într-un aranjament imens de tabele relaționate;
- motoare de căutare (ca de exemplu Google)-utilizează baze de date pentru a stoca indexuri de pagină URL⁴³, link-uri și cuvinte cheie;
- baze de cunoștințe (ca de exemplu Microsoft Support)-utilizează baze de date care păstrează foarte multe informații și legături către documente sau alte resurse.
- site-uri media -păstrează articolele în bazele de date.

V.2. ACCESAREA BAZELOR DE DATE WEB

Accesarea unei baze de date într-o aplicație web are un scenariu complet diferit față de accesarea unei baze de date prin intermediul unei aplicații desktop client-server.

În aplicațiile tradiționale cu baze de date, clienții stabilesc o conexiune cu baza de date și mențin această conexiune deschisă până la încheierea executării aplicației.

Conexiunile deschise necesită alocarea de resurse sistem. Atunci când menținem mai multe conexiuni deschise server-ul de baze de date va răspunde mai lent la comenzile clienților întrucât cele mai multe baze de date permit un număr foarte mic de conexiuni concurente.

O aplicație Web trebuie să asigure în principal furnizarea rezultatelor către mai mulți utilizatori simultan, operație influențată de resursele de memorie și de conexiunile la baza de date. Dacă aplicația este proiectată astfel încât să mențină conexiunea deschisă chiar și

⁴¹ în limba engleză *Relational Database Management System*

⁴² Această operație se numește tranzacție.

⁴³ (în limba engleză *Uniform Resource Locator*) - adresă pentru localizarea resurselor

pentru câteva secunde în plus pentru un utilizator, atunci ceilalți utilizatori vor observa cu siguranță întârzierea. Concurența este una dintre problemele pe care aplicația trebuie să le gestioneze pentru că modificările făcute de utilizatorii conectați simultan pot conduce la inconsistența datelor.

O altă problemă pe care trebuie să o rezolve o aplicație Web care lucrează cu baze de date este cea a modului de deconectare de la Internet. După cum știm HTTP⁴⁴ este un protocol static. Atunci când un utilizator emite o cerere printr-o aplicație ASP.NET, serverul Web procesează codul, returnează paginile HTML și închide conexiunea, astfel încât utilizatorul are senzația că lucrează cu o aplicație care rulează continuu deși el beneficiază doar de pagini statice. Aplicația ASP.NET trebuie să efectueze operația solicitată de utilizator printr-o singură cerere.

ADO.NET permite și lucrul în stil conectat dar și lucrul în **stil deconectat**, aplicațiile conectându-se la server-ul de baze de date numai pentru extragerea și actualizarea datelor. Acest lucru permite **reducerea numărului de conexiuni deschise** simultan la sursele de date.

ADO.NET oferă instrumentele de utilizare și reprezentare **XML** pentru transferul datelor între aplicații și surse de date, furnizând o reprezentare comună a datelor, ceea ce permite accesarea datelor din diferite surse de diferite tipuri și prelucrarea lor ca entități, fără să fie necesar să convertim explicit datele în format XML sau invers.

Aceste caracteristici sunt determinante în stabilirea beneficiilor furnizate de ADO.NET:

- **Interoperabilitate.** ADO.NET poate interacționa ușor cu orice componentă care suportă XML.
- **Durabilitate.** ADO.NET permite dezvoltarea arhitecturii unei aplicații datorită modului de transfer a datelor între nivelele arhitecturale.
- **Programabilitate.** ADO.NET simplifică programarea pentru diferite task-uri cum ar fi comenzile SQL, ceea ce duce la o creștere a productivității și la o scădere a numărului de erori.
- **Performanță.** Nu mai este necesară conversia explicită a datelor la transferul între aplicații, fapt care duce la creșterea performanțelor acestora.
- **Accesibilitate.** Utilizarea arhitecturii deconectate permite accesul simultan la același set de date. Reducerea numărului de conexiuni deschise simultan determină utilizarea optimă a resurselor.

⁴⁴ (în limba engleză *Hypertext Transfer Protocol*) este metoda cea mai des utilizată pentru accesarea informațiilor păstrate pe servere World Wide Web

V.3. PROIECTAREA BAZELOR DE DATE

Prima etapă în realizarea unei aplicații ce utilizează baze de date este analiza datelor și realizarea unui model conceptual corespunzător.

Această etapă este foarte importantă pentru că pe baza ei se realizează baza de date. Este mult mai simplu să modificăm un model conceptual decât să modificăm o bază de date în care au fost adăugate deja date.

Crearea unui model conceptual presupune o reprezentare grafică a datelor și a relațiilor dintre acestea și este independentă de implementare. Reprezentarea datelor și relațiilor într-o formă convențională se numește **diagramă entități-relații** sau **ERD (Entity Relationship Diagram)**.

V.3.1. Entități, instanțe, atribute, identificator unic

Principalele concepte folosite într-un model conceptual sunt: entitate, atribut, relație.

O **entitate** este un obiect, real sau abstract, pentru care se memorează date și are semnificație pentru problema modelată.

O entitate este reprezentată în ERD printr-un dreptunghi ce are colțurile rotunjite. Numele entității este un substantiv la singular. În figura 5-1 sunt reprezentate două entități (PROFESOR și CLASA).

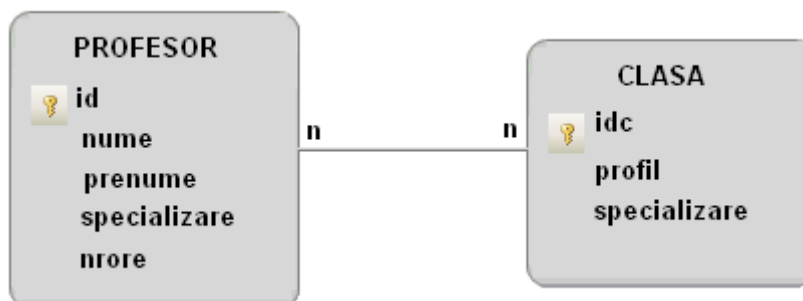


figura V-1

O entitate poate fi privită ca o clasă de obiecte pentru care există mai multe instanțe.

O instanță a unei entități reprezintă un obiect, un eveniment particular, din clasa de obiecte care formează entitatea. De exemplu clasa **9C**, profil **real**, specializare **matematică-informatică** reprezintă o instanță a entității **CLASA**.

Atunci când se precizează o instanță a unei entități se specifică descrierea aceluși obiect. Un **atribut** este o caracteristică a unei entități. În cadrul diagramei entități-relații, atributele unei entități se scriu în dreptunghiul corespunzător entității pe care o caracterizează, imediat sub numele acesteia și sunt substantive la singular.

De exemplu, atributele entității CLASA din figura 5-1 sunt **idc**, **profil** și **specializare**.

Un atribut poate fi obligatoriu sau opțional. Dacă un atribut este obligatoriu, atunci fiecare instanță a acelei entități trebuie să fie caracterizată printr-o valoare a acelui atribut. De exemplu, atributul **nume** din entitatea PROFESOR este un atribut obligatoriu pentru că trebuie să cunoaștem numele fiecărui profesor memorat în baza de date.

În cazul unui atribut opțional pot exista instanțe pentru care să nu se cunoască valoarea atributului respectiv. De exemplu, putem adăuga entității PROFESOR un atribut opțional numit **email**. Această caracteristică (opționalitatea) rezultă din faptul că pot exista profesori care nu au un cont de email.

Instanțele pot fi identificate prin valorile pe care le au atributele ce le caracterizează.

Un atribut sau un set de atribute care identifică în mod unic o instanță a unei entități se numește **identificator unic**. Pentru exemplul din figura 5.1 identificatorul unic al entității CLASA este **idc** (indicativ al clasei). Nu există două clase cu același indicativ deci orice clasă este unic determinată de valoarea acestui atribut. Atributul **profil** nu poate fi ales drept identificator unic pentru că pot exista mai multe clase cu același profil (de exemplu, într-un liceu există mai multe clase cu profil real).

V.3.2. Relații între entități

În lumea reală, obiectele sunt văzute ca elemente componente ale unor sisteme, în interacțiune cu alte obiecte. De exemplu, interacțiunea dintre o instanță **X** a entității PROFESOR și o instanță **Y** a entității CLASA poate fi caracterizată prin expresia: „profesorul **X** predă la clasa **Y**”.

O **relație** este o asociere între două entități (sau între o entitate și ea însăși) și exprimă legătura dintre ele.

Orice relație este caracterizată prin: **nume**, **opționalitate**, **cardinalitate**.

O relație de la o entitate A la o entitate B este **obligatorie** dacă oricărei instanțe a entității A îi este asociată, prin această relație, o instanță a entității B.

O relație de la o entitate A la o entitate B este **opțională** dacă există instanțe ale entității A care nu au asociate, prin această relație, instanțe ale entității B.

Dacă presupunem că entitățile din figura 5-1 se referă la profesorii și clasele unui liceu, atunci relația între cele două entități este obligatorie pentru că orice profesor trebuie să predea la cel puțin o clasă și la orice clasă trebuie să predea cel puțin unul dintre profesori.

Cardinalitatea unei relații indică numărul maxim de instanțe ale fiecărei entități care participă la relație.

Între două entități pot exista relații de tip:

- **1-1 (one-to-one)** – unei instanțe a primei entități îi corespunde cel mult o instanță a celei de-a doua entități;

- **1-n (one-to-many)** – o instanță a primei entități poate fi asociată cu una sau mai multe instanțe ale celeilalte entități;
- **n-n (many-to-many)**- o instanță a primei entități poate fi asociată cu una sau mai multe instanțe ale celei de-a doua entități și pot exista mai multe instanțe ale primei entități asociate unei instanțe a celeilalte entități.

În proiectarea unei baze de date nu se recomandă utilizarea relațiilor de tip n-n.

Rezolvarea relațiilor n-n constă în introducerea unei noi entități numită **entitate de intersecție**, legată de entitățile inițiale prin câte o relație de tip 1-n.

Relația din figura 5-1 poate fi rezolvată prin introducerea entității **INCADRARE** ce preia atributele ce definesc identificatorii unici în cele două entități și are, în plus, atributul **nrore** ce reprezintă numărul de ore pe care le are un profesor la o clasă.

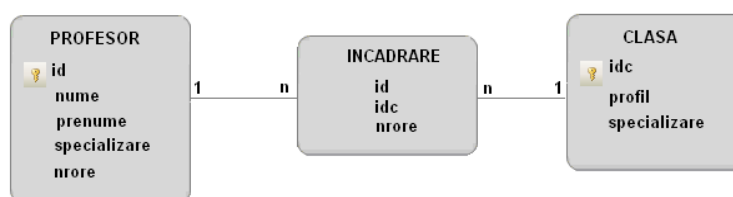


figura V-2



V.3.3. Evaluare

- 1) Dați exemple de activități din lumea reală care presupun accesarea unei baze de date Web.
- 2) Care dintre următoarele activități pot implica accesarea unei baze de date Web ?
 - a. căutarea unui cod poștal
 - b. achiziționarea unui produs
 - c. micșorarea dobânzii bancare
 - d. achitarea facturii telefonice
- 3) Pentru a mări viteza de accesare a unei baze de date Web trebuie să lucrăm în stil
 - a. conectat
 - b. deconectat
- 4) Problema accesului concurent la o bază de date este rezolvată de către
 - a. utilizator
 - b. administratorul bazei de date
 - c. server
 - d. proiectantul bazei de date
- 5) Care dintre următoarele **nu** reprezintă un beneficiu furnizat de ADO.NET?
 - a. performanță
 - b. durabilitate
 - c. interoperabilitate
 - d. concurență
- 6) Care dintre următoarele **nu** poate fi considerat atribut pentru entitatea HOTEL?
 - a. NumărCamere
 - b. PersonalDeIngrijire
 - c. PrețCameră
 - d. Vedere
- 7) Enumerați cinci entitățile care apar în gestionarea activității unui centru comercial. Pentru fiecare dintre aceste entități menționați atributele ce le caracterizează.

8) Realizați asocieri care să corespundă tipurilor de relații :

- | | |
|----------------------|----------|
| a. Elev-Clasă | I. 1-1 |
| b. Mașină-Proprietar | II. 1-n |
| c. Carte-Autor | III. 2-n |
| d. Spectacol-Artist | |
| e. Țară-Președinte | |

9) Între două entități poate exista cel mult o relație.

- | | |
|-------------|---------|
| a. Adevărat | b. Fals |
|-------------|---------|

10) Construiți modelul conceptual pentru următorul scenariu:


„O companie de transport aerian deține mai multe avioane. Fiecare avion zboară pe o rută bine stabilită, după un orar prefixat. Compania are mai multi angajați, printre care și cei ce formează echipajele de zbor. Componenta echipajelor se poate schimba. Un echipaj poate zbura pe mai multe rute și cu mai multe avioane. Fiecare zbor este identificat printr-un număr, aeroportul de unde are loc decolarea, aeroportul unde are loc aterizarea, ora de plecare și durata zborului. Unele zboruri se fac cu escală și atunci se memorează aeroportul unde se face esca, ora sosirii pe acel aeroport și timpul de staționare.”


V.4. CONFIGURAREA BAZEI DE DATE

Pentru a putea executa o comandă care accesează baza de date aveți nevoie de un server de baze de date care să preia și să execute această comandă. Majoritatea aplicațiilor ASP.NET utilizează **Microsoft SQL Server**.

Responsabilitatea creării unei baze de date accesată de către o aplicație Web revine, de cele mai multe ori, dezvoltatorului ASP.NET. Este însă posibil ca aceasta să existe deja sau să fie în responsabilitatea unui administrator dedicat.

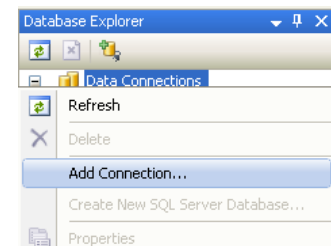
Dacă utilizați o versiune de SQL Server, aveți posibilitatea de a utiliza interfața grafică oferită de SQL Server Management Studio pentru a crea și gestiona baze de date.

Pentru conectarea la bazele de date existente sau pentru a crea altele noi din  **Microsoft Visual Web Developer** , alegeți opțiunea Database Explorer din meniul View și urmați în continuare pașii:

 Faceți click dreapta pe nodul Data Connections, și alegeți **Adăugare conexiune**⁴⁵. În fereastra **Data Source**, selectați sursa de date.

Dacă sursa de date este **Microsoft SQL Server Database File** atunci fișierul accesat va avea extensia **mdf**.

În figura 5-3 se alege o sursă de date Microsoft SQL Server: **.\sqlexpress.master.dbo**.



⁴⁵ în limba engleză **Add Connection**

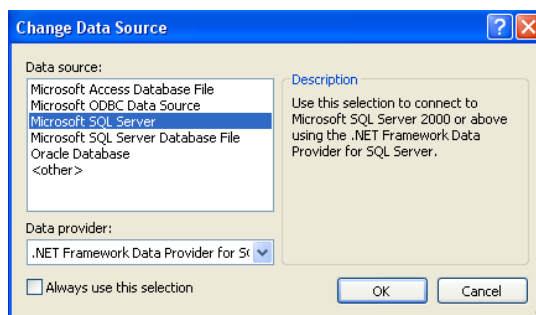


figura V-3 Sursă de date Microsoft SQL Server

În figura 5-4 conectarea se face la baza de date **magazin.mdf**.

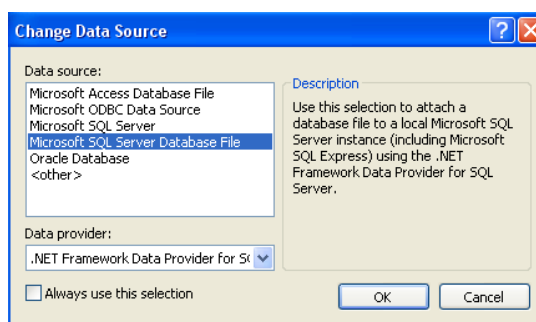


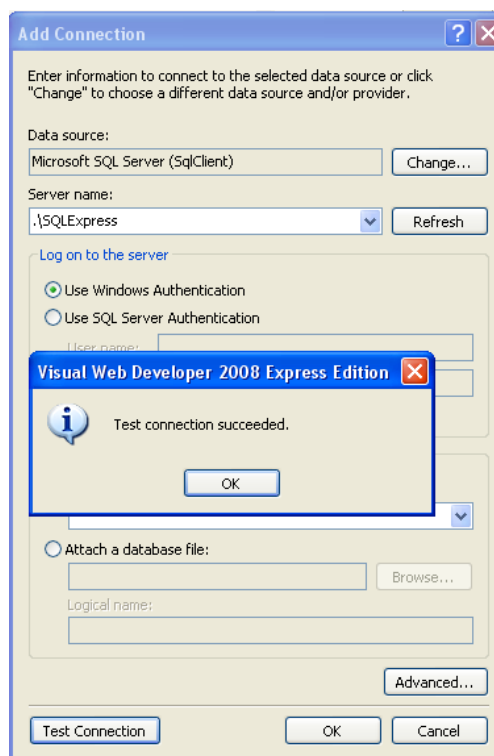


figura V-4 Sursă de date Microsoft SQL Server Database File

 Dacă utilizați o versiune completă de SQL Server, introduceți ca **localhost** numele serverului dacă serverul de baze de date este pe computerul local sau **numele unui computer la distanță** atunci când serverul nu este pe computerul local. Dacă utilizați SQL Server Express trebuie să folosiți numele de server **\\SQLEXPRESS**, după cum se arată în figura alăturată. **SQLEXPRESS** indică faptul că vă conectați la o instanță (cu nume) a SQL Server-ului. În mod implicit, acesta este modul în care SQL Server Express configurează server-ul la prima instalare.

 Faceți click pe **Test Connection** pentru a verifica dacă aceasta este locația unde se găsește baza de date. Dacă nu ați instalat un produs care lucrează cu baze de date acest pas va eșua. Altfel, veți ști că serverul de baze de date este instalat și rulează.





Selectați sau introduceți numele bazei de date⁴⁶

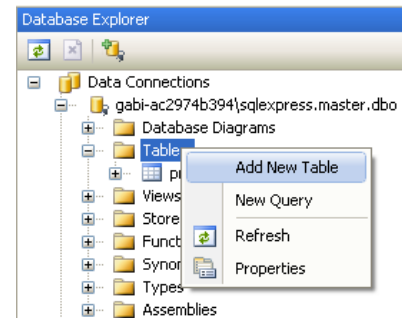
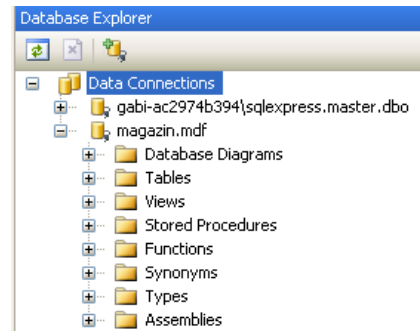
și faceți click pe butonul Ok.

Baza de date apare în fereastra **Database Explorer** și puteți explora grupuri pentru a vedea și edita tabele, proceduri stocate, etc.

Pentru a adăuga o tabelă la o bază de date, alegeți, din fereastra Database Explorer, baza de date corespunzătoare, apăsați click dreapta pe grupul **Tables** și selectați opțiunea **Add New Table**.

Definiți apoi, pentru fiecare dintre câmpurile tabelului, numele, tipul și constrângerile⁴⁷ (ca în exemplul din figura 5-5).

În fereastra **Column Properties** sunt afișate proprietățile câmpului selectat. În această zonă se pot stabili, de exemplu, formule de calcul pentru câmpurile calculate⁴⁸.



Tipul de date asociat unei coloane indică :

- ce fel de date putem scrie pe acea coloană (numere, caractere, imagini etc.);
- lungimea/dimensiunea valorilor stocate;
- precizia/scala (pentru valori numerice)

Tabelul următor conține tipurile de date pe care le puteți asocia coloanelor:

bigint	binary	bit	char	date	datetime	datetimeoffset	decimal
float	geography	geometry	hierarchyid	image	int	money	nchar
ntext	numeric	nvarchar	real	smalldatetime	smallint	smallmoney	sql_variant
text	time	timestamp	tinyint	varbinary	varchar	uniqueidentifier	xml

⁴⁶ În limba engleză **Select or Enter a Database Name**

⁴⁷ Constrângerile permit stabilirea regulilor de integritate care să garanteze că datele introduse în baza de date sunt corecte și valide. Constrângerea **Allow Nulls** atașată unei coloane permite/inhibă prezența valorilor NULL pe acea coloană.

⁴⁸ Formulele se scriu în zona **Formula** din secțiunea **Computed Column Specification**.

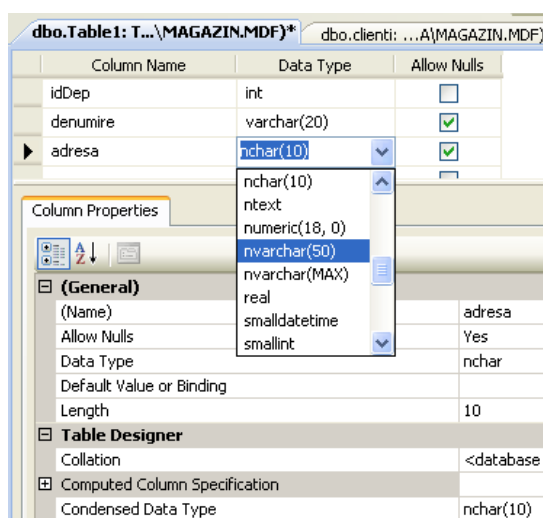
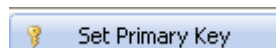


figura V-5 Exemplu de stabilire a structurii tabelii

Pentru a preciza cheia primară⁴⁹ a tabelii, selectați rândul unde doriți să stabiliți cheia primară și apoi executați un click pe butonul din dreapta al mouse-ului și alegeți opțiunea:



. În exemplul din figura 5-6 se stabilește cheia primară **idDep**.

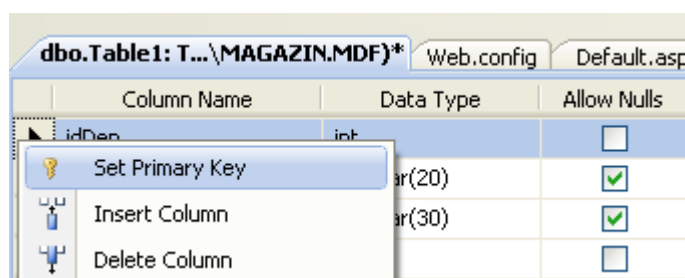


figura V-6

Selectarea uneia dintre opțiunile **Save...** ale meniului File duce la salvarea structurii tabelii curente.

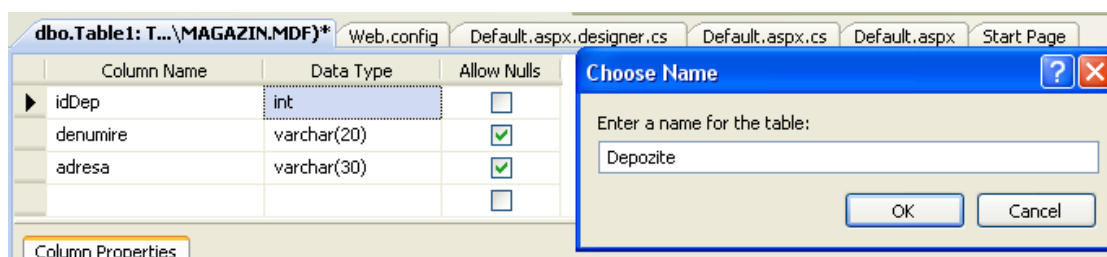


figura V-7 Salvarea tabelii cu numele Depozite

După salvarea definiției, tabela este afișată în zona **Tables** a bazei de date selectate (ca în figura 5-8).

⁴⁹ Identificator unic

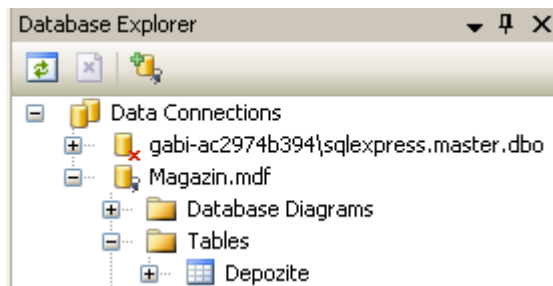
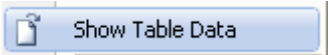


figura V-8

Pentru a introduce sau vizualiza datele din tabelă, executați click dreapta pe numele tabelii și alegeți, din meniul contextual, opțiunea .

Dacă la introducerea datelor în tabelă se încalcă o constrângere (regulă de integritate) atunci suntem avertizați și inserarea eșuează. În exemplul din figura 5-9 a fost încălcată constrângerea dată de cheia primară (Not Allow NULL).

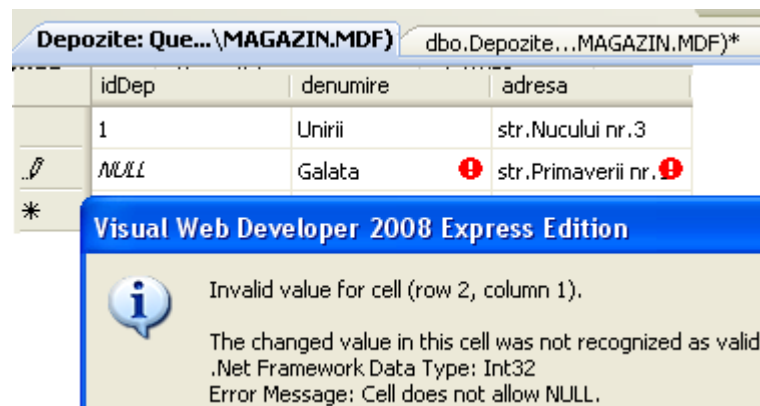


figura V-9

Transformarea **modelului conceptual** într-un **model relațional**⁵⁰ implică transformarea entităților și relațiilor astfel:

- **entitățile** devin **tabele**;
- **atributele** devin **coloane** în tabela provenită din entitatea corespunzătoare;
- **identificatorul unic** devine **cheie primară**;
- **instanțele** devin **linii** în tabelă;
- **relațiile 1-1** devin **chei străine**⁵¹, cheia străină fiind plasată în tabela cu mai puține înregistrări;
- **relațiile 1-n** devin **chei străine plasate în tabela în care se află partea many(n)** a relației;
- **relațiile n-n** sunt rezolvate prin intermediul **entității de intersecție care are două chei străine pentru cele două tabele asociate**.

⁵⁰ Această operație se numește **mapare**.

⁵¹ Cheia străină este un câmp ce corespunde unei chei primare din tabela de referință.

Vom defini, ca bază pentru noțiunile ce urmează să fie prezentate în acest capitol, un **model conceptual simplificat** corespunzător bazei de date **MASTER** ce conține trei tabele (**clienți**, **produse** și **comenzi**). Fiecare client poate lansa una sau mai multe comenzi și fiecare produs poate fi comandat de una sau mai multe ori.

În imaginile următoare avem descrierea și conținutul celor trei tabele.


	Column Name	Data Type	Allow Nulls
	CodClient	int	<input type="checkbox"/>
	NumeClient	nvarchar(50)	<input type="checkbox"/>
	PrenumeClient	nvarchar(30)	<input type="checkbox"/>
	Localitate	nvarchar(30)	<input type="checkbox"/>
	Strada	nvarchar(20)	<input type="checkbox"/>
	Numar	int	<input type="checkbox"/>
	Bloc	nvarchar(5)	<input checked="" type="checkbox"/>
	Scara	char(1)	<input checked="" type="checkbox"/>
	Telefon	numeric(10, 0)	<input type="checkbox"/>

figura V-10 Structura tabelii CLIENTI

CodClient	NumeClient	PrenumeClient	Localitate	Strada	Numar	Bloc	Scara	Telefon
1	Avram	Ionut	Bucuresti	Pacii	2	NULL	NULL	722123456
2	Marcu	Dana	Bacau	Florilor	1	3BIS	A	744515151
3	Andrei	Anca	Bacau	Bradului	3	NULL	NULL	745332211
4	Marin	Marius	Bucuresti	Pacii	5	NULL	NULL	723456791
5	Cerbu	Vlad	Iasi	Dancului	2	123	B	745457885

figura V-11 Conținutul tabelii CLIENTI


	Column Name	Data Type	Allow Nulls
	CodProdus	int	<input type="checkbox"/>
	DenumireProdus	nvarchar(50)	<input type="checkbox"/>
	Categorie	nchar(10)	<input type="checkbox"/>
	PretUnitar	real	<input type="checkbox"/>
	Stoc	int	<input type="checkbox"/>
	Descriere	nvarchar(50)	<input checked="" type="checkbox"/>

figura V-12 Structura tabelii PRODUSE

CodProdus	DenumireProdus	Categorie	PretUnitar	Stoc	Descriere
1	locomotiva	jucarii	123	34	3-5 ani
2	pix	papetarie	15	2300	albastru metalic
3	bicicleta	jucarii	234	87	5-7 ani
4	bicicleta	jucarii	120	61	3-5 ani
5	acuarele	papetarie	5	300	12 culori
6	stilou	papetarie	34	234	InoxCrom
7	papusa	jucarii	89	763	Barbie

figura V-13 Conținutul tabelii PRODUSE


	Column Name	Data Type	Allow Nulls
🔑	IdComanda	int	<input type="checkbox"/>
	CodClient	int	<input type="checkbox"/>
	CodProdus	int	<input type="checkbox"/>
	DataComenzii	date	<input type="checkbox"/>
	NumarProduce	int	<input type="checkbox"/>
	Transmitere	nvarchar(30)	<input checked="" type="checkbox"/>

figura V-14 Structura tabelii COMENZI

IdComanda	CodClient	CodProdus	DataComenzii	NumarProduce	Transmitere
1	1	2	12.11.2008	1	Curier
2	1	4	12.11.2008	1	Curier
3	2	2	10.10.2008	5	PR
4	2	6	07.10.2008	5	PR
5	3	1	08.08.2008	2	Curier
6	3	7	08.08.2008	2	PR
7	3	7	10.10.2008	1	PR
8	4	3	11.11.2008	1	Curier
9	5	5	10.10.2008	2	PR

figura V-15 Conținutul tabelii COMENZI

Relaționarea tabelor este realizată pe baza cheilor străine.

Selectarea opțiunii  Relationships... pe un câmp selectat în fereastra de definire a tabelii permite afișarea relațiilor existente și introducerea altora. Numele unei relații poate fi generat automat și este prefixat de șirul „FK_” urmat de numele celor două tabele. Acest nume poate fi schimbat .

În exemplul din figura 5-16 se definește relația **FK_Comenzi_produce** dintre tabela de referință (părinte) **produce** și tabela (copil) **comenzi** (relație de tip **1-n** – “fiecare produs poate să apară în una sau mai multe comenzi”)

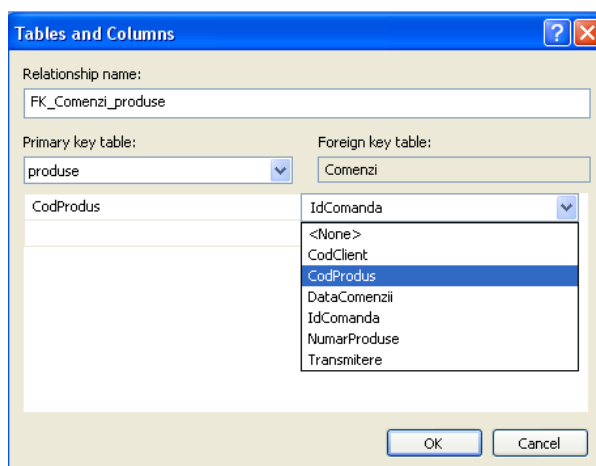


figura V-16

Pentru baza de date aleasă ca model avem două relații de tipul 1-n între tabele:

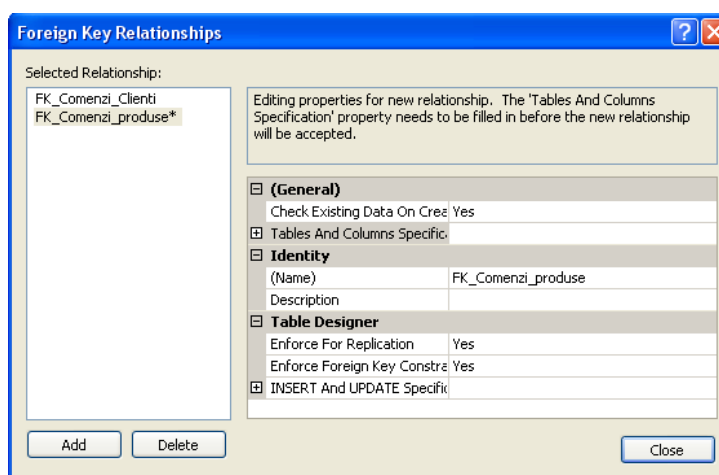
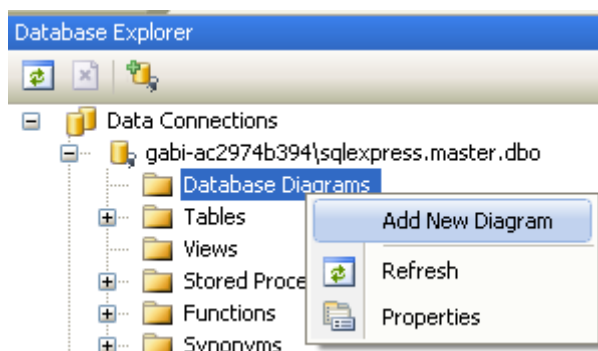
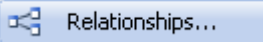


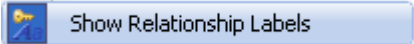
figura V-17

Relația **FK_Comenzi_Clienti** din figura 5-17 marchează faptul că un client poate lansa mai multe comenzi (**CodClient** este **cheie primară** în tabela **Clienti** și **cheie străină** în tabela **Comenzi**), iar relația **FK_Comenzi_Produse** indică faptul că un produs poate să fie comandat de mai multe ori (**CodProdus** este **cheie primară** în tabela **Produce** și **cheie străină** în tabela **Comenzi**)

Opțiunea **Add New Diagram** asociată grupului **Database Diagrams** permite realizarea diagramelor corespunzătoare modelului conceptual ales, pe baza tabelor selectate.



După selectarea tabelelor prezente în diagramă (opțiunea **Add Table**) se stabilesc relațiile între tabele. Pentru stabilirea unei relații între tabele, se apasă click dreapta pe tabelă și se alege opțiunea  și se urmează pașii descriși anterior .

Opțiunea  permite/inhibă afișarea numelui relațiilor.

Pentru modelul ales se obține diagrama din figura 5-18.

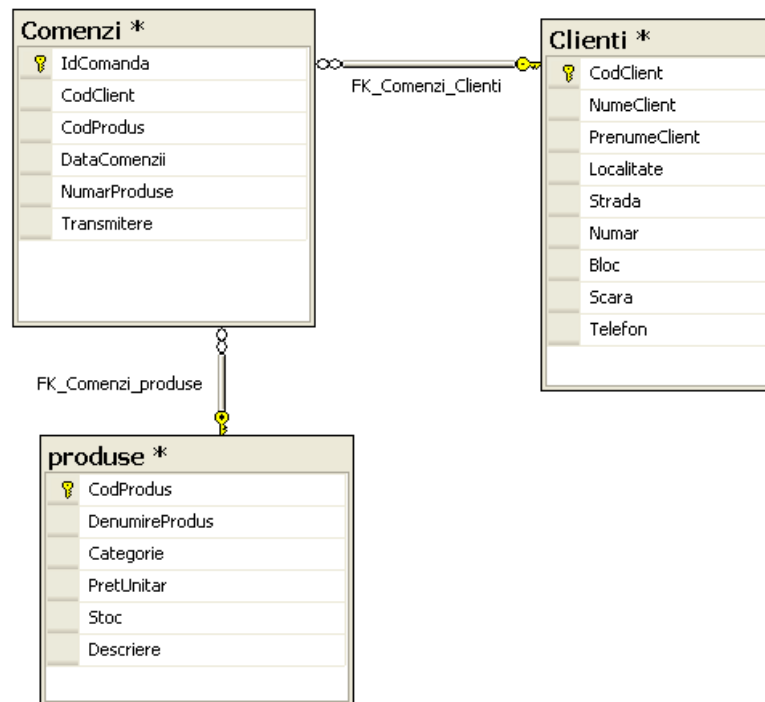
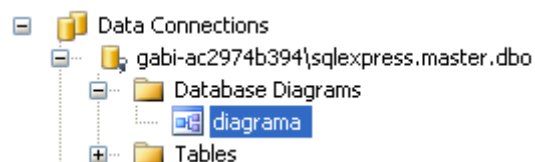


Figura V-18 Diagrama Clienti-Comenzi-Produse

După salvare diagramele sunt păstrate în secțiunea Database Diagrams a bazei de date curente.



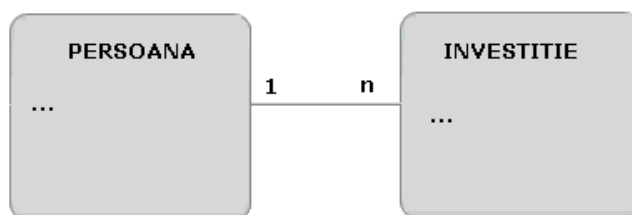
V.4.1. Evaluare

- 1) Într-o tabelă, o linie este obținută prin maparea
 - a. unei relații
 - b. unui atribut

c. unei instanțe

d. unei constrângeri

2) Se mapează relația 1-n dintre tabelele PERSOANA și INVESTITIE.



Atunci

- a. cheia primară din tabela PERSOANE devine cheie primară în tabela INVESTIȚII;
 - b. cheia primară din tabela PERSOANE devine cheie străină în tabela INVESTIȚII;
 - c. cheia primară din tabela INVESTIȚII devine cheie primară în tabela PERSOANE;
 - d. cheia primară din tabela INVESTIȚII devine cheie străină în tabela PERSOANE;
- 3) Atunci când inițiem o conexiune sursa de date la care ne conectăm trebuie să existe.
- a. adevărat
 - b. fals
- 4) Un fișier cu extensia **.mdf** este o bază de date.
- a. adevărat
 - b. fals
- 5) Pentru diagrama din figura următoare



- a. adăugați atribute adecvate celor trei entități
- b. realizați o conexiune către fișierul **cinema.mdf** care va conține cele trei tabele
- c. definiți tabele ACTORI, DISTRIBUTIE, FILME prin maparea entităților din diagramă
- d. mapați relațiile dintre cele trei tabele
- e. adăugați înregistrări în cele trei tabele
- f. construiți diagrama **DActoriFilme** care să conțină reprezentarea celor trei tabele și a relațiilor dintre acestea.

V.5. ACCESUL DIRECT LA DATE

Accesul direct reprezintă cea mai ușoară cale de a interacționa cu o bază de date, ce presupune construirea și executarea unor comenzi SQL.

V.5.1. Limbajul SQL- Elemente de bază

SQL (Structured Query Language) este un limbaj neprocedural pentru interogarea și prelucrarea informațiilor din baza de date. Întrucât este un limbaj declarativ, utilizatorul trebuie să descrie ceea ce trebuie să obțină fără să indice modul de obținere a rezultatului. Compilatorul limbajului SQL generează automat o procedură care accesează baza de date și execută comanda.

Principalele caracteristici prin care limbajului SQL diferă de alte limbaje sunt:



asigură accesarea automată a datelor;



operează asupra unor seturi de date, și nu asupra elementelor individuale;



permite programarea la nivel logic, fără a pune accent pe detaliile implementării.

Instrucțiunile SQL pot fi grupate în mai multe categorii, în funcție de tipul acțiunii pe care o realizează. Unele dintre aceste categorii sunt evidențiate ca limbaje în cadrul SQL, după cum urmează:



DDL⁵² – limbajul de definire a datelor.

Instrucțiunile DDL sunt utilizate pentru definirea structurii obiectelor. Aceste comenzi permit :

- crearea, modificarea, ștergerea obiectelor din bază , inclusiv baza însăși (**CREATE, ALTER, DROP**);
- ștergerea datelor din obiectele bazei, cu păstrarea structurii acestora (**TRUNCATE**)
- redenumirea obiectelor (**RENAME**)
- acordarea / revocarea unor privilegii (**GRANT, REVOKE**)



DML⁵³ – limbajul de manipulare (prelucrare) a datelor.

Instrucțiunile DML sunt utilizate pentru interogarea și prelucrarea obiectelor bazei de date. Aceste comenzi permit :

- selectarea datelor din tabele (**SELECT**);
- adăugarea înregistrărilor în tabele sau vizualizări⁵⁴ (**INSERT**);
- modificarea valorilor unor coloane din înregistrări existente în tabele sau vizualizări (**UPDATE**);
- adăugarea sau actualizarea condiționată a unor înregistrări în tabele sau vizualizări (**MERGE**);
- ștergerea înregistrărilor din tabele sau vizualizări (**DELETE**);

⁵² în limba engleză *Data Definition Language*

⁵³ în limba engleză *Data Manipulation Language*

⁵⁴ O vizualizare (în limba engleză *view*) este o tabelă virtuală, cu rol de filtrare a datelor, care nu memorează date propriu-zise.

 **DCL**⁵⁵ – limbajul de control al datelor.

Instrucțiunile DCL gestionează modificările efectuate de către comenzile DML și grupează aceste comenzi în unități logice, numite tranzacții. Aceste comenzi permit :

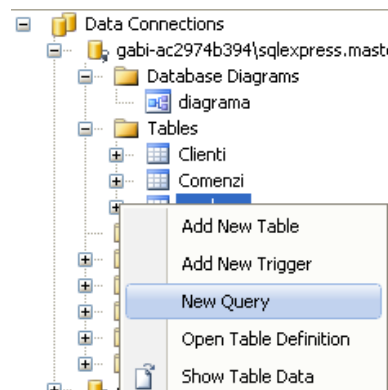
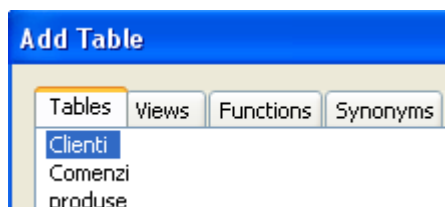
- permanentizarea modificărilor unor tranzacții (**COMMIT**);
- anularea totală sau parțială a modificărilor dintr-o tranzacție (**ROLLBACK**);
- definirea unui punct până la care se poate anula tranzacția (**SAVEPOINT**);

Un produs complex de accesare și prelucrare a bazelor de date așa cum este SQL Server permite scrierea unor secvențe de cod SQL complexe, grupate în proceduri stocate sau declanșatori. Majoritatea aplicațiilor utilizează în principal instrucțiuni **DML**.

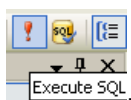
SQL Server 2008 (și 2005) include un instrument numit **sqlcmd.exe** care permite introducerea comenzilor SQL de la linia de comandă⁵⁶. Pentru lansare alegem din meniul Start: **Programs → Microsoft Visual Studio 2008 → Visual Studio Tools → Visual Studio 2008 Command Prompt**.

Mediul de dezvoltare **Visual Studio** dispune de instrumente puternice și sugestive pentru utilizarea bazelor de date în aplicații. Pentru a scrie și a executa în acest mediu o interogare trebuie să urmăm pașii :

- click dreapta pe conexiune și alegem **New Query** ;
- selectăm obiectele utilizate în interogare (tabelele, vizualizări, funcții, sinonime);



- construim interogarea (selectăm câmpurile ce urmează să fie afișate, grupăm datele, introducem criteriile de filtrare și ordonare, etc) ;



- executăm interogarea

În fereastra de proiectare a unei interogări se pot identifica patru zone. În prima zonă sunt evidențiate tabelele, pe baza cărora se construiește interogarea, relațiile dintre ele, și sunt marcate câmpurile prelucrate de interogare. În cea de-a doua zonă se precizează modul în care câmpurile selectate intervin în interogare și sunt marcate cele ce urmează să fie afișate. Fereastra a treia conține codul SQL, cod în care se poate interveni direct iar în cea de-a patra fereastră se afișează rezultatul interogării.

⁵⁵ în limba engleză **Data Control Language**

⁵⁶ este instalat în folder-ul c:\Program Files\Microsoft SQL Server\90\Tools\Binn

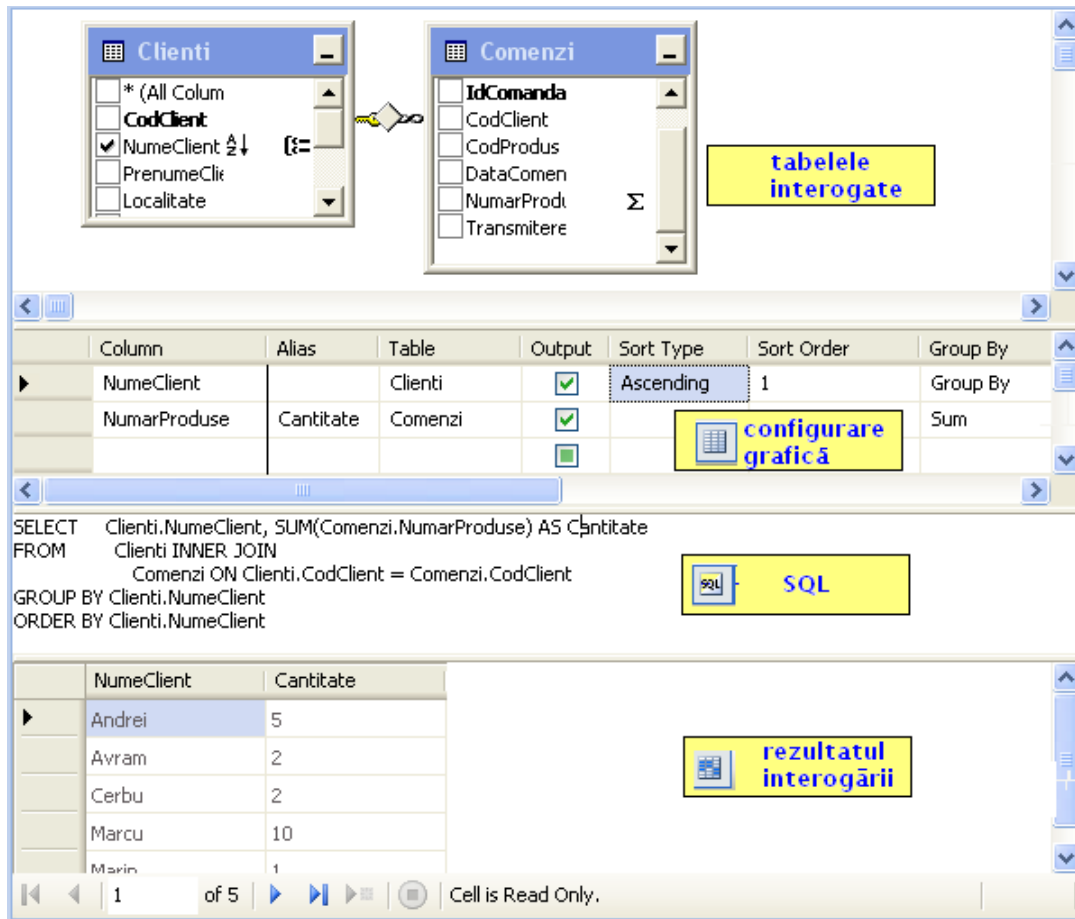


figura V-19 Fereastra de proiectare a unei interogări

V.5.2. Comenzi de manipulare a datelor

V.5.2.1 Comanda SELECT

```
SELECT <listă de coloane>
FROM <listă de tabele >
WHERE <condiție>
GROUP BY <condiție>
HAVING <condiție>
ORDER BY <condiție>
```

Comanda SELECT este utilizată pentru a extrage date din baza de date. Setul de date returnate prin intermediul unei comenzi SELECT are o structură asemănătoare cu a unei tabele, datele putând fi afișate sau pot fi utilizate pentru completarea unei alte baze de date.

Prin executarea unei comenzi SELECT se efectuează următoarele operații:

- **selecție** – permite filtrarea liniilor ce vor fi afișate (returnate) pe baza condițiilor din clauza WHERE;
- **proiecție** – permite filtrarea coloanelor afișate (returnate) pe baza listei de coloane menționate;

- **join** – permite prelucrarea datelor din două sau mai multe tabele pe baza unor criterii bine definite.

În clauza **FROM** se specifică obiectele (tabelele și vizualizările) din care se extrag date. Aceste obiecte pot fi însoțite de un **alias**. Dacă un obiect se află într-o bază de date situată la distanță atunci se va specifica numele legăturii către aceasta (**database link**).

Cea mai simplă formă a instrucțiunii SELECT este **SELECT * FROM tabela;** Caracterul ' * ' indică afișarea tuturor coloanelor tabelului. Dacă dorim să afișăm doar informații din câteva coloane ale tabelului vom preciza numele acestor coloane în clauza SELECT.

În exemplul următor se afișează, pentru fiecare client: numele, prenumele și localitatea.

```
SELECT NumeClient, PrenumeClient, Localitate
FROM Clienti
```

	NumeClient	PrenumeClient	Localitate
▶	Avram	Ionut	Bucuresti
	Marcu	Dana	Bacau
	Andrei	Anca	Bacau
	Marin	Marius	Bucuresti
	Cerbu	Vlad	Iasi

Dacă lista de coloane din clauza SELECT este precedată de cuvântul DISTINCT atunci se afișează doar liniile din tabelă ce corespund combinațiilor distincte de valori din aceste coloane.

În exemplul următor se utilizează specificatorul DISTINCT pentru a afișa categoriile de produse.

```
SELECT DISTINCT Categorie
FROM Produse
```

	Categorie
▶	jucarii
	papetarie

În clauza SELECT putem să scriem și expresii. De exemplu, pentru a afișa valoarea stocului corespunzător fiecărui produs, folosim comanda:

```
SELECT DenumireProdus, PretUnitar * Stoc AS Expr1
FROM produse
```

	DenumireProdus	Expr1
▶	creion	68
	pix	34500
	bicicleta	20358
	bicicleta	7320

Aliasul unei coloane permite afișarea unei alte expresii în locul numelui unei coloane. Această expresie poate fi încadrată de paranteze și este precedată de cuvântul **AS**. Exemplul anterior poate fi rescris:

```
SELECT DenumireProdus, PretUnitar * Stoc AS [Valoare stoc]
FROM produse
```

	DenumireProdus	Valoare stoc
	creion	68
	pix	34500

Dacă dorim să afișăm în aceeași coloană informații din mai multe coloane atunci putem să construim un șir de caractere obținut prin alipirea acestor informații cu ajutorul operatorului „+”. În exemplul următor se afișează în aceeași coloană numele și prenumele clienților.

```
SELECT NumeClient + ' ' + PrenumeClient AS Client, Localitate
FROM Cienti
```

	Client	Localitate
▶	Avram Ionut	Bucuresti
	Marcu Dana	Bacau
	Andrei Anca	Bacau

Selectarea liniilor returnate de comanda SELECT este realizată de clauza **WHERE**. În această clauză precizăm condițiile pe care trebuie să le îndeplinească o anumită linie pentru a fi returnată. Operatorii logici (NOT, AND, OR) și cei relaționali (<, <=, >, >=, =, <>) permit combinarea mai multor condiții în clauză. În exemplul următor se afișează produsele ce au prețul mai mic ca 12 și stocul mai mare ca 10.

```
SELECT DenumireProdus, PretUnitar, Stoc
FROM produse
WHERE (Stoc > 10) AND (PretUnitar < 12)
```

	DenumireProdus	PretUnitar	Stoc
▶	creion	2	34
	acuarele	5	300
	creion	2	23
	pix	4	400

Operatorul **LIKE** este utilizat în interogări pentru a verifica dacă un șir de caractere respectă un anumit „șablon”. Dacă valoarea se potrivește șablonului, atunci operatorul va returna valoarea adevărat, în caz contrar, va returna valoarea fals. În șablon se pot utiliza două caractere speciale:

- caracterul „_” ține locul unui singur caracter, oricare ar fi acesta;
- caracterul „%” ține locul oricărui subșir de caractere.


În exemplul următor se afișează toate produsele a căror denumire începe cu litera **b**.

```
SELECT DenumireProdus, Descriere
FROM produse
WHERE (DenumireProdus LIKE 'b%')
```

	DenumireProdus	Descriere
▶	bicicleta	5-7 ani
	bicicleta	3-5 ani

V.5.2.2 Gruparea datelor

Funcțiile de grup sunt funcții care returnează o singură valoare pentru fiecare grup de linii dintr-un tabel. Principalele funcții de grup sunt :

 **COUNT (x)** – returnează numărul de valori ale lui x;

Dacă x este înlocuit de caracterul "*" atunci funcția determină numărul de linii returnate iar dacă x este precedat de cuvântul **DISTINCT** atunci funcția returnează numărul de valori distincte și nenule ale expresiei x.

În exemplul următor se afișează numărul total de produse, numărul de produse cu denumiri diferite și numărul de categorii.

```
SELECT COUNT(*) AS [Numar produse], COUNT(DISTINCT DenumireProdus) AS [Produse diferite],
COUNT(DISTINCT Categorie) AS Categori
FROM produse
```

	Numar produse	Produse diferite	Categorii
▶	9	6	2

 **SUM (x)** – calculează suma tuturor valorilor din câmpul numeric x.

Dacă dorim să aflăm numărul total de produse este suficient să scriem comanda:

```
SELECT SUM(Stoc) AS [Numar total de produse]
FROM produse
```

	Numar total de produse
▶	4202

 **MIN (x)** – calculează cea mai mică valoare din câmpul numeric x.

Cel mai mic preț al unui produs se obține cu interogarea:


```
SELECT MIN(PretUnitar) AS [Pret minim]
FROM Produse
```

	Pret minim
▶	2

Dacă dorim să afișăm produsele care au cel mai mic preț atunci trebuie să includem în clauza WHERE o subinterogare⁵⁷:

```
SELECT DenumireProdus AS [Produsul cel mai ieftin]
FROM   Produse
WHERE  (PretUnitar =
        (SELECT MIN(PretUnitar) AS Expr1
         FROM   Produse AS Produse_1))
```

	Produsul cel mai ieftin
▶	creion
*	NULL

 **MAX (x)** – calculează cea mai mare valoare din câmpul numeric x.

Interogarea din figura următoare afișează stocul maxim (numărul maxim de produse de același tip)

```
SELECT DISTINCT MAX(Stoc) AS [stoc maxim]
FROM   Produse
```

	stoc maxim
▶	2300

Subinterogarea din exemplul următor este utilizată pentru a afișa denumirile produselor cu cel mai mare stoc.

```
SELECT DenumireProdus
FROM   Produse
WHERE  (Stoc =
        (SELECT MAX(Stoc) AS Expr1
         FROM   Produse AS Produse_1))
```

	DenumireProdus
▶	pix
*	NULL

 **AVG (x)** – calculează media valorilor din câmpul numeric x.

În exemplul următor se afișează prețul produsului cel mai ieftin, prețul celui mai scump produs și prețul mediu.

```
SELECT MIN(PretUnitar) AS [Cel mai ieftin], MAX(PretUnitar) AS [Cel mai scump], AVG(PretUnitar) AS [Pret mediu]
FROM   produse
```

	Cel mai ieftin	Cel mai scump	Pret mediu
▶	2	234	56,111111111111...

Clauza **GROUP BY** permite gruparea datelor dintr-o tabelă și obținerea informațiilor despre grupurile diferite. Utilizarea acestei clauze implică respectarea următoarelor condiții:

- toate câmpurile care apar în SELECT în afara funcțiilor de grup trebuie să apară în clauza GROUP BY;
- nu se pot folosi funcții de grup în clauza WHERE;
- în clauza GROUP BY pot să apară și coloane care nu apar în SELECT;

⁵⁷ O subinterogare este o interogare aflată în interiorul unei alte comenzi SQL. Subinterogările sunt rulate întotdeauna înaintea comenzii în care sunt incluse.

- funcțiile de grup pot fi imbricate.

Comanda următoare permite afișarea numărului de produse din fiecare categorie.

```
SELECT Categorie, COUNT(CodProdus) AS [Numar produse]
FROM produse
GROUP BY Categorie
```

	Categorie	Numar produse
▶	jucarii	3
	papetarie	6

Dacă dorim să selectăm numai o parte dintre grupurile obținute prin folosirea clauzei GROUP BY trebuie să utilizăm clauza **HAVING**.

În exemplul următor se utilizează clauza HAVING pentru a afișa localitățile pentru care avem cel puțin 2 clienți.

```
SELECT Localitate, COUNT(CodClient) AS [Numar clienti]
FROM Clienti
GROUP BY Localitate
HAVING (COUNT(CodClient) >= 2)
```

	Localitate	Numar clienti
▶	Bacau	2
	Bucuresti	2

V.5.2.3 Sortarea datelor

Clauza **ORDER BY** se utilizează atunci când se dorește afișarea datelor din tabelă ordonate după anumite criterii. În această clauză se precizează coloanele sau expresiile după care se vor ordona liniile unui tabel înainte de afișare.

Opțiunea **ASC** precizează că ordonarea se face crescător. Această opțiune poate fi omisă întrucât, implicit, datele sunt ordonate crescător. Opțiunea **DESC** precizează că sortarea se face descrescător.

Exemplul următor afișează produsele pornind de la cel mai scump și terminând cu cel mai ieftin.

```
SELECT DenumireProdus, Descriere, PretUnitar
FROM produse
ORDER BY PretUnitar DESC
```

	DenumireProdus	Descriere	PretUnitar
▶	bicicleta	5-7 ani	234
	bicicleta	3-5 ani	120
	papusa	Barbie	89
	stilou	InoxCrom	34
	pix	rosu metalic	15
	acuarele	12 culori	5
	pix	albastru	4
	creion	negru HB	2

În cazul ordonării descrescătoare valorile NULL se trec la sfârșit, în timp ce, în cazul ordonării crescătoare valorile NULL apar la început.

De exemplu, dacă se adaugă în tabela PRODUSE un nou produs, **creta**, fără descriere atunci acest produs va fi, într-o ordonare crescătoare după coloana **Descriere**, primul produs afișat.

```
SELECT DenumireProdus, Descriere
FROM Produse
ORDER BY Descriere
```

	DenumireProdus	Descriere
▶	creta	NULL
	acuarele	12 culori
	lovomotiva	3-5 ani
	bicicleta	3-5 ani
	bicicleta	5-7 ani
	pix	albastru metalic
	papusa	Barbie
	creion	HB
	stilou	InoxCrom

V.5.2.4 Interogări multiple

Comenzile SELECT prezentate până acum se încadrează în categoria cererilor simple (monorelație) întrucât referă date aflate într-o singură tabelă.

Cererile multirelație intervin atunci când este necesară accesarea datelor din mai multe tabele. Operația de regăsire a datelor din două sau mai multe tabele pe baza valorilor comune ale unor coloane se numește **JOIN**. De obicei aceste coloane reprezintă cheia primară, respectiv cheia străină a tabelelor.

Există mai multe moduri de legare a tabelelor și anume:



Produsul cartezian – leagă fiecare înregistrare dintr-o tabelă cu toate înregistrările din cealaltă tabelă. Pentru a obține produsul cartezian se folosește clauza **CROSS JOIN** în cadrul clauzei FROM.

O astfel de legătură se realizează atunci când afișăm pentru fiecare client fiecare produs disponibil.

```
SELECT  Clienti.NumeClient + ' ' + Clienti.PrenumeClient AS [Clientul ar putea cumpara],
        produse.DenumireProdus, produse.Descriere
FROM    Clienti CROSS JOIN produse
```

	Clientul ar putea cumpara	DenumireProdus	Descriere
▶	Avram Ionut	creion	negru X
	Avram Ionut	pix	rosu metalic
	Avram Ionut	bicicleta	5-7 ani
	Avram Ionut	bicicleta	3-5 ani
	Avram Ionut	acuarele	12 culori

Cell is Read Only.



Equijoin – leagă două tabele cu ajutorul unei condiții de egalitate. Pentru a specifica acest tip de legătură în cadrul clauzei FROM se folosește clauza **NATURAL JOIN** sau clauza **INNER JOIN..ON** urmată de o condiție de egalitate.

Acest tip de legătură se utilizează atunci când vrem să afișăm numărul de produse cumpărate de fiecare dintre clienți.

```
SELECT  Clienti.CodClient, Clienti.NumeClient, COUNT(*) AS [ Produse cumparate]
FROM    Clienti INNER JOIN Comenzi ON Clienti.CodClient = Comenzi.CodClient
GROUP BY Clienti.CodClient, Clienti.NumeClient
```

	CodClient	NumeClient	Produse cumparate
▶	1	Avram	2
	2	Marcu	2
	3	Andrei	3
	4	Marin	1
	5	Cerbu	1



Nonequijoin –leagă tabelele fără a utiliza o condiție de egalitate. În locul operatorului de egalitate se utilizează de cele mai multe ori operatorul **BETWEEN** pentru a verifica apartenența la un interval.

În exemplul următor se consideră tabela **repere** ce păstrează indicatori pentru volumul cumpărăturilor. De exemplu, dacă am achiziționat un produs sau două atunci putem spune că volumul achizițiilor este mic.

id1	id2	valoare
1	2	mic
3	5	mediu
6	10	mare
11	100	imens

Legătura dintre tabela **comezi** și tabela **repere** permite afișarea volumului de cumpărături pentru fiecare comandă.

```
SELECT Comenzi.IdComanda, Comenzi.NumarProduce, repere.valoare
FROM Comenzi INNER JOIN
      repere ON Comenzi.NumarProduce BETWEEN repere.id1 AND repere.id2
```

	IdComanda	NumarProduce	valoare
▶	1	1	mic
	2	1	mic
	5	2	mic
	6	2	mic
	7	1	mic
	8	1	mic
	9	2	mic
	3	5	mediu
	4	5	mediu



Self Join- este un equijoin dintre o tabelă și ea însăși.

Un exemplu clasic este cel al angajaților unei instituții în care unii dintre ei pot fi șefii altor angajați ai aceleiași instituții.



Outer Join – se utilizează atunci când dorim să afișăm datele dintr-o tabelă indiferent dacă au sau nu corespondent în tabela de legătură. Dacă datele fără corespondent se află în tabela din stânga clauzei JOIN atunci se utilizează opțiunea **LEFT OUTER JOIN** iar altfel opțiunea **RIGHT OUTER JOIN**.

O astfel de legătură se realizează atunci când vrem să afișăm toți clienții indiferent dacă au comandat sau nu produse.

```
SELECT Clienti.NumeClient, Comenzi.IdComanda
FROM Clienti LEFT OUTER JOIN
      Comenzi ON Clienti.CodClient = Comenzi.CodClient
```

	NumeClient	IdComanda
	Andrei	7
	Marin	8
	Cerbu	9
	Dinea	NULL

V.5.2.5 Comanda UPDATE

UPDATE [tabela]
SET [expresie de modificare]
WHERE [condiție]

Comanda UPDATE selectează înregistrările conform condiției din clauza WHERE și le modifică conform expresiei de modificare.

Această comandă este utilizată în exemplul următor pentru a modifica numărul de telefon al clientului ce are codul 4.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Clienți' table is expanded, showing columns: CodClient, NumeClient, PrenumeClient, Localitate, Strada, Numar, Bloc, Scara, and Telefon. The 'Telefon' column is selected. In the center, a message box from Visual Web Developer 2008 Express Edition states '1 row affected by last query'. Below this, a table shows the query details:

Column	Table	Set	New Value	Filter	Or...
Telefon	Clienti	<input checked="" type="checkbox"/>	7234283940		
CodClient	Clienti	<input type="checkbox"/>		= 4	

Below the table, the SQL query is displayed:

```
UPDATE Clienti
SET Telefon = 7234283940
WHERE (CodClient = 4)
```

At the bottom, a small window shows the SQL command: UPDATE: {Clienti.Telefon WITH VALUE (7234283940)}

V.5.2.6 Comanda INSERT

INSERT INTO [tabela] ([lista câmpuri])
VALUES ([lista de valori])

Comanda INSERT permite adăugarea de înregistrări într-o tabelă.

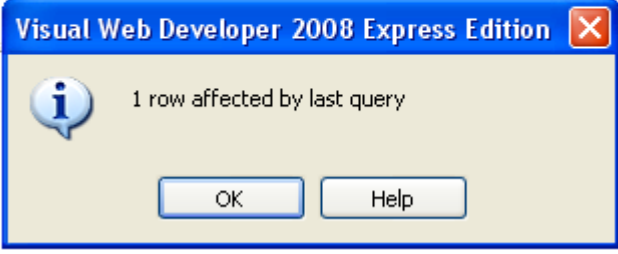
Dacă lista de coloane lipsește trebuie să introducem valori pentru toate câmpurile tabelii în ordinea în care au fost definite.

De exemplu, adăugarea unui nou reper care să indice un volum al cumpărăturilor mare atunci când numărul produselor comandate aparține intervalului [12,20] se poate face

fără precizarea câmpurilor dacă în lista valorilor datele se introduc în ordinea (id1,id2,valoare):

Column	New Value
id1	12
id2	20
valoare	'mare'

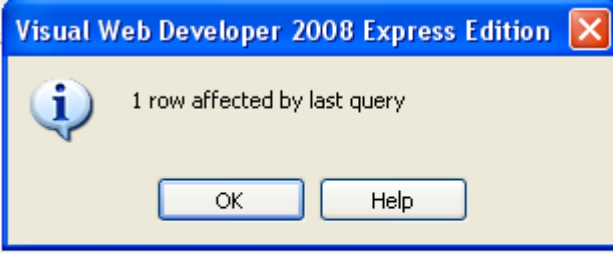
```
INSERT INTO repere
VALUES (12, 20, 'mare')
```



Dacă vrem să adăugăm un nou produs este suficient să cunoaștem câmpurile obligatorii și să le includem în lista câmpurilor ce urmează a fi completate apoi să completăm în aceeași ordine lista de valori. Câmpurile nemenționate în listă se inițializează cu NULL.

Column	New Value
CodProdus	10
DenumireProdus	'ursulet'
Categorie	'jucarii'
PretUnitar	18

```
INSERT INTO produse
(CodProdus, DenumireProdus, Categorie, PretUnitar, Stoc)
VALUES (10, 'ursulet', 'jucarii', 18, 100)
```



V.5.2.7 Comanda DELETE


**DELETE FROM [tabela]
WHERE [condiție de filtrare]**

Comanda DELETE determină ștergerea din tabelă a înregistrărilor ce verifică condiția din clauza WHERE.

Ștergerea produsului ce are codul 10 necesită utilizarea comenzii:

Column	Table	Filter	Or.
CodProdus	produse	= 10	

```
DELETE FROM produse
WHERE (CodProdus = 10)
```



V.5.3. Comenzi de definire a datelor

Așa cum am precizat la începutul acestui capitol, etapa de creare a modelului conceptual este urmată de crearea obiectelor care compun baza de date utilizată în aplicație.

V.5.3.1 Crearea tabelelor

Comanda **CREATE** poate fi utilizată pentru a crea o tabelă, o vizualizare, un index sau un sinonim.

Pentru crearea unei tabele se folosește comanda **CREATE TABLE**.

În cadrul comenzii CREATE TABLE putem utiliza clauza **DEFAULT** pentru a defini valori implicite pentru coloanele tabelului. Această clauză precizează ce valoare va avea un atribut atunci când, la inserare, nu se specifică în mod explicit valoarea acestuia. Valoarea implicită este dată de expresia ce urmează clauzei DEFAULT. Aceasta poate fi o constantă, o expresie SQL sau chiar o funcție SQL.

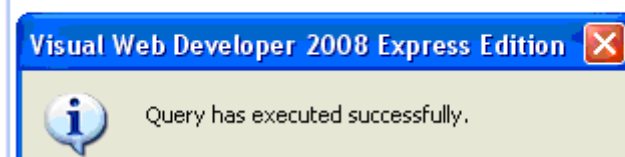
Utilizarea clauzei DEFAULT previne introducerea valorilor NULL în tabelă și permite utilizatorului să controleze unde și când au fost introduse valorile implicite.

O formă simplificată a comenzii CREATE TABLE este următoarea:

```
CREATE TABLE <nume tabelă>
(<coloană 1> <tip coloană 1 [DEFAULT <expresie1>]>
    [CONSTRAINT] [DISABLE] <constrângere>,
...
<coloană n> <tip coloană n [DEFAULT <expresie n>]>
    [CONSTRAINT] [DISABLE] <constrângere>);
```

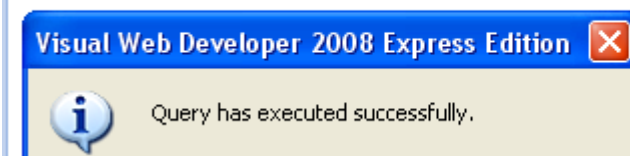
În exemplul următor se construiește tabela CARTE cu trei câmpuri (CodCarte, Titlu și AnAparitie). Nu au fost precizate valori implicite pentru coloanele tabelului.

```
CREATE TABLE CARTE(CodCarte int, Titlu varchar(50),AnAparitie int);
```



Dacă dorim ca anul apariției să fie implicit 2008 atunci putem folosi clauza DEFAULT pentru această coloană.

```
CREATE TABLE CARTE(CodCarte int, Titlu varchar(50),AnAparitie int DEFAULT 2008);
```



V.5.3.2 Modificarea structurii unei tabele

Modificarea structurii unei tabele se realizează cu ajutorul comenzii **ALTER TABLE**. Această comandă permite adăugarea sau ștergerea unei coloane, modificarea definiției unei coloane, adăugarea sau eliminarea unor constrângeri.

Pentru **adăugarea unei coloane** se utilizează clauza **ADD** a comenzii ALTER TABLE:

```
ALTER TABLE <nume tabelă>
ADD <coloană 1> <tip coloană 1 [DEFAULT <expresie1>]>
           [CONSTRAINT] [DISABLE] <constrângere>,
...
   <coloană n> <tip coloană n [DEFAULT <expresie n>]>
           [CONSTRAINT] [DISABLE] <constrângere>;
```

Coloana adăugată va fi ultima din tabelă. Dacă tabela conține deja date, atunci, pentru liniile existente vom avea, în coloana adăugată, valoarea NULL. Rezultă de aici că nu putem adăuga coloane cu restricția NOT NULL la o tabelă ce conține date.

În exemplul următor, se adaugă tablei CARTE două coloane.

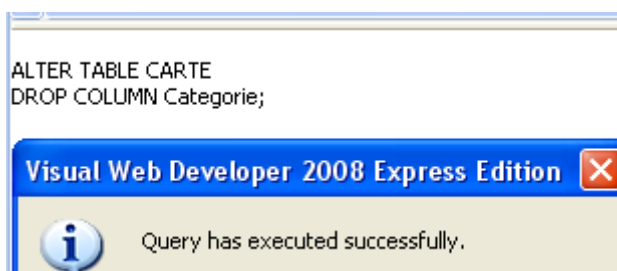


Ștergerea unei coloane se realizează cu ajutorul clauzei **DROP COLUMN** a comenzii ALTER TABLE:

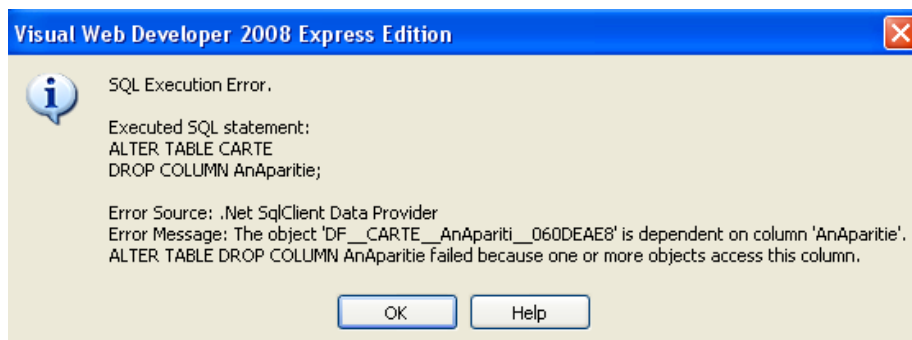
```
ALTER TABLE <nume tabelă>
DROP COLUMN <nume coloană>
```

Ștergerea unei coloane duce automat la ștergerea restricțiilor definite pentru aceasta și care nu implică alte obiecte.

În exemplul următor se șterge coloana **categorie** din tabela CARTE.



Dacă încercăm să ștergem coloana AnApariție atunci se obține o eroare (ca în imaginea următoare) pentru că există obiecte care accesează această coloană. În acest caz trebuie să utilizăm o ștergere în cascadă.



Modificarea unei coloane se realizează cu ajutorul clauzei **ALTER COLUMN** a comenzii ALTER TABLE.

Modificarea unei coloane include operații care presupun modificarea tipului unei coloane, a dimensiunii sau a valorii implicite (inclusiv adăugarea unei valori implicite).

```
ALTER TABLE <nume tabelă>  
ALTER COLUMN (<coloană > <tip > [constrângere]
```

În exemplul următor se modifică dimensiunea coloanei autor și se adaugă o constrângere NOT NULL.

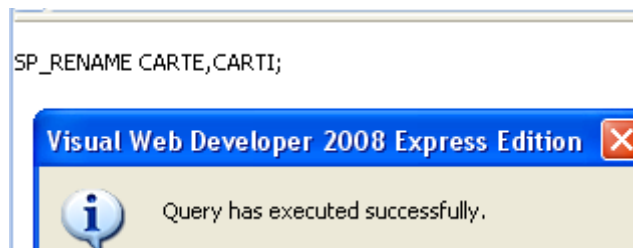


V.5.3.3 Redenumirea și ștergerea unei tabele

Redenumirea unei tabelei se realizează cu ajutorul comenzii SP_RENAME :

```
SP_RENAME <nume vechi>,<nume nou>
```

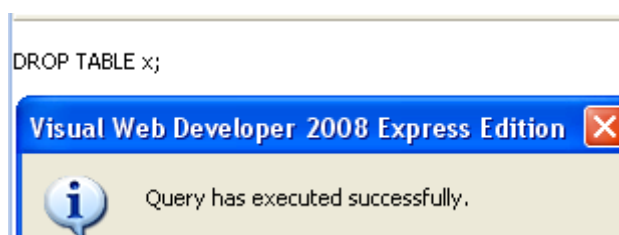
În exemplul următor tabela CARTE devine tabela CARTI.



Ștergerea unei tabele se face cu ajutorul comenzii

DROP TABLE <nume tabelă>

ca în exemplul din figura următoare:

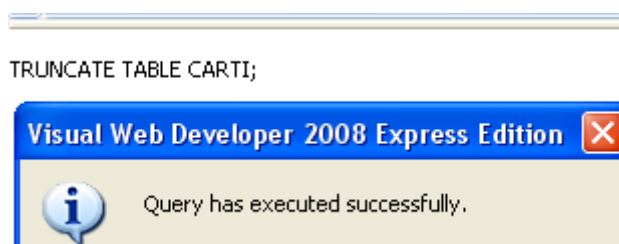


Dacă o tabelă este referită printr-o cheie străină atunci ea nu poate fi ștearsă până când nu este ștearsă tabela care o referă (se șterge mai întâi tabela copil și numai după aceea tabela părinte).

Pentru a șterge toate liniile tabelii și a elibera spațiul utilizat pentru stocarea datelor, fără a șterge definiția tabelii, se utilizează comanda

TRUNCATE TABLE <nume tabelă>

ca în exemplul din figura următoare:



V.5.3.4 Acordarea / revocarea unor privilegii

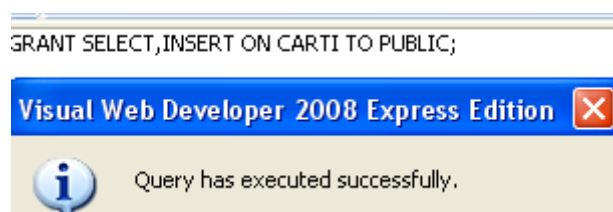
Pentru a accesa o bază de date un utilizator trebuie să conecteze cu un nume de utilizator. Orice utilizator are un domeniu de securitate care determină privilegiile și resursele pe care le poate utiliza.

Acordarea de privilegii asupra obiectelor se realizează cu ajutorul comenzii **GRANT**. Se pot acorda privilegii de accesare, inserare, actualizare la nivelul fiecărei coloane dintr-o tabelă.

```
GRANT <privilegiu> [(<coloană 1>, <coloană 2>, ... <coloană n>)]
ON <obiect>
TO {<user 1>[, <user 2>, .. <user n>] | <rol> | [PUBLIC]}
[WITH GRANT OPTION]
```

Opțiunea **WITH GRANT OPTION** permite unui utilizator să aloce altor utilizatori drepturile primite cu această opțiune.

În exemplul următor se acordă drepturi de selecție și inserare în tabela **Carti** tuturor utilizatorilor.



Revocarea privilegiilor implică utilizarea comenzii **REVOKE**:

```
REVOKE {<privilegiu 1> [, <privilegiu 2>, ... <privilegiu n> [ALL]}
ON <obiectt>
FROM | TO {<user 1>[, <user 2>, <user n>] | <rol> | [PUBLIC]}
[CASCADE]
```

Comanda

```
REVOKE INSERT ON CARTI TO User1;
```

revocă utilizatorului **User1** dreptul la inserare în tabela **CARTI**.

Opțiunea **CASCADE** indică faptul că privilegiul va fi revocat tuturor utilizatorilor cărora le-a fost acordat de către utilizatorul căruia i se revocă acest privilegiu, prin intermediul opțiunii **WITH GRANT OPTION**.



V.5.4. Evaluare

I.

- 1) Atunci când vrem să construim o aplicație Web legată la o bază de date:
 - a) realizăm mai întâi modelul conceptual și după aceea modelul fizic
 - b) construim modelul fizic și apoi modelul conceptual
 - c) modelul fizic se construiește în paralel cu modelul conceptual
- 2) Care dintre următoarele afirmații despre relațiile stabilite între tabele sunt adevărate:
 - a) Sunt implementate cu ajutorul cheilor străine
 - b) Se stabilesc între două tabele diferite
 - c) Se pot defini numai între tabele aflate în aceeași bază de date
- 3) Dacă se generează automat numele unei relații, atunci acesta are prefixul:
 - a) PK_
 - b) FK_
 - c) PFK_
 - d) FPK_
- 4) O tabelă poate avea o singură cheie primară
 - a) Adevărat
 - b) Fals
- 5) Care dintre următoarele afirmații referitoare la o cheie primară sunt adevărate:
 - a) Fiecare tabelă trebuie să aibă o cheie primară
 - b) Cheia primară realizează o indexare automată a înregistrărilor din tabelă
 - c) O tabelă poate avea mai multe chei primare
- 6) Care dintre următoarele reprezintă tipuri de relații
 - a) 1-n
 - b) 0-n
 - c) n-n
- 7) Se consideră două tabele T1 și T2 între care există o relație de tip 1-n (cheia primară din T1 este cheie străină în T2). Atunci:
 - a) Putem șterge oricând o înregistrare din T1
 - b) Putem șterge oricând o înregistrare din T2
 - c) O înregistrare din T1 poate fi ștearsă numai dacă în tabela T2 nu există înregistrări care să-i corespundă
- 8) Într-o tabelă
 - a) orice câmp poate avea valoare NULL
 - b) un câmp care reprezintă o cheie primară poate conține valori NULL
 - c) un câmp care reprezintă o cheie străină poate conține valori NULL
- 9) Ștergerea unei tabele părinte
 - a) Conduce la ștergerea tuturor tabelor copii
 - b) duce la invalidarea accesului la toate tabelele copii
 - c) Se poate face numai după ce au fost șterse toate tabelele copii
 - d) Nu se poate face în orice condiții

10) În Visual Studio, o relație între două tabele se poate defini automat, din

- Tabela părinte
- Tabela copil
- Oricare dintre cele două tabele

II. Se consideră modelul conceptual din **Figura.5-20**.

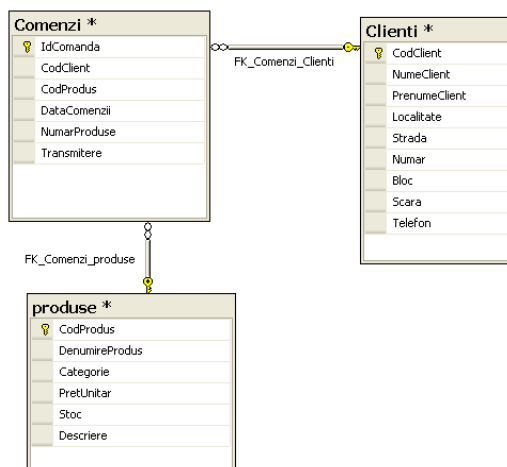


figura V-20

Scriveți instrucțiunile SQL pentru rezolvarea următoarelor cereri:

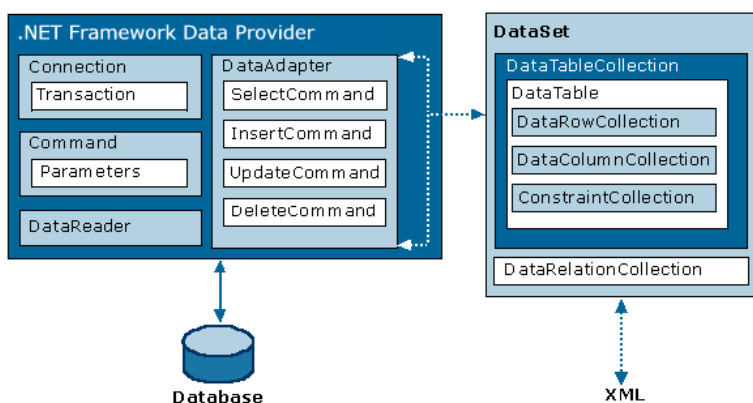
- Să se afișeze, fără duplicate, denumirile produselor
- Să se listeze denumirea și prețul produselor cu prețul mai mare decât prețul mediu al produselor din baza de date
- Să se afișeze, pentru fiecare client, numele și numărul produselor comandate
- Să se afișeze, pentru fiecare client, numele și produsele comandate
- Să se afișeze numele clientului care a comandat cele mai multe produse
- Să se afișeze denumirea produselor comandate de cel puțin 3 clienți
- Să se afișeze pentru clientul ce are codul 100 numărul comenzilor
- Să se afișeze pentru clientul ce are codul 100 valoarea totală a comenzilor sale
- Să se adauge clientul Popescu Ion, ce are codul 200 și adresa "Str.Sperantei, bl.1, sc.A, ap.1 Bucuresti"
- Pentru clientul ce are codul 200, să se adauge comanda : trei produse de cod 10, și un produs de cod 1.
- Să se modifice comanda anterioară astfel încât clientul să comande două produse de cod 1.
- Să se ștergă comenzile mai vechi de un an
- Să se ștergă toți clienții care nu au făcut nicio comandă
- Să se ștergă produsele care nu au fost comandate în ultimul an
- Să se afișeze toate comenzile făcute în luna octombrie a anului curent

V.6. MANIPULAREA BAZELOR DE DATE WEB PRIN INTERMEDIUL OBIECTELOR ADO.NET

În acest capitol vom vedea cum putem să combinăm comenzile SQL cu obiecte ADO.NET pentru a extrage și manipula bazele de date din aplicațiile Web.

V.6.1. Arhitectura ADO.NET

Componentele principale ale ADO.NET sunt **DataSet** și **Data Provider**. Ele au fost proiectate pentru accesarea și manipularea datelor.

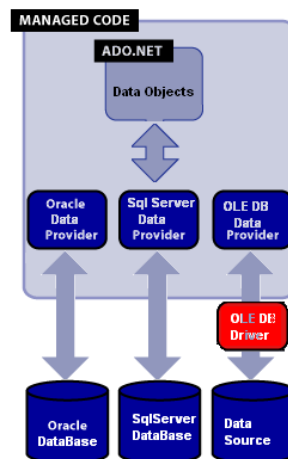


V.6.2. Furnizori de date (Data Providers)

Din cauza existenței mai multor tipuri de surse de date este necesar ca pentru fiecare tip de protocol de comunicare să se folosească o bibliotecă specializată de clase.

.NET Framework include **SQL Server.NET Data Provider** pentru interacțiune cu Microsoft SQL Server, **Oracle Data Provider** pentru bazele de date Oracle și **OLE DB Data Provider** pentru accesarea bazelor de date ce utilizează tehnologia OLE DB pentru expunerea datelor (de exemplu Access, Excel sau SQL Server versiune mai veche decât 7.0)

Furnizorul de date permite unei aplicații să se conecteze la sursa de date, execută comenzi și salvează rezultate. Fiecare furnizor de date cuprinde componentele **Connection**, **Command**, **DataReader** și **DataAdapter** :



	SQL Server Data Provider	OLE DB Data Provider	Oracle Data Provider
Connection	SqlConnection	OleDbConnection	OracleConnection
Command	SqlCommand	OleDbCommand	OracleCommand
DataReader	SqlDataReader	OleDbDataReader	OracleDataReader
DataAdapter	SqlDataAdapter	OleDbDataAdapter	OracleDataAdapter

V.6.3. Accesul direct la date prin intermediul ADO.NET

Așa cum am precizat în capitolul anterior accesul direct la date presupune construcția unei comenzi SQL, conectarea la baza de date, executarea comenzii și deconectarea de la baza de date fără memorarea directă a rezultatelor.

Dacă se lucrează în mod deconectat atunci se păstrează o copie a datelor într-un obiect de tip DataSet și acestea pot fi prelucrate și după deconectarea de la baza de date.

Modelul accesului direct la date corespunde paginilor ASP.NET unde nu este necesară memorarea datelor pentru perioade mai lungi de timp. O pagină ASP.NET este încărcată la cerere și accesul se încheie atunci când rezultatul cererii este furnizat user-ului, ceea ce înseamnă o pagină care are de obicei o viață de numai câteva secunde.

Rezultă așadar că o interogare directă a datelor presupune executarea pașilor următori:

- crearea obiectelor de tip Connection, Command și DataReader ;
- obținerea informațiilor din baza de date cu ajutorul obiectelor de tip DataReader și afișarea acestora într-un controler de pe un formular web;
- închiderea conexiunii;
- trimiterea paginii către utilizator; în acest moment nu mai avem o legătură directă între ceea ce vede utilizatorul și datele din baza de date, obiectele de tip ADO.NET fiind distruse.

Adăugarea, ștergerea sau modificarea datelor se realizează în doi pași:

- crearea obiectelor de tip Connection, Command;
- executarea comenzii directe.

În figura 5-21 sunt reprezentați pașii menționați anterior.

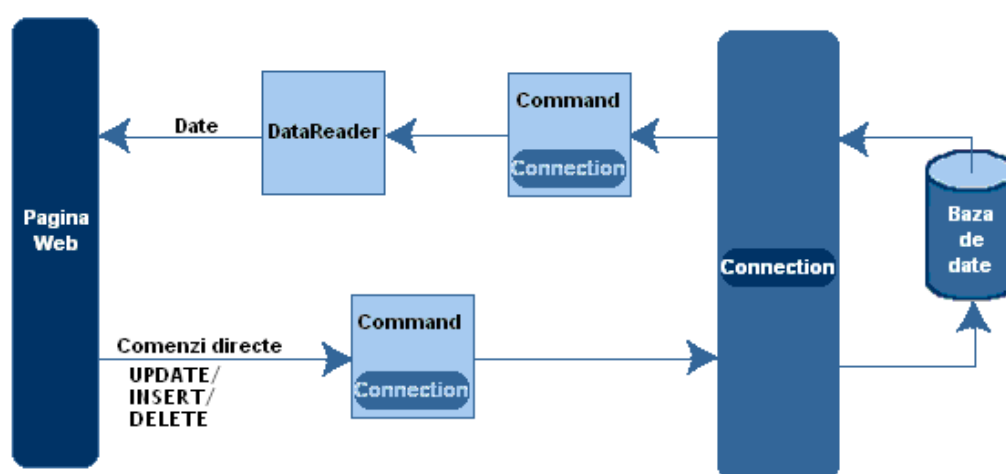


figura V-21

V.6.4. Crearea unei conexiuni

Înainte de orice operație cu o sursă de date externă, trebuie realizată o conexiune (legătură) cu acea sursă. Clasele din categoria *Connection* (*SqlConnection*, *OleDbConnection*, *OracleConnection* etc.) conțin date referitoare la sursa de date (locația, numele și parola contului de acces, etc.), metode pentru deschiderea/închiderea conexiunii, pornirea unei tranzacții etc. Aceste clase se găsesc în subspații (*SqlClient*, *OleDb*, *OracleClient* etc.) ale spațiului *System.Data*. În plus, ele implementează interfața *IdbConnection*.

Pentru deschiderea unei conexiuni prin program se poate instanția un obiect de tip conexiune, precizându-i ca parametru un șir de caractere conținând date despre conexiune.

Exemple de conectare

a) conectare la o sursă de date SQL

```
using System.Data.SqlClient;
SqlConnection con = new SqlConnection();
con.ConnectionString = "Data Source=localhost; User ID=profesor;pwd=info;
                    Initial Catalog=Orar";
con.Open();
```

b) conectare la o sursă de date SQL

```
using System.Data.SqlClient;
SqlConnection con = new SqlConnection(@"Data Source=serverBD;
                                     Database=scoala; User ID=elev;
                                     Password=secret");
con.Open();
```

c) conectare la o sursă de date Access

```
using System.Data.OleDb;
OleDbConnection con = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;
                                           Data Source=C:\Date\scoala.mdb");
con.Open();
```

În general, numărul maxim de conexiuni concurente este un număr fix și de aceea trebuie să menținem conexiunea deschisă cât mai puțin timp. Secvența de conectare se scrie de obicei într-un bloc try/catch care permite gestionarea erorilor ce pot apărea la conectare.

Atunci când se utilizează providerul **SQL Server** avem nevoie de două spații de nume:

```
using System.Data;
using System.Data.SqlClient;
```


Dacă se utilizează o variantă EXPRESS atunci șirul de conectare va include numele instanței:

```
SqlConnection conn;
conn = new SqlConnection("Data Source='.\SQLEXPRESS';
                        Initial Catalog=master;
                        Integrated Security=SSPI");
```

Proprietăți

- a) **ConnectionString** (String, cu accesorii de tip **get** și **set**) definește un șir care permite identificarea tipului și sursei de date la care se face conectarea și eventual contul și parola de acces. Conține lista de parametri necesari conectării sub forma *parametru=valoare*, separați prin ;.

Parametru	Descriere
Provider	Specifică furnizorul de date pentru conectarea la sursa de date. Acest furnizor trebuie precizat doar dacă se folosește OLE DB .NET Data Provider, și nu se specifică pentru conectare la SQL Server.
Data Source	Identifică serverul, care poate fi local, un domeniu sau o adresă IP ⁵⁸ .
Initial Catalog	Specifică numele bazei de date accesate. Baza de date trebuie să se găsească pe serverul dat în Data Source. ⁵⁹
Integrated Security ⁶⁰	Logarea se face cu user-ul configurat pentru Windows.
User ID	Numele unui user care are acces de logare pe server.
Password	Parola corespunzătoare ID-ului specificat.

- b) **ConnectionTimeout** (int, cu accesoriu de tip **get**): specifică numărul de secunde pentru care un obiect de conexiune poate să aștepte pentru realizarea conectării la server înainte de a se genera o excepție. (implicit 15). Se poate specifica o valoare diferită de 15 în ConnectionString folosind parametrul Connect Timeout, Valoarea Timeout=0 specifică așteptare nelimitată.

Exemplu de conectare:

```
using System.Data.SqlClient;
SqlConnection con = new SqlConnection("Data Source=serverBD;
                                    Database=scoala;
                                    User ID=elev;Password=secret;
                                    Connect Timeout=30");
```

⁵⁸ Dacă este instalat SQL Server Express Edition sursa de date este **localhost\SQLEXPRESS** sau **.\SQLEXPRESS**, pentru că numele instanței este SQLEXPRESS.

⁵⁹ Connection.ChangeDatabase() permite schimbarea numelui bazei de date.

⁶⁰ **User Id** și **Password** pot înlocui parametrul **Integrated Security**.

- c) **Database** (string, read-only): returnează numele bazei de date la care s-a făcut conectarea. Este necesară pentru a arăta unui utilizator care este baza de date pe care se face operarea
- d) **Provider** (de tip string, read-only): returnează furnizorul de date
- e) **ServerVersion** (string, read-only): returnează versiunea de server la care s-a făcut conectarea.
- f) **State** (enumerare de componente *ConnectionState*, read-only): returnează starea curentă a conexiunii. Valorile posibile: *Broken*, *Closed*, *Connecting*, *Executing*, *Fetching*, *Open*.

Metode

- a) **Open()**: deschide o conexiune la baza de date
- b) **Close()** și **Dispose()**: închid conexiunea și eliberează toate resursele alocate pentru ea
- c) **BeginTransaction()**: pentru executarea unei tranzacții pe baza de date; la sfârșit se apelează **Commit()** sau **Rollback()**.
- d) **ChangeDatabase()**: se modifică baza de date la care se vor face conexiunile. Noua bază de date trebuie să existe pe același server ca și precedenta.
- e) **CreateCommand()**: creează o comandă (un obiect de tip *Command*) validă asociată conexiunii curente.

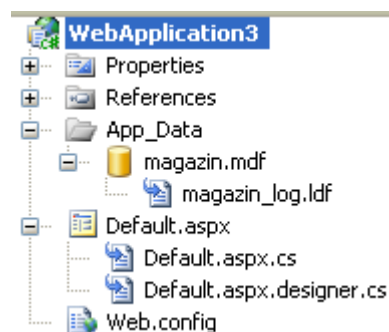
Evenimente

- a) **StateChange**: apare atunci când se schimbă starea conexiunii. Handlerul corespunzător (de tipul delegat *StateChangeEventHandler*) spune între ce stări s-a făcut tranziția.
- b) **InfoMessage**: apare când furnizorul trimite un avertisment sau un mesaj către client.

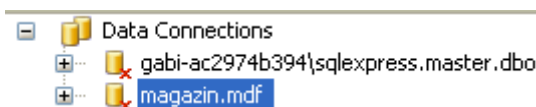
Instanțierea unei baze de date folosind mediul vizual

Visual Studio oferă două funcționalități care permit lucrul cu baze de date în folderul *App_Data*.

Așa cum am precizat la începutul acestui capitol, pentru a crea o bază de date alegem din *Web* → *Add New Item* → *SQL Server Database*. Noile fișiere *.mdf* și *.ldf* sunt plasate în folderul *App_Data* și pot fi vizualizate în *Solution Explorer*.



Când deschideți o aplicație web, **Visual Studio adaugă automat o conexiune** pentru fiecare bază de date din folderul App_Data. Pentru a alege o conexiune se apasă dublu click pe fișierul .mdf în Solution Explorer.



În mod obișnuit, toate bazele de date dintr-o aplicație vor utiliza același șir de caractere pentru precizarea modului de conectare.

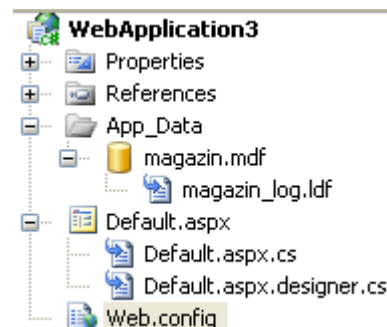
Din acest motiv se păstrează șirul de caractere ce precizează modul de conectare ca variabilă membru într-o clasă sau, chiar mai bine, într-un fișier de configurare.

Putem astfel să construim un obiect de tip Connection și să-i furnizăm, utilizând un constructor dedicat, șirul de caractere ce indică modul de conectare:

```
SqlConnection conn = new SqlConnection(StringConectare);
```

Menționăm aici că șirul de caractere care marchează o conexiune se găsește în secțiunea **<connectionStrings>** a fișierului Web.config.

```
<configuration>
...
<connectionStrings>
  <add name="sir"
    connectionString="DataSource=.\SQLEXPRESS;
                    Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
...
</configuration>
```

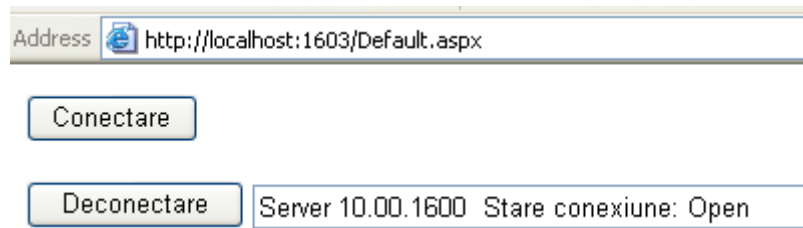


Pentru preluarea acestuia este necesară includerea spațiului de nume **System.Web.Configuration** iar pentru conectare se folosește codul:

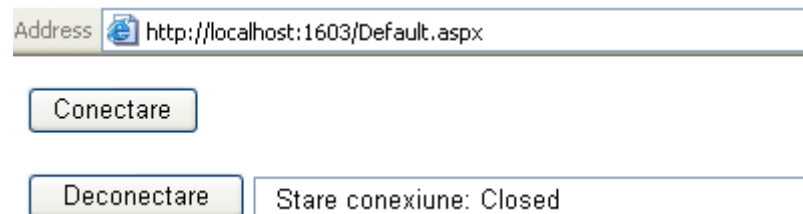
```
conn = WebConfigurationManager.ConnectionStrings["sir"].ConnectionString;
```

Înainte de efectuarea unor operații pe baza de date este bine să verificăm dacă am reușit să ne conectăm la sursa de date.

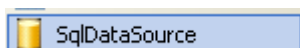
În exemplul următor apăsarea butonului **Open** determină conectarea la sursă, afișarea informațiilor legate de conectare și a stării acesteia (ca în imaginea umătoare)



iar apăsarea butonului **Close** determină închiderea conexiunii.

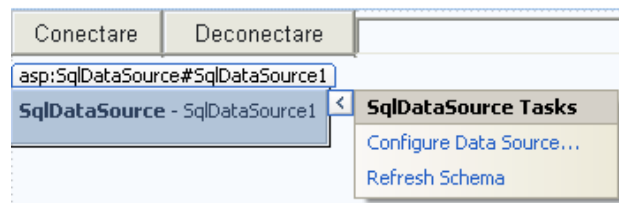


```
public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    public _Default()
    {
        conn = new SqlConnection("Data Source='.\SQLEXPRESS';
                                Initial Catalog=master;
                                Integrated Security=SSPI");
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            conn.Open();
            text1.Text = "Server " + conn.ServerVersion;
            text1.Text += " Stare conexiune: " + conn.State.ToString();
        }
        catch (Exception err)
        { text1.Text = "Error reading the database. " + err.Message; }
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        conn.Close();
        text1.Text = " Stare conexiune: " + conn.State.ToString();
    }
}
```

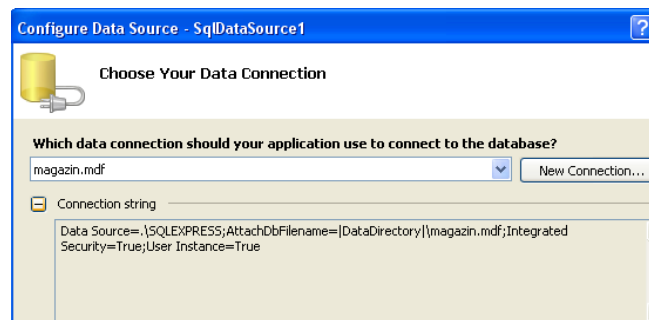


Sursa de date poate fi restricționată la o singură tabelă a bazei de date (exemplul 1 de mai jos) sau, mai mult decât atât la anumite linii ale tabelului (exemplul 2 de mai jos) sau chiar la anumite coloane ale tabelului (exemplul 3).

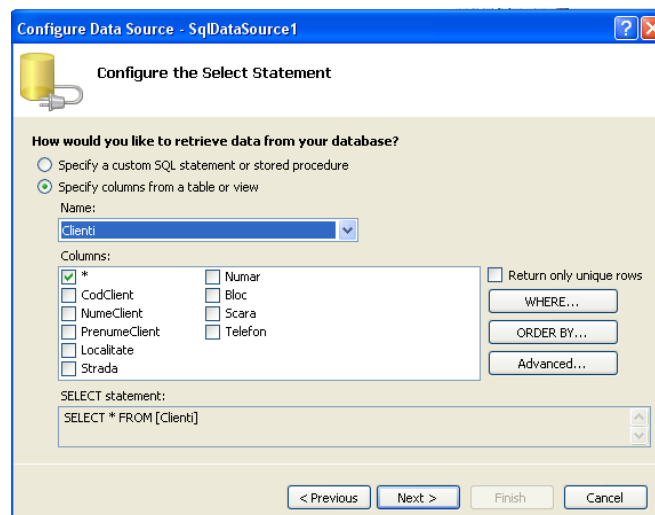
Exemplul 1. Sunt preluate toate coloanele tabeli clienți.



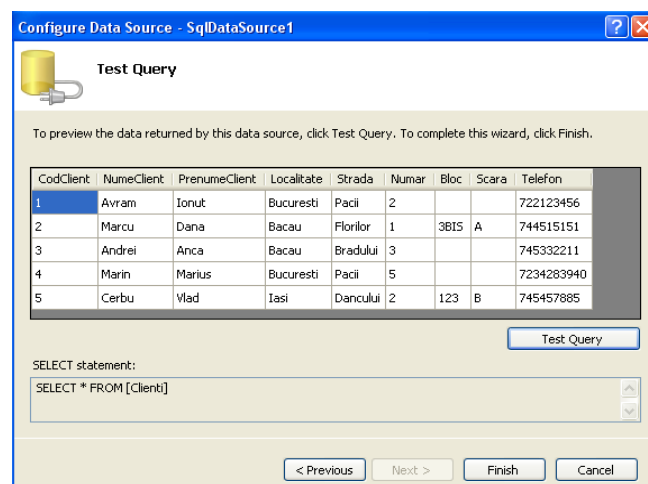
- precizăm baza de date;



- selectăm toate coloanele tabeli **Clienți** din care se preiau datele;



- verificăm dacă sursa conține toate informațiile din tabela **Clienți**.



Exemplul 2. Se preiau doar înregistrările clienților dintr-o localitate.

- adăugăm o clauză care să realizeze proiecția;

Add one or more conditions to the WHERE clause for the statement. For each condition you can specify either a literal value or a parameterized value. Parameterized values get their values at runtime based on their properties.

Column: Localitate

Operator: =

Source: None

SQL Expression: [Localitate] = @Localitate

Parameter properties
Value: Bacau

WHERE clause:

SQL Expression	Value
[Localitate] = @Localitate	Bacau

- clauza este afișată în fereastra ce conține expresia SQL de selecție;

WHERE clause:

SQL Expression	Value
[Localitate] = @Localitate	Bacau

- verificăm dacă sursa conține numai înregistrările ce corespund clauzei WHILE

Test Query

To preview the data returned by this data source, click Test Query. To complete this wizard, click Finish.

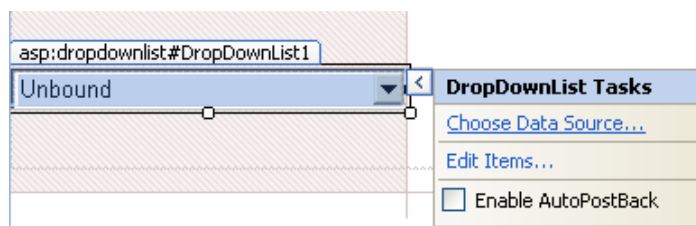
CodClient	NumeClient	PrenumeClient	Localitate	Strada	Numar	Bloc	Scara	Telefon
2	Marcu	Dana	Bacau	Florilor	1	3BIS	A	744515151
3	Andrei	Anca	Bacau	Bradului	3			745332211

Test Query

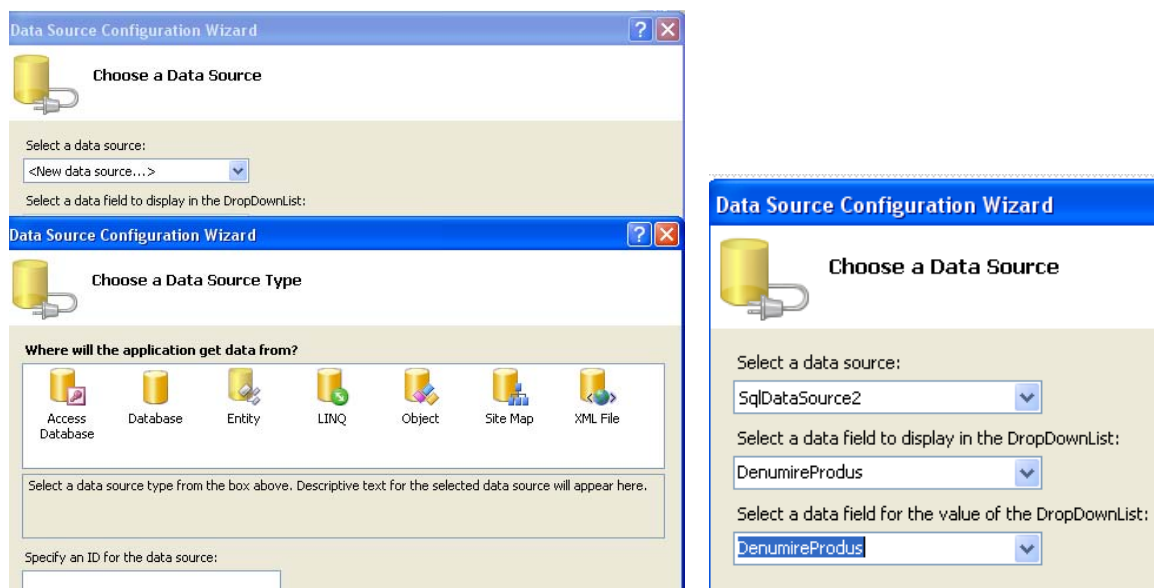
SELECT statement:
SELECT * FROM [Cienti] WHERE ([Localitate] = @Localitate)

< Previous Next > Finish Cancel

Exemplul 3. Am adăugat pe pagină o listă de tip DropDownList pentru care **sursa de date este reprezentată de numele, denumirea și prețul produselor.**

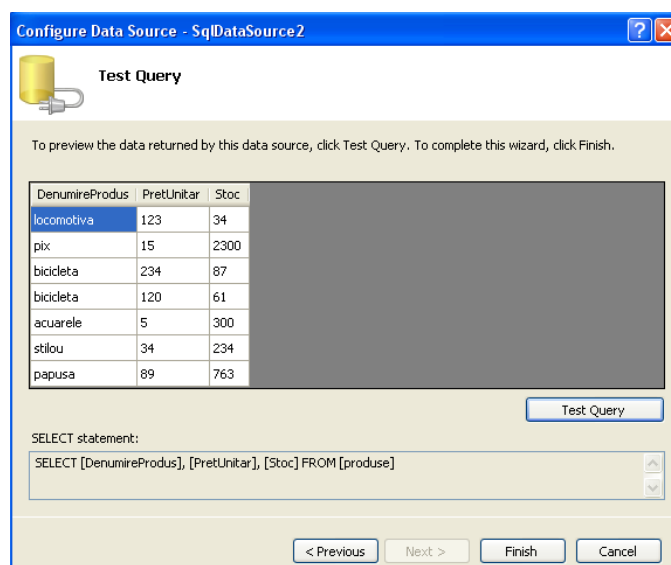


- precizăm sursa și numele câmpului ce va fi afișat în listă (DenumireProdus)



```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%"$ ConnectionStrings:ConnectionString %>"
    SelectCommand="SELECT [DenumireProdus], [PretUnitar], [Stoc] FROM
    [produse]"></asp:SqlDataSource>
```

- Verificăm dacă sursa conține cele trei coloane selectate.



V.6.5. Command

Clasele din categoria Command (**SqlCommand**, **OleDbCommand** etc.) conțin date referitoare la o comandă SQL (SELECT, INSERT, DELETE, UPDATE) și metode pentru executarea unei comenzi sau a unor proceduri stocate. Aceste clase implementează interfața **IDbCommand**. Ca urmare a interogării unei baze de date se obțin obiecte din categoriile **DataReader** sau **DataSet**. O comandă se poate executa numai după ce s-a stabilit o conexiune cu baza de date corespunzătoare.

Proprietăți

- a) **CommandText** (String): conține comanda SQL sau numele procedurii stocate care se execută pe sursa de date.
- b) **CommandTimeout** (int): reprezintă numărul de secunde care trebuie să fie așteptat pentru executarea comenzii. Dacă se depășește acest timp, atunci se generează o excepție.
- c) **CommandType** (enumerare de componente de tip *CommandType*): reprezintă tipul de comandă care se execută pe sursa de date. Valorile pot fi: *StoredProcedure* (apel de procedură stocată), *Text* (comandă SQL obișnuită), *TableDirect* (numai pentru OleDb)
- d) **Connection** (System.Data.[Provider].PrefixConnection): conține obiectul de tip conexiune folosit pentru legarea la sursa de date.
- e) **Parameters** (System.Data.[Provider].PrefixParameterCollection): returnează o colecție de parametri care s-au transmis comenzii.
- f) **Transaction** (System.Data.[Provider].PrefixTransaction): permite accesul la obiectul de tip tranzacție care se cere a fi executat pe sursa de date.

Metode

- a) Constructori:

SqlCommand()

SqlCommand(string CommandText)

SqlCommand(string CommandText, SqlConnection con)

SqlCommand(string CommandText,SqlConnection con,SqlTransaction trans)

- b) **Cancel()** oprește o comandă aflată în executare.
- c) **Dispose()** distruge obiectul comandă.

- d) **ExecuteNonQuery()** execută o comandă care nu returnează un set de date din baza de date; dacă comanda a fost de tip INSERT, UPDATE, DELETE, se returnează numărul de înregistrări afectate.

În exemplul următor sunt șterse din tabela elevi toate înregistrările în care regăsim numele BARBU și afișăm câte înregistrări am șters.

```
SqlCommand cmd = new SqlCommand();
cmd.CommandText = "DELETE FROM elevi WHERE nume = 'BARBU'";
cmd.Connection = con;
Console.WriteLine(cmd.ExecuteNonQuery().ToString());
```

- e) **ExecuteReader()** execută comanda și returnează un obiect de tip DataReader.

În exemplul următor se obține conținutul tabelii elevi într-un obiect de tip SqlDataReader.

```
SqlCommand cmd = new SqlCommand("SELECT * FROM elevi",con);
SqlDataReader reader = cmd.ExecuteReader();
while(reader.Read()) { Console.WriteLine("{0} - {1}",
    reader.GetString(0),reader.GetString(1));
}
reader.Close();
```

Metoda **ExecuteReader()** mai are un argument opțional de tip enumerare, **CommandBehavior**, care descrie rezultatele și efectul asupra bazei de date:

CloseConnection (conexiunea este închisă atunci când obiectul DataReader este închis),

KeyInfo (returnează informații despre coloane și cheia primară),

SchemaOnly (returnează doar informații despre coloane),

SequentialAccess (pentru manevrarea valorilor binare cu GetChars() sau GetBytes()),

SingleResult (se returnează un singur set de rezultate),

SingleRow (se returnează o singură linie).

- f) **ExecuteScalar()** execută comanda și returnează valoarea primei coloane de pe primul rând a setului de date rezultat; folosit pentru obținerea unor rezultate statistice.

Exemplu.

```
SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM elevi",con);
SqlDataReader reader = cmd.ExecuteScalar();
Console.WriteLine(reader.GetString(0));
```

- g) **ExecuteXmlReader()** returnează un obiect de tipul XmlReader obținut prin interogare.

```
SqlCommand CMD=
    new SqlCommand("SELECT * FROM elevi FOR XML MATE,EXAMEN", con);
System.Xml.XmlReader myXR = CMD.ExecuteXmlReader();
```

Obiectele de tip `SqlCommand` pot fi utilizate într-un scenariu ce presupune deconectarea de la sursa de date dar și în operații elementare care presupun obținerea unor rezultate imediate.

Vom exemplifica utilizarea obiectelor de tip `Command` în operații ce corespund acestui caz.

Presupunem că am stabilit conexiunea:

```
using System.Data.SqlClient;
SqlConnection conn = new SqlConnection(@"DataSource=serverBD;Database=MAGAZIN;
                                     User ID=adm;Password=eu");
conn.Open();
```

Instanțierea unui obiect de tip `SqlCommand`

```
SqlCommand cmd = new SqlCommand("select DenumireProdus from PRODUSE", conn);
```

conține un string ce precizează comanda care se execută și o referință către obiectul `SqlConnection`.

V.6.5.1 Selectarea datelor.

Pentru extragerea datelor cu ajutorul unui obiect `SqlCommand` trebuie să utilizăm metoda `ExecuteReader` care returnează un obiect `SqlDataReader`.

```
// Instanțiem o comandă cu o cerere și precizăm conexiunea
SqlCommand cmd = new SqlCommand("select DenumireProdus from PRODUSE", conn);
// Obținem rezultatul cererii
SqlDataReader rdr = cmd.ExecuteReader();
```

V.6.5.2 Inserarea datelor.

Pentru a insera date într-o bază de date utilizăm metoda `ExecuteNonQuery` a obiectului `SqlCommand`.

```
// șirul care păstrează comanda de inserare
string insertString = @"insert into PRODUSE(DenumireProdus, Descriere)
                       VALUES ('Barbie', 'papusa');
// Instanțiem o comandă cu această cerere și precizăm conexiunea
SqlCommand cmd = new SqlCommand(insertString, conn);
// Apelăm metoda ExecuteNonQuery pentru a executa comanda
cmd.ExecuteNonQuery();
```

V.6.5.3 Actualizarea datelor.

```
// șirul care păstrează comanda de actualizare
string updateString = @"update PRODUSE
                        set DenumireProdus = 'Locomotiva Thomas'
                        where DenumireProdus = 'Thomas';

// Instanțiem o nouă comandă fără să precizăm conexiunea
SqlCommand cmd = new SqlCommand(updateString);
// Stabilim conexiunea
cmd.Connection = conn;2
// Apelăm ExecuteNonQuery pentru executarea comenzii
cmd.ExecuteNonQuery();
```

V.6.5.4 Ștergerea datelor.

Se utilizează aceeași metodă ExecuteNonQuery.⁶¹

```
// șirul care păstrează comanda de ștergere
string deleteString = @"delete from PRODUSE where DENUMIRE_PRODUS = 'Barbie';
// Instanțiem o comandă
SqlCommand cmd = new SqlCommand();22
// Setăm proprietatea CommandText
cmd.CommandText = deleteString;
// Setăm proprietatea Connection
cmd.Connection = conn;
// Executăm comanda
cmd.ExecuteNonQuery();
```

Câteodată avem nevoie să obținem din baza de date o singură valoare, care poate fi o sumă, o medie sau alt rezultat al unei funcții agregat. O alegere ineficientă ar fi utilizarea metodei ExecuteReader și apoi calculul valorii. În acest caz, cea mai bună alegere este să lucrăm direct asupra bazei de date și să obținem această valoare.⁶²

⁶¹ În acest exemplu am ales să apelăm constructorul SqlCommand fără parametri pentru a exemplifica cum putem stabili explicit conexiunea și comanda.

⁶² În exemplul prezentat este necesară conversia întrucât rezultatul returnat de ExecuteScalar este de tip object

```
// Instanțiem o comandă nouă
SqlCommand cmd = new SqlCommand("select count(*) from PRODUSE", conn);
// Executăm comanda și obținem valoarea
int count = (int)cmd.ExecuteScalar();
```

V.6.6. DataReader

Datele pot fi explorate în mod conectat (cu ajutorul unor obiecte din categoria **DataReader**), sau pot fi preluate de la sursă (dintr-un obiect din categoria **DataAdapter**) și înglobate în aplicația curentă (sub forma unui obiect din categoria **DataSet**).

Clasele **DataReader** permit parcurgerea într-un singur sens a sursei de date, fără posibilitate de modificare a datelor la sursă. Dacă se dorește modificarea datelor la sursă, se va utiliza ansamblul **DataAdapter + DataSet**.

Datorită faptului că citește doar înainte (*forward-only*) acest tip de date este foarte rapid în citire. Overhead-ul asociat este foarte mic (overhead generat cu inspectarea rezultatului și a scrierii în baza de date). Dacă într-o aplicație este nevoie doar de informații care vor fi citite o singură dată, sau rezultatul unei interogări este prea mare ca să fie reținut în memorie (caching) **DataReader** este soluția cea mai bună.

Un obiect **DataReader** nu are constructor⁶³, ci se obține cu ajutorul unui obiect de tip **Command** și prin apelul metodei `ExecuteReader()` (vezi exercițiile de la capitolul anterior). Evident, pe toată durata lucrului cu un obiect de tip **DataReader**, conexiunea trebuie să fie activă. Toate clasele **DataReader** (**SqlDataReader**, **OleDbDataReader** etc.) implementează interfața **IDataReader**.

Proprietăți:

- IsClosed** (boolean, read-only)- returnează true dacă obiectul este deschis și fals altfel
- HasRows** (boolean, read-only)- verifică dacă reader-ul conține cel puțin o înregistrare
- Item** (indexator de câmpuri)
- FieldCount**-returnează numărul de câmpuri din înregistrarea curentă

Metode:

⁶³ Dacă pentru instanțiere este folosit operatorul **new** veți obține un obiect cu care nu puteți face nimic pentru că nu are o conexiune și o comandă atașate.

- a) **Close()** închidere obiectul și eliberarea resursele; trebuie să precedă închiderea conexiunii;
- b) **GetBoolean()**, **GetByte()**, **GetChar()**, **GetDateTime()**, **GetDecimal()**, **GetDouble()**, **GetFloat()**, **GetInt16()**, **GetInt32()**, **GetInt64()**, **GetValue()**, **GetString()** returnează valoarea unui câmp specificat, din înregistrarea curentă;
- c) **GetBytes()**, **GetChars()** citirea unor octeți/caractere dintr-un câmp de date binar;
- d) **GetDataTypeName()**, **GetName()** returnează tipul/numele câmpului specificat;
- e) **IsDBNull()** returnează true dacă în câmpul specificat prin index este o valoare NULL;
- f) **NextResult()** determină trecerea la următorul rezultat stocat în obiect (vezi exemplul);
- g) **Read()** determină trecerea la următoarea înregistrare, returnând *false* numai dacă aceasta nu există; de reținut că inițial poziția curentă este **înaintea** primei înregistrări.

DataReader obține datele într-un stream secvențial. Pentru a citi aceste informații trebuie apelată metoda **Read**; aceasta citește un singur rând din tabelul rezultat. Metoda clasică de a citi informația dintr-un **DataReader** este de a itera într-o buclă while.

DataReader implementează și indexatori (în exemplul anterior am afișat primele coloane folosind indexatori numerici).

Nu este foarte clar pentru cineva care citește codul care sunt coloanele afișate decât dacă s-a uitat și în baza de date. Din aceasta cauză este preferată utilizarea indexatorilor de tipul string. Valoarea indexului trebuie să fie numele coloanei din tabelul rezultat.

Indiferent că se folosește un index numeric sau unul de tipul string indexatorii întorc întotdeauna un obiect de tipul object fiind necesară conversia.

V.6.7. Comenzi parametrizate

Atunci când lucrați cu bazele de date aveți nevoie, de cele mai multe ori, să filtrați rezultatul după diverse criterii. De obicei acest lucru se face în funcție de condițiile specificate de către utilizator (ex: se afișează doar păpușile Barbie).

Cea mai simplă metodă de filtrare a rezultatelor este să construim dinamic string-ul **SqlCommand** dar această metodă nu este recomandată deoarece poate afecta baza de date (ex. Accesarea informațiilor confidențiale).

Dacă folosim interogări cu parametri atunci orice valoare pusă într-un parametru nu va fi tratată drept cod SQL, ci ca valoare a unui câmp, făcând aplicația mai sigură.

Pentru a folosi interogări cu parametri trebuie să :

- a) construiți string-ul pentru **SqlCommand** folosind parametri; ⁶⁴

```
SqlCommand cmd =  
new SqlCommand("SELECT * FROM PRODUSE WHERE DENUMIRE = @den", conn);
```

- b) construiți un obiect **SqlParameter** asignând valorile corespunzătoare;

Exemplu

```
SqlParameter param = new SqlParameter();  
param.ParameterName = "@Cden";  
param.Value = sir;
```

- c) adăugați obiectul **SqlParameter** la obiectul **SqlCommand**, folosind proprietatea **Parameters**.

```
cmd.Parameters.Add(param);
```

Astfel, comanda

```
SELECT * FROM produse WHERE CodProbus = 34;
```

devine

```
SELECT * FROM produse WHERE CodProbus = @CodProbus;
```

V.6.8. Studiu de caz

Pentru a exemplifica operațiile **DML** în **ASP.NET** am construit o aplicație care să permită vizualizarea informațiilor din tabela produse, modificarea atributelor unui produs, ștergerea unui produs din tabelă și adăugarea unui produs .

Am adăugat mai întâi pe pagină o listă de tip DropDown pentru care fiecare item conține o referire la o înregistrare din sursa de date menționată în exemplul 3 din secțiunea

⁶⁴ Atunci când comanda va fi executată `@den` va fi înlocuit cu valoarea aflată în obiectul `SqlParameter` atașat. Dacă nu asociem o instanță de tipul `SqlParameter` pentru un parametru din string-ul de interogare sau avem mai multe instanțe `SqlParameter` pentru un parametru vom obține o eroare la rulare

5.7.4 (sunt preluate din tabela **Produse** coloanele **CodProdus**, **DenumireProdus**, **PretUnitar** și **Stoc**)

Am adăugat apoi casete text în care vom afișa/introduce valorile atributelor unui produs și butoane a căror acționare determină executarea unei operații de inserare, modificare sau ștergere.

Design-ul aplicației este prezentat în figura 5-22.

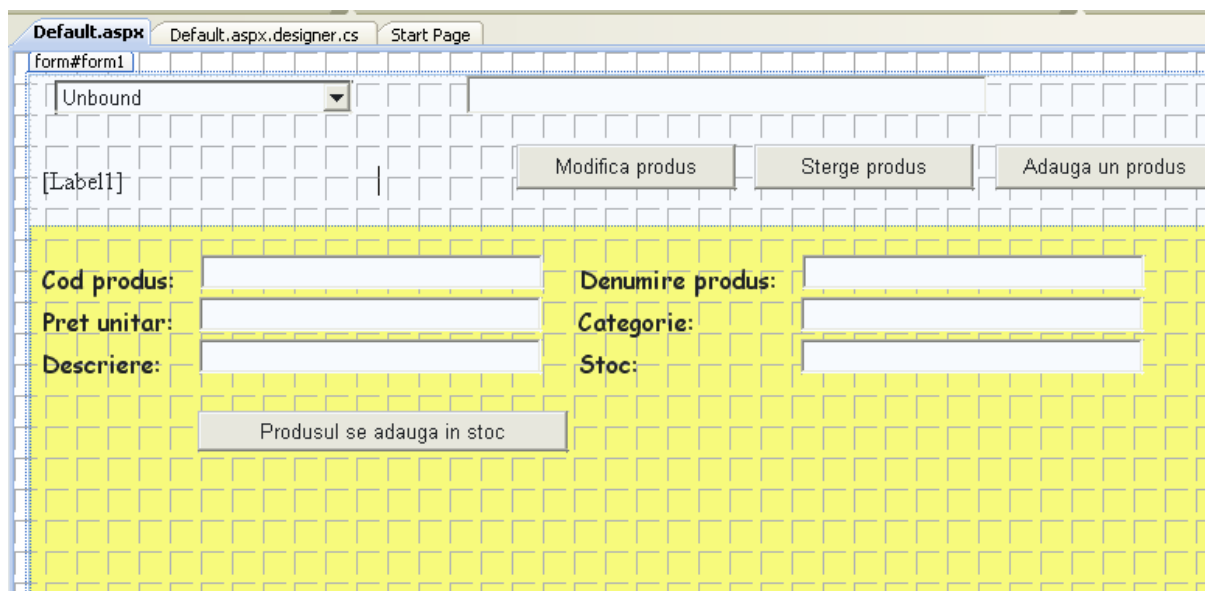


figura V-22

```
<asp:DropDownList ID="DropDownList1" runat="server"
    DataSourceID="SqlDataSource2"
    DataTextField="DenumireProdus"
    DataValueField="DenumireProdus" Height="45px"
    style="margin-top: 32px; margin-bottom: 46px" Width="219px"
    onselectedindexchanged="DropDownList1_SelectedIndexChanged">
</asp:DropDownList>

protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        ListaProduse();
    }
}
```

Fiecare item al listei este un șir de caractere format prin alipirea informațiilor din câmpurile **DenumireProdus**, **PretUnitar** și **Stoc**. Lista de produse va fi afișată la fiecare încărcare a paginii.

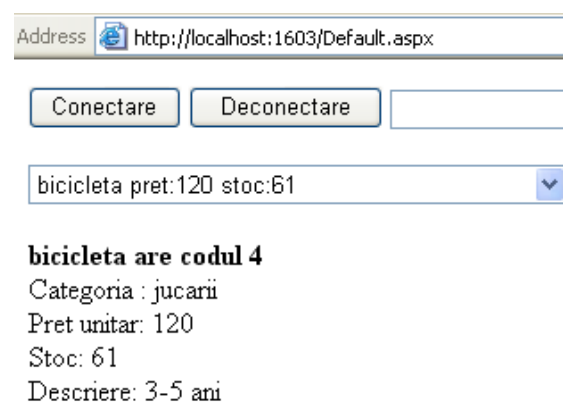
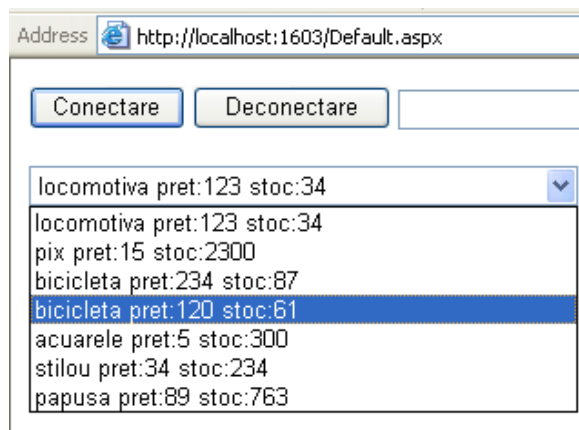
În cazul în care operația de conectare la baza de date sau operația de citire din baza de date eșuează se afișează un mesaj de eroare.

Subprogramul **ListaProduse()** conține secvența de conectare la sursa de date și secvența de completare a listei, conform descrierii anterioare, cu valori din tabela **Produse**.

```
private void ListaProduse()
{
    DropDownList2.Items.Clear();
    string selectSQL =
        "SELECT CodProdus,DenumireProdus,PretUnitar,Stoc FROM produse";
    conn = new SqlConnection
        ("Data Source='.\SQLEXPRESS';Initial Catalog=master; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(selectSQL, conn);
    SqlDataReader reader;

    try
    {
        conn.Open();
        reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            ListItem newItem = new ListItem();
            newItem.Text =
                reader["DenumireProdus"] + "pret:" +reader["PretUnitar"]+" stoc:" +reader["Stoc"];
            newItem.Value = reader["CodProdus"].ToString();
            DropDownList2.Items.Add(newItem);
        }
        reader.Close();
    }
    catch (Exception err)
    {
        text1.Text = "Eroare la citire :" + err.Message;
    }
    finally
    {
        conn.Close();
    }
}
```

La încărcarea paginii lista conține un număr de itemi egal cu numărul înregistrărilor din tabela **produse**, astfel încât utilizatorul poate selecta oricare dintre produsele aflate în această tabelă.




```

protected void DropDownList2_SelectedIndexChanged(Object sender,EventArgs e)
{
    string selectSQL=
    "SELECT*FROM produse WHERE CodProdus='"+DropDownList2.SelectedItem.Value +"'";

    conn = new SqlConnection
    ("Data Source='.\SQLEXPRESS';Initial Catalog=master; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(selectSQL, conn);
    SqlDataReader reader;
    try
    {
        conn.Open();
        reader = cmd.ExecuteReader();
        reader.Read();
        StringBuilder sb = new StringBuilder();
        sb.Append("<b>");sb.Append(reader["DenumireProdus"]);
        sb.Append(" are codul ");sb.Append(reader["CodProdus"]);
        sb.Append("</b><br />");
        sb.Append("Categorie:");sb.Append(reader["Categorie"]);sb.Append("<br />");
        sb.Append("Pret unitar:");sb.Append(reader["PretUnitar"]);sb.Append("<br />");
        sb.Append("Stoc: ");sb.Append(reader["Stoc"]);sb.Append("<br />");
        sb.Append("Descriere:");sb.Append(reader["Descriere"]);sb.Append("<br />");
        Label1.Text = sb.ToString();
        reader.Close();
    }
    catch (Exception err)
    {
        text1.Text = "Eroare: " + err.Message;
    }
    finally
    {
        conn.Close();
    }
}

```


Selectarea unui item din lista de produse determină afișarea unor informații suplimentare despre acest produs, într-o etichetă, cu ajutorul unui obiect de tip **StringBuilder**:

Selectarea unui item din listă determină executarea evenimentului

DropDownList2_SelectedIndexChanged AutoPostBack True 

eveniment ce determină preluarea informațiilor ce corespund itemului curent din tabela produse și afișarea acestor informații în formatul din imaginea alăturată.

Subprogramul anterior permite doar afișarea informațiilor referitoare la un produs, nu și modificarea acestora. De aceea, alegem o altă metodă de afișare a detaliilor referitoare la produsele din tabelă, și anume, fiecare valoare a unui atribut va fi afișată într-o casetă text.

Address  http://localhost:1603/Default.aspx

Alegeti produsul

Cod produs:	<input type="text" value="2"/>	Denumire produs:	<input type="text" value="pix"/>
Pret unitar:	<input type="text" value="15"/>	Categorie:	<input type="text" value="papetarie"/>
Descriere:	<input type="text" value="albastru metalic"/>	Stoc:	<input type="text" value="2300"/>

```
protected void DropDownList2_SelectedIndexChanged(Object sender, EventArgs e)
{
    string selectSQL=
    "SELECT*FROM produse WHERE CodProdus=" + DropDownList2.SelectedItem.Value + "";
    conn = new SqlConnection
    ("Data Source=.\SQLEXPRESS;Initial Catalog=master; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(selectSQL, conn);
    SqlDataReader reader;

    try
    {
        conn.Open();
        reader = cmd.ExecuteReader();
        reader.Read();
        icp.Text = reader["CodProdus"].ToString();
        ipu.Text = reader["PretUnitar"].ToString();
        il.Text = reader["Descriere"].ToString();
        idp.Text = reader["DenumireProdus"].ToString();
        ic.Text = reader["Categorie"].ToString();
        ist.Text = reader["Stoc"].ToString();
        reader.Close();
        text1.Text = "";
    }
    catch (Exception err)
    {
        text1.Text = "Eroare: " + err.Message;
    }
    finally
    {
        conn.Close();
    }
}
```

Adăugarea unei înregistrări se declanșează la apăsarea butonului „**Modifică produs**” care șterge textul afișat în casetele text și afișează butonul „Produsul se adaugă în

stoc”.

```
protected void ButonAdauga_Click(object sender, EventArgs e)
{
    icp.Text = ipu.Text = il.Text = "";
    idp.Text = ic.Text = ist.Text = text1.Text = "";
    ButonInserare.Visible = true;
}
```

Înainte de executarea operației de inserare se verifică dacă am completat câmpurile obligatorii. Dacă nu este îndeplinită această condiție atunci suntem avertizați și operația se încheie fără succes.

Address <http://localhost:1603/Default.aspx>

Alegeti produsul

Atributele Cod,Denumire,Stoc,Pret,Categorie sunt OBLIGATORII

Cod produs:	<input type="text"/>	Denumire produs:	<input type="text" value="minge"/>
Pret unitar:	<input type="text" value="3"/>	Categorie:	<input type="text" value="jucarii"/>
Descriere:	<input type="text"/>	Stoc:	<input type="text" value="3400"/>

Dacă au fost completate toate casetele text corespunzătoare câmpurilor obligatorii atunci inserarea se face cu succes și se afișează, într-o casetă text, care ne asigură de acest fapt.

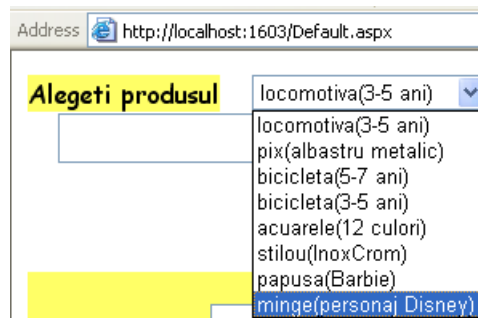
Address <http://localhost:1603/Default.aspx>

Alegeti produsul

Numar produse inserate:1

Cod produs:	<input type="text" value="10"/>	Denumire produs:	<input type="text" value="minge"/>
Pret unitar:	<input type="text" value="12"/>	Categorie:	<input type="text" value="jucarii"/>
Descriere:	<input type="text" value="personaj Disney"/>	Stoc:	<input type="text" value="340"/>

La o nouă accesare a listei de produse va fi afișat și produsul nou adăugat. În imaginea următoare avem detalii referitoare la produs adăugat la pasul anterior.



```

protected void ButonInserare_Click(object sender, EventArgs e)
{
    // verific daca au fost completate campurile obligatorii

    if (icp.Text == "" || ipu.Text == "" || idp.Text == "" || ic.Text == "" || ist.Text == "")
    {
        text1.Text = "Attributele Cod,Denumire,Stoc,Pret,Categorie sunt OBLIGATORII";
        text1.Visible = true;
        return;
    }

    string insertSQL =
    "INSERT INTO produse(CodProdus,DenumireProdus,PretUnitar,Categorie,Stoc,Descriere)
    VALUES ("";
    insertSQL += icp.Text + ", " + idp.Text + ", " + ipu.Text + ", " +
        ic.Text + ", " + ist.Text + ", " + il.Text + ")";

    conn = new SqlConnection
    ("Data Source=.\SQLEXPRESS;Initial Catalog=master; Integrated Security=SSPI");

    SqlCommand cmd = new SqlCommand(insertSQL, conn);
    int NrProduseAadaugate = 0;
    try
    {
        conn.Open();
        NrProduseAadaugate = cmd.ExecuteNonQuery();
        text1.Text = " Numar produse inserate:" + NrProduseAadaugate.ToString();
        text1.Visible = true;
    }
    catch (Exception err)
    {
        text1.Text = "Eroare la inserare " + err.Message;
    }
    finally
    {
        conn.Close();
    }
    // Actualizam lista de produse
    if (NrProduseAadaugate > 0)
    {
        ListaProduse();
    }
}

```

Secvența de inserare a unui nou produs poate fi rescrisă , utilizând **comenzile parametrizate**.

```


...
string insertSQL;
insertSQL = "INSERT INTO produse
(CodProdus,DenumireProdus,PretUnitar,Categorie,Stoc,Descriere) VALUES (";
insertSQL +=
"@CodProdus,@DenumireProdus,@PretUnitar,@Categorie,@Stoc,@Descriere)";

SqlCommand cmd = new SqlCommand(insertSQL, conn);

cmd.Parameters.AddWithValue("@CodProdus", icp.Text);
cmd.Parameters.AddWithValue("@DenumireProdus", idp.Text);
cmd.Parameters.AddWithValue("@PretUnitar", ipu.Text);
cmd.Parameters.AddWithValue("@Categorie", ic.Text);
cmd.Parameters.AddWithValue("@Stoc", ist.Text);
cmd.Parameters.AddWithValue("@Descriere", il.Text);
...

```

Inserarea se va face, și în acest caz, cu succes.


ddress  http://localhost:1603/Default.aspx

Alegeti produsul ▼

Numar produse inserate: 1

Cod produs:	<input type="text" value="15"/>	Denumire produs:	<input type="text" value="trenulet"/>
Pret unitar:	<input type="text" value="130"/>	Categorie:	<input type="text" value="jucarii"/>
Descriere:	<input type="text" value="electric, 12 vagoane"/>	Stoc:	<input type="text" value="200"/>

Modificarea unui produs presupune mai întâi selectarea acestuia din lista produselor, modificarea valorilor din casetele text și după aceea efectuarea operației propriuzise. Valoarea câmpului **CodProdus** nu se poate modifica întrucât regăsirea produsului ce urmează a fi modificat se face pe baza acestei chei.

ddress  http://localhost:1603/Default.aspx

Alegeti produsul ▼

Cod produs:	<input type="text" value="15"/>	Denumire produs:	<input type="text" value="trenulet"/>
Pret unitar:	<input type="text" value="130"/>	Categorie:	<input type="text" value="jucarii"/>
Descriere:	<input type="text" value="electric, 20 vagoane"/>	Stoc:	<input type="text" value="200"/>

```

protected void ButonModificare_Click(object sender, EventArgs e)
{
    string updateSQL;
    updateSQL = "UPDATE produse
                SET DenumireProdus=@DenumireProdus,PretUnitar=@PretUnitar,";
    updateSQL += "Categorie=@Categorie, Descriere=@Descriere ";
    updateSQL += "WHERE CodProdus=@CodProdus";

    conn = new SqlConnection
("Data Source='.\SQLEXPRESS';Initial Catalog=master; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(updateSQL, conn);

    cmd.Parameters.AddWithValue("@DenumireProdus", idp.Text);
    cmd.Parameters.AddWithValue("@PretUnitar", ipu.Text);
    cmd.Parameters.AddWithValue("@Categorie", ic.Text);
    cmd.Parameters.AddWithValue("@Stoc", ist.Text);
    cmd.Parameters.AddWithValue("@Descriere", il.Text);

    cmd.Parameters.AddWithValue("@CodProdus",DropDownList2.SelectedItem.Value);
    int NrProduseModificate = 0;
    try
    {
        conn.Open();
        NrProduseModificate = cmd.ExecuteNonQuery();
        text1.Text = "Produse modificate:" +NrProduseModificate.ToString();
    }
    catch (Exception err){text1.Text = "Eroare la modif" +err.Message;}
    finally {conn.Close();}

    if (NrProduseModificate > 0) ListaProduse();
}

```

După modificare, lista de produse este actualizată, și sunt afișate noile valori pentru produsul modificat.

Address <http://localhost:1603/Default.aspx>

Alegeti produsul

Produse modificate:1

Cod produs:	<input type="text" value="15"/>	Denumire produs:	<input type="text" value="trenulet"/>
Pret unitar:	<input type="text" value="130"/>	Categorie:	<input type="text" value="jucarii"/>
Descriere:	<input type="text" value="electric, 20 vagoane"/>	Stoc:	<input type="text" value="200"/>

Ștergerea unui produs din tabelă se face numai dacă nu sunt încălcate condițiile de integritate date de relațiile dintre tabele. De exemplu, nu putem șterge un produs care a fost deja comandat .

În exemplul din imaginea următoare se încercă ștergerea produsului **pix**. Întrucât acesta apare în cel puțin o comandă, operația de ștergere se încheie fără succes.

The screenshot shows a web application interface. At the top, the address bar displays 'http://localhost:1603/Default.aspx'. Below it, there is a section titled 'Alegeti produsul' with a dropdown menu showing 'pix(rosu metalic)'. To the right of the dropdown, a red error message reads 'The DELETE statement conflicted with the REFERENCE'. Below this, there are three buttons: 'Modifica produs', 'Sterge produs', and 'Adauga un produs'. At the bottom, there is a table with the following data:

Cod produs:	2	Denumire produs:	pix
Pret unitar:	15	Categorie:	papetarie
Descriere:	rosu metalic	Stoc:	2300

```
protected void ButonStergere_Click(object sender, EventArgs e)
{
    string deleteSQL;
    deleteSQL = "DELETE FROM produse WHERE CodProdus=@CodProdus";
    string StringConectare;
    StringConectare = "Data Source='.\SQLEXPRESS';Initial Catalog=master; Integrated Security=SSPI";
    conn = new SqlConnection(StringConectare);
    SqlCommand cmd = new SqlCommand(deleteSQL, conn);
    cmd.Parameters.AddWithValue("@CodProdus ", DropDownList2.SelectedItem.Value);
    int NrProduseSterse = 0;
    try
    {
        conn.Open();
        NrProduseSterse = cmd.ExecuteNonQuery();
        text1.Text = "Produse sterse:" + NrProduseSterse.ToString();
    }
    catch (Exception err) {text1.Text = err.Message;}
    finally {conn.Close();}
    if (NrProduseSterse > 0) ListaProduse();
}
}
```

Dacă produsul ales nu este comandat atunci el poate fi șters. În exemplul din imaginea următoare produsul **creion** poate fi șters întrucât niciun client nu a comandat acest produs.

Address <http://localhost:1603/Default.aspx>

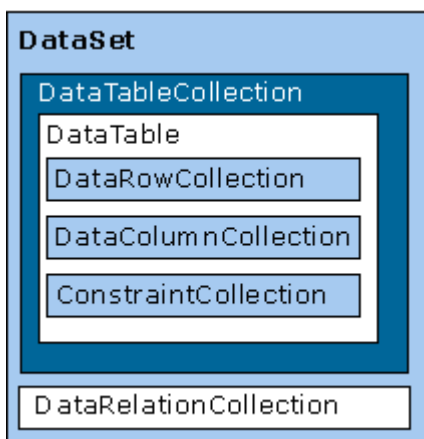
Alegeti produsul Produse sterse:1

Cod produs:	10	Denumire produs:	minge
Pret unitar:	12	Categorie:	jucarii
Descriere:	personaj Disney	Stoc:	340

V.7. LUCRUL ÎN MOD DECONECTAT

Folosirea combinată a obiectelor **DataAdapter** și **DataSet** permite operații de selectare, ștergere, modificare și adăugare la baza de date.

V.7.1. DataAdapter



Clasele **DataAdapter** generează obiecte care funcționează ca o interfață între sursa de date și obiectele **DataSet** interne aplicației, permițând prelucrări pe baza de date. Ele gestionează automat conexiunea cu baza de date astfel încât conexiunea să se facă numai atunci când este necesar.

Un obiect **DataSet** este de fapt un set de tabele relaționate. Folosește serviciile unui obiect **DataAdapter** pentru a-și procura datele și trimite modificările înapoi către baza de date. Datele sunt stocate de un **DataSet** în format

XML, folosit și pentru transferul datelor.

În exemplul următor se preiau datele din tabelele elevi și profesori:

```

//transferă datele în datasetul ds sub forma unei tabele locale numite elevi
SqlDataAdapter de=new SqlDataAdapter("SELECT Nume,Pren,clasa FROM Elevi", conn);
de.Fill(ds,"elevi");

//transferă datele în datasetul ds sub forma unei tabele locale numite profi
SqlDataAdapter dp=new SqlDataAdapter("SELECT Nume,Pren FROM Profesori", conn);
dp.Fill(ds,"profesori");
  
```

Proprietăți

- a) **DeleteCommand**, **InsertCommand**, **SelectCommand**, **UpdateCommand** (Command), conțin comenzile ce se execută pentru selectarea sau modificarea datelor în sursa de date.
- b) **MissingSchemaAction** (enumerare) determină ce se face atunci când datele aduse nu se potrivesc peste schema tablei în care sunt depuse. Poate avea următoarele valori:
 - a. **Add** - implicit, **DataAdapter** adaugă coloana la schema tablei
 - b. **AddWithKey** – se adugă coloana și informații relativ la cheia primară
 - c. **Ignore** - se ignoră lipsa coloanei respective, ceea ce duce la pierdere de date
 - d. **Error** - se generează o excepție de tipul *InvalidOperationException*.

Metode

- a) Constructori: **SqlDataAdapter()** | **SqlDataAdapter(obiect_comanda)**
SqlDataAdapter(string_comanda, conexiune);
- b) **Fill()** permite umplerea unei tabele dintr-un obiect **DataSet** cu date. Permite specificarea obiectului **DataSet** în care se depun datele, eventual a numelui tablei din acest **DataSet**, numărul de înregistrare cu care să se înceapă popularea (prima având indicele 0) și numărul de înregistrări care urmează a fi aduse.
- c) **Update()** permite transmiterea modificărilor efectuate într-un **DataSet** către baza de date.

V.7.2. DataSet

Un **DataSet** este format din **Tables** (colecție formată din obiecte de tip **DataTable**; **DataTable** este compus la rândul lui dintr-o colecție de **DataRow** și **DataColumn**), **Relations** (colecție de obiecte de tip **DataRelation** pentru memorarea legăturilor părinte-copil) și **ExtendedProperties** ce conține proprietăți definite de utilizator.

Scenariul uzual de lucru cu datele dintr-o tabelă conține următoarele etape:

- popularea succesivă a unui **DataSet** prin intermediul unuia sau mai multor obiecte **DataAdapter**, apelând metoda **Fill** (vezi exemplul de mai sus)
- procesarea datelor din **DataSet** folosind numele tabelelor stabilite la umplere, `ds.Tables["elevi"]`, sau indexarea acestora, `ds.Tables[0]`, `ds.Tables[1]`
- actualizarea datelor prin obiecte comandă corespunzătoare operațiilor INSERT, UPDATE și DELETE. Un obiect **CommandBuilder** poate construi automat o combinație de comenzi ce reflectă modificările efectuate.

Așadar, **DataAdapter** deschide o conexiune doar atunci când este nevoie și o închide imediat când aceasta nu mai este necesară.

De exemplu **DataAdapter** realizează următoarele operațiuni atunci când trebuie să populeze un **DataSet**: *deschide conexiunea, populează DataSet-ul, închide conexiunea* și următoarele operațiuni atunci când trebuie să facă update pe bază de date: *deschide conexiunea, scrie modificările din DataSet în baza de date, închide conexiunea*. Între operațiunea de populare a **DataSet**-ului și cea de update conexiunile sunt închise. Între aceste operații în **DataSet** se poate scrie sau citi.

Crearea unui obiect de tipul **DataSet** se face folosind operatorul **new**.

Exemplu.

```
DataSet dsProduce = new DataSet();
```

Constructorul unui **DataSet** nu necesită parametri. Există totuși o supraîncărcare a acestuia care primește ca parametru un string și este folosit atunci când trebuie să se facă o serializare a datelor într-un fisier XML. În exemplul anterior avem un **DataSet** gol și avem nevoie de un **DataAdapter** pentru a-l popula.

Un obiect **DataAdapter** conține mai multe obiecte **Command** (pentru inserare, modificare, ștergere și selecție) și un obiect **Connection** pentru a citi și scrie date.

În exemplul următor construim un obiect de tipul **DataAdapter**, **daProd**. Comanda SQL specifică cu ce date va fi populat un **DataSet**, iar conexiunea **conn** trebuie să fi fost creată anterior, dar nu și deschisă. **DataAdapter**-ul va deschide conexiunea la apelul metodelor **Fill** și **Update**.

```
SqlDataAdapter daProd =  
new SqlDataAdapter("SELECT IdProduce, DenumireProduce FROM PRODUCE", conn);
```

Prin intermediul constructorului putem instanția doar comanda de interogare. Instanțierea celorlalte comenzi se face fie prin intermediul proprietăților pe care le expune **DataAdapter**, fie folosind obiecte de tipul **CommandBuilder**.

```
SqlCommandBuilder cmdBldr = new SqlCommandBuilder(daProd);
```

La inițializarea unui **CommandBuilder** se apelează un constructor care primește ca parametru un adapter, pentru care vor fi construite comenzile. **SqlCommandBuilder** care nu poate construi decât comenzi simple și care se aplică unui singur tabel. Atunci când trebuie

să utilizăm comenzi care vor folosi mai multe tabele este recomandată construirea separată a comenzilor și apoi atașarea lor adapterului, folosind proprietăți.

Popularea DataSet-ului se face după ce am construit cele două instanțe:

```
daProd.Fill(dsProduce, "PRODUSE");
```

În exemplul următor va fi populat DataSet-ul dsProduce. Cel de-al doilea parametru (string) reprezintă numele tabelului (nu numele tabelului din baza de date, ci al tabelului rezultat în DataSet) care va fi creat. Scopul acestui nume este identificarea ulterioară a tabelului. În cazul în care nu sunt specificate numele tabelelor, acestea vor fi adăugate în DataSet sub numele *Table1*, *Table2*, ...

Un **DataSet** poate fi folosit ca **sursă de date** pentru un **DataGrid** din ASP.Net .

Exemplu.⁶⁵

```
DataGrid dgProduce = new DataGrid();  
dgProduce.DataSource = dsProduce;  
dgProduce.DataMembers = "PRODUSE";
```

După ce au fost făcute modificări într-un **DataSet** acestea trebuie scrise și în baza de date. Actualizarea se face prin apelul metodei **Update**.

```
daProd.Update(dsProduce, "PRODUSE");
```

Completarea listei de produse pentru lucrul în mod deconectat se rescrie :

⁶⁵ Se pot afișa mai multe tabele dintr-un *DataSet*, semnul "+" permițându-i utilizatorului să aleagă care tabel să fie afișat. Pentru a suprima afișarea aceluși semn "+" setăm proprietatea *DataMembers* pe numele tabelului care va fi afișat. Numele tabelului este același care l-am folosit ca parametru în apelul metodei *Fill*.

```

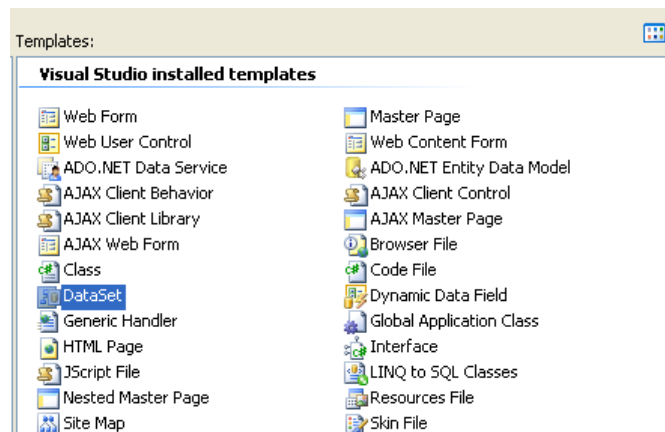
private void ListaProduce()
{
    DropDownList2.Items.Clear();
    string selectSQL =
    "SELECT CodProduce,DenumireProduce,PretUnitar,Stoc,Descriere FROM produse";
    conn = new SqlConnection(StringConectare);
    SqlCommand cmd = new SqlCommand(selectSQL, conn);
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    try{conn.Open();
    // Toate informatiile sunt transferate folosind o singura comanda
    // care creaza o tabela noua numita "produse" în DataSet.

    adapter.Fill(ds, "produse");
    }
    catch (Exception err){text1.Text = "Eroare la citire " + err.Message;}
    finally {conn.Close();}
    foreach (DataRow produs in ds.Tables["produse"].Rows)
    {
        ListItem newItem = new ListItem();
        newItem.Text = produs["DenumireProduce"] + "(" + produs["Descriere"] + ")";
        newItem.Value = produs["CodProduce"].ToString();
        DropDownList2.Items.Add(newItem);
    }
}
}

```

V.7.3. Proiectare DataSet în mediu vizual

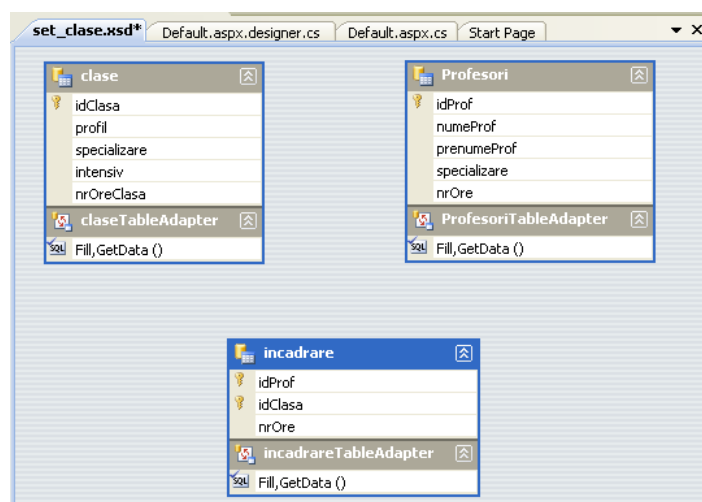
Pentru a construi un obiect de tip DataSet în mediu vizual trebuie să folosim opțiunea **Add → New Item** și să selectăm template-ul **DataSet**.



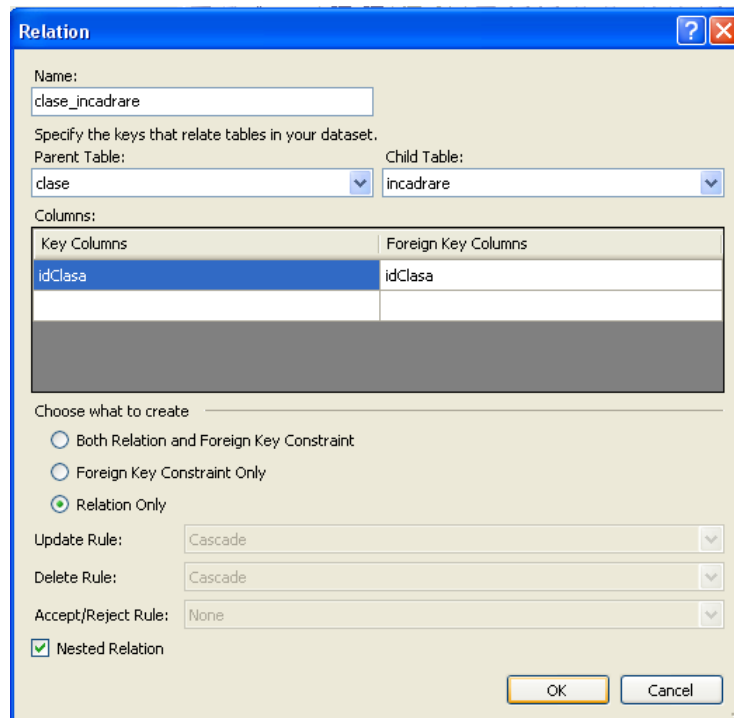
Putem să adăugăm itemi din Database Explorer (tabele, vizualizări, etc), prin selecție simplă și deplasare pe suprafața de design sau să adăugăm elemente noi pe o suprafața corespunzătoare unui obiect DataSet.



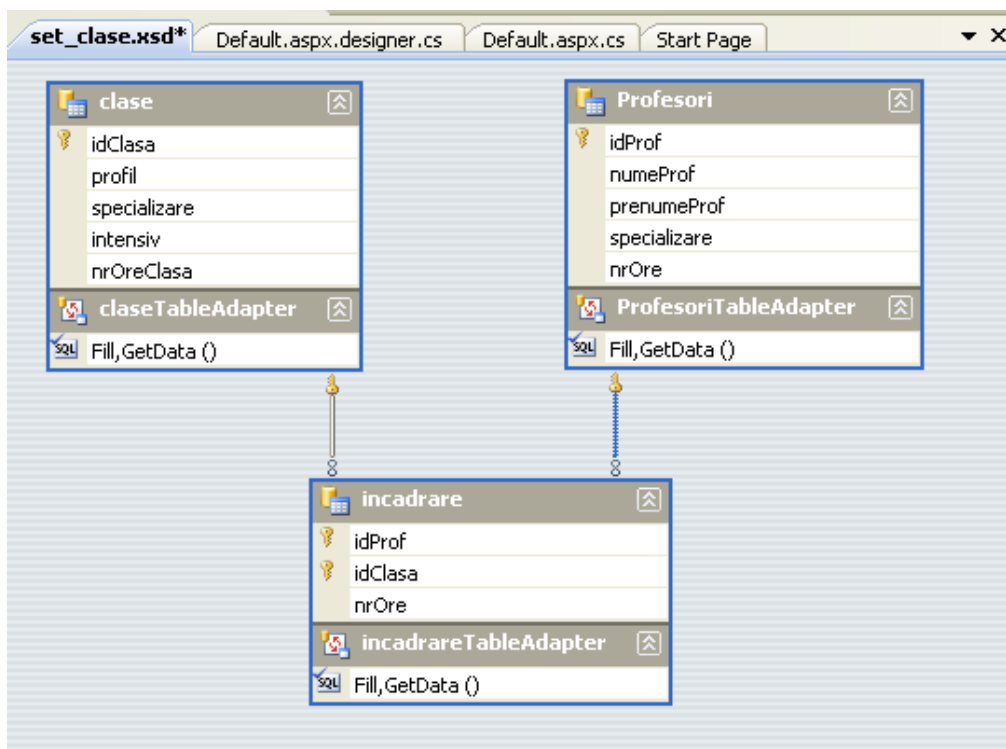
În acest exemplu, obiectul DataSet va conține înregistrări ale tabelelor **clase**, **profesori** și **încadrare** din baza de date **școală**.



Pentru a trasa relațiile dintre tabele putem folosi opțiunea **Relation** din meniul Toolbox atașat. După alegerea acestei unelte trebuie să stabilim tabelele relaționate și tipul relațiilor.



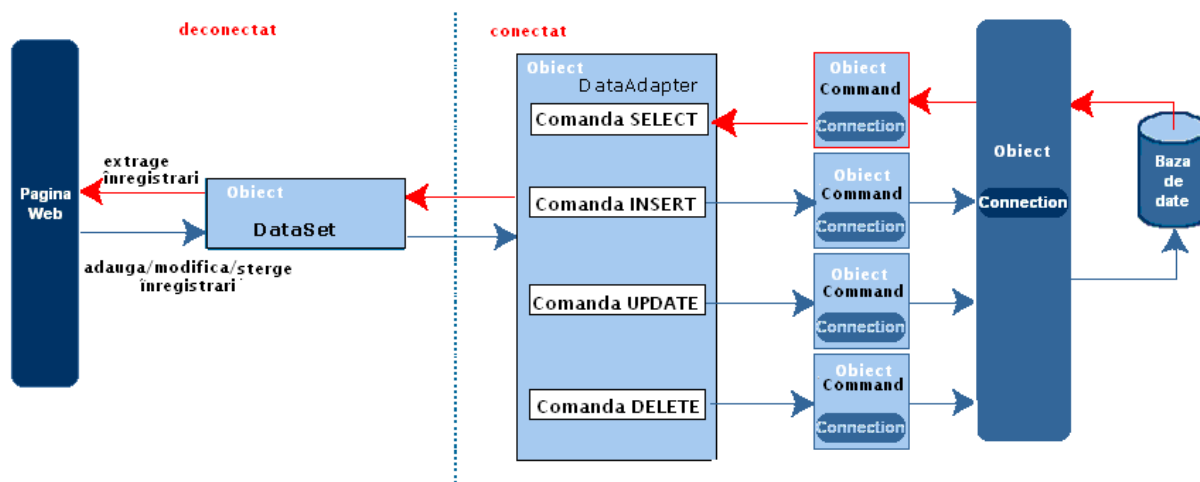
În cazul exemplului ales au fost marcate cele două relații de tip 1-n dintre tabele.



În concluzie, dacă doriți să extrageți înregistrări dintr-o bază de date și vreți să le puneți într-un obiect de tip DataSet trebuie să utilizați un obiect de tip **DataAdapter**. Fiecare obiect de tip DataAdapter conține patru tipuri de comenzi (SELECT,INSERT,UPDATE,DELETE). Acest lucru vă permite să utilizați un singur obiect DataAdapter pentru mai multe sarcini.

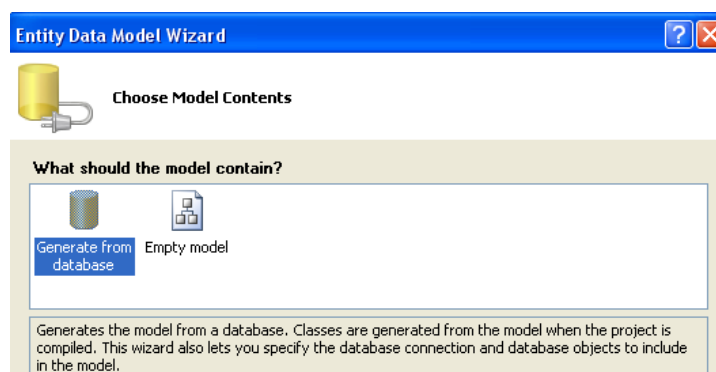
Obiectul de tip comandă transmis constructorului este automat asignat proprietății `DataAdapter.SelectCommand`.

Imaginea următoare este o reprezentare a modului în care aplicația Web interacționează cu baza de date.

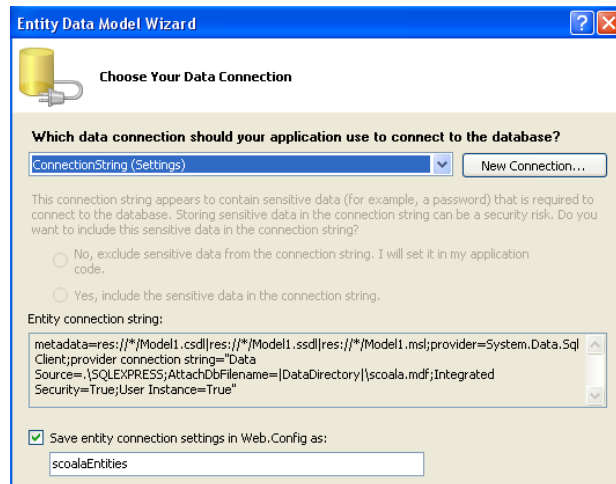


Modelul conceptual atașat unei aplicații poate fi generat automat prin selectarea din Solution Explorer a opțiunii `Add` → `New Item` → `ADO.NET Entity Data Model`

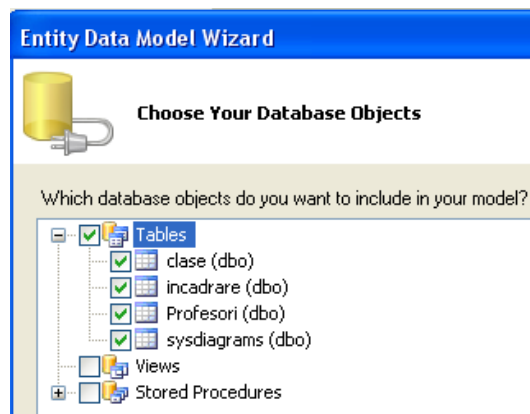
Pentru generare putem folosi tabelele unei baze de date deja construită sau putem crea un model conceptual nou.



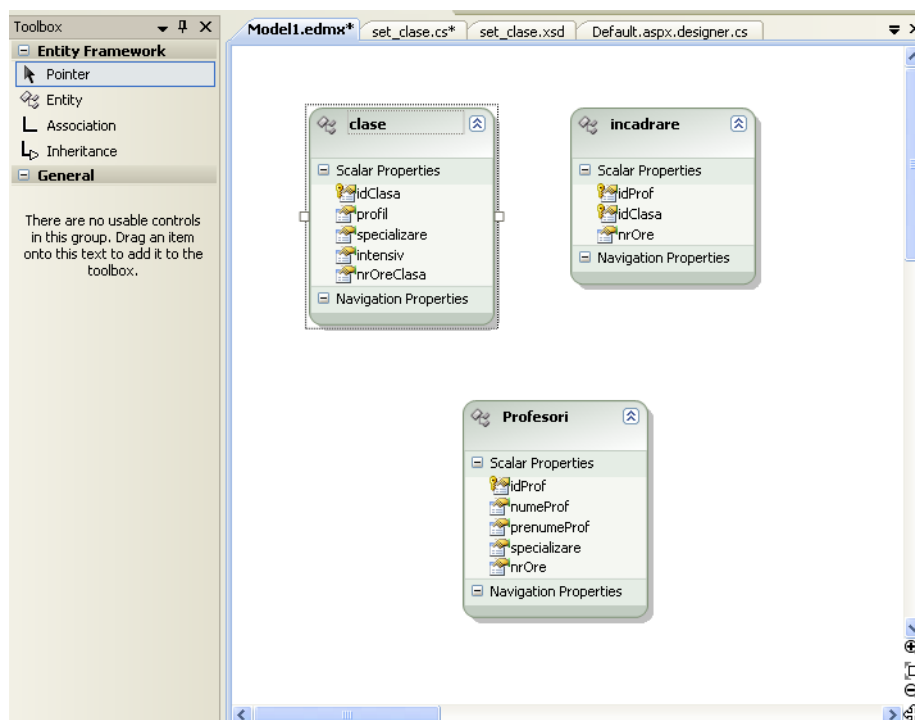
Pentru a alege itemii ce urmează a fi reprezentați în modelul conceptual trebuie să ne conectăm la baza de date aferentă și să selectăm elementele (tabele, vizualizări etc.)



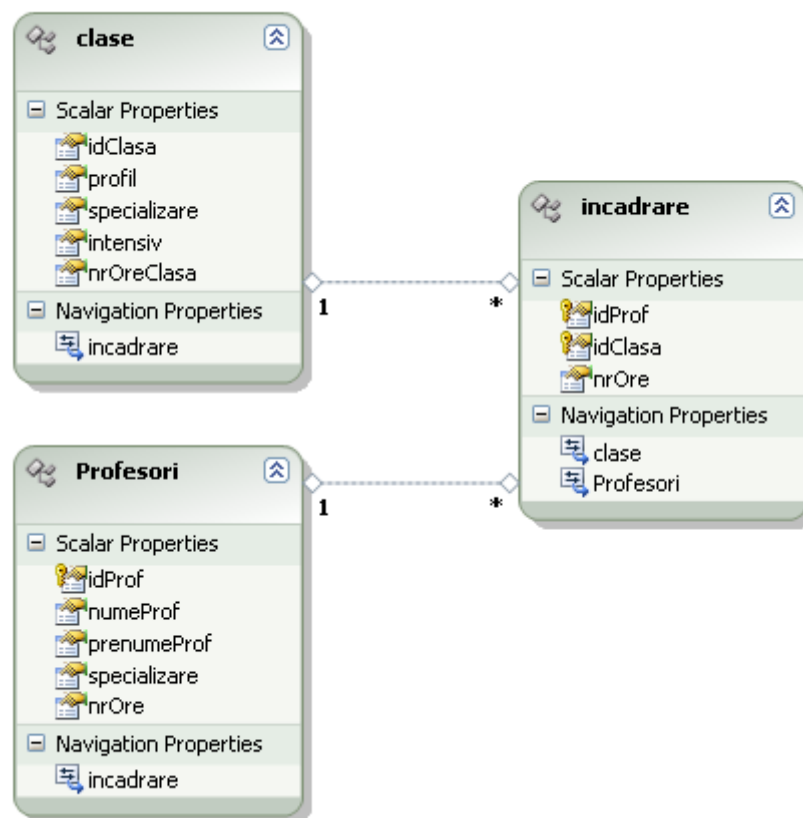
În exemplul următor se construiește un model conceptual pe baza tabelelor și a relațiilor existente între tablele clase, încadrare, profesori.



După alegerea elementelor acestea vor apărea pe suprafața de design.



Uneltele **Association** și **Inheritance** permit reprezentarea grafică a relațiilor dintre entități.



V.8. Lucrul cu mai multe tabele

Cele mai multe dintre aplicații combină informații din mai multe tabele. Pentru aceasta trebuie să utilizăm o interogare de tip JOIN așa cum am menționat în capitolul 5.5. pe baza relațiilor dintre cele două tabele.

În cazul exemplului ales avem două relații de tip 1-n între tabela Comenzi și tabela Clienți, respectiv Produse.

Definirea acestor relații presupune utilizarea obiectelor de tip **DataRelation**.

```
DataRelation Clienți_Comenzi = new DataRelation("Clienți_Comenzi",
ds.Tables["Clienți"].Columns["CodClient"],
ds.Tables["Comenzi"].Columns["CodClient"]);
```

```
DataRelation Produse_Comenzi = new DataRelation("Produse_Comenzi",
ds.Tables["Produse"].Columns["CodProdus"],
ds.Tables["Comenzi"].Columns["CodProdus"]);
```

```

ds.Relations.Add(Clienti_Comenzi);
ds.Relations.Add(Produse_Comenzi);

foreach (DataRow client in ds.Tables["Clienti"].Rows)
{
    text1.Text += "<br /><b>" + client["NumeClient"];
    text1.Text += " " + client["PrenumeClient"] + "</b><br />";
    foreach (DataRow comanda in client.GetChildRows(Clienti_Comenzi))
    {
        foreach (DataRow produs in comanda.GetParentRows(produse_Comenzi))
        {
            text1.Text += "&nbsp;&nbsp;&nbsp;";
            text1.Text += produs["DenumireProdus"] + "<br />";
        }
    }
}
}

```

Dacă vrem să modificăm aplicația prezentată și să afișăm o listă cu numărul de produse cumpărate de fiecare client trebuie să adăugăm în SqlDataAdapter informațiile din tabela Clienti și apoi cele din tabela Comenzi.

```

string selectSQL = "SELECT CodClient,NumeClient FROM Clienti";
conn = new SqlConnection
("Data Source='.\SQLEXPRESS';Initial Catalog=master; Integrated Security=SSPI");
SqlCommand cmd = new SqlCommand(selectSQL, conn);

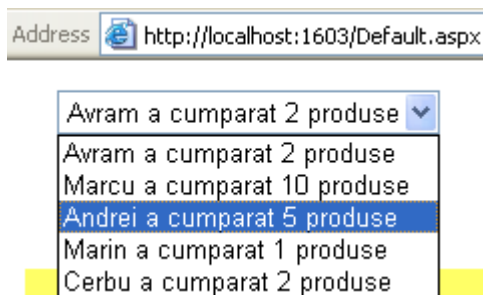
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
DropDownList2.Items.Clear();

try
{
    conn.Open();
    adapter.Fill(ds, "Clienti");
    cmd.CommandText = "SELECT CodClient,IdComanda,NumarProduse FROM
Comenzi";
    adapter.Fill(ds, "Comenzi");
}
catch (Exception err){text1.Text = "Eroare: " + err.Message;}
finally {conn.Close();}

foreach (DataRow client in dsPubs.Tables["Clienti"].Rows)
{
    ListItem newItem = new ListItem();
    newItem.Text = client["NumeClient"] + " a cumparat ";
    int x = 0;
    foreach (DataRow comanda in dsPubs.Tables["Comenzi"].Rows)
    {
        if(Convert.ToInt32(client["CodClient"])==Convert.ToInt32(comanda["CodClient"]))
            x += Convert.ToInt32(comanda["NumarProduse"]);
        newItem.Text += x + " produse";
        newItem.Value = client["CodClient"].ToString();
        DropDownList2.Items.Add(newItem);
    }
}

```

Lista va conține acum informații din cele două tabele relaționate.



V.9. PROCEDURI STOCATE (Stored Procedures)

O procedură stocată este o secvență de instrucțiuni SQL, salvată în baza de date, care poate fi apelată de aplicații diferite. Sql Server compilează procedurile stocate, ceea ce crește eficiența utilizării lor. De asemenea, procedurile stocate pot avea parametri.

O procedură stocată poate fi apelată folosind obiectul SqlCommand:

```
SqlCommand cmd = new SqlCommand("NumeProceduraStocata", conn);  
cmd.CommandType = CommandType.StoredProcedure;  
//Tipul obiectului comandă este procedura stocată
```

Primul parametru al constructorului este un șir de caractere ce reprezintă numele procedurii stocate. A doua instrucțiune de mai sus, spune obiectului SqlCommand ce tip de comandă va fi executată, prin intermediul proprietății CommandType.

Exemplu:

```
SqlCommand cmd = new SqlCommand("StoredProcedure1", conn);  
cmd.CommandType = CommandType.StoredProcedure;  
DsClienti = new DataSet();  
DaClienti = new SqlDataAdapter("", conn);  
DaClienti.SelectCommand = cmd;  
DaClienti.Fill(DsClienti, "Clienti");
```

Apelul procedurilor stocate parametrizate, este asemănător cu cel al interogărilor cu parametri.

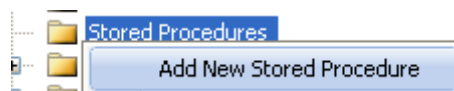
```
//Obiect Comanda, in care primul parametru este numele procedurii stocate
```

```
SqlCommand cmd = new SqlCommand("Localitate", conn);  
cmd.CommandType = CommandType.StoredProcedure;
```

```
//Tipul obiectului comandă este procedura stocată
cmd.Parameters.Add(new SqlParameter("@Localitate", inputLoc));
DsClienti = new DataSet();
DaClienti = new SqlDataAdapter("", conn);
DaClienti.SelectCommand = cmd;
DaClienti.Fill(DsClienti, "TabelaClienti");
```

Primul argument al constructorului obiectului **SqlCommand** este numele procedurii stocate. Această procedură are un parametru numit **@Localitate**. De aceea trebuie folosit un obiect de tip **SqlParameter** pentru a adauga acest parametru la obiectul de tip **Command**.

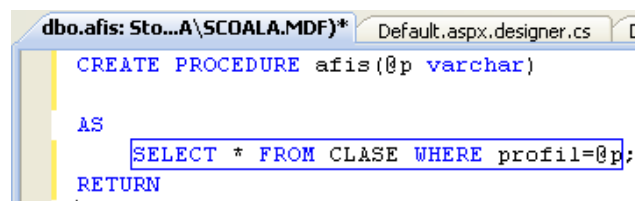
Vizual, o procedură stocată este generată după selectarea componentei **Stored**

Procedures din Database Explorer, alegerea opțiunii  și personalizarea șablonului:

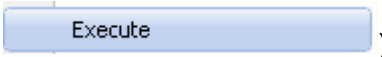
```
CREATE PROCEDURE dbo.numa
/*
(@parameter1 int = 5,@parameter2 datatype OUTPUT)
*/
AS
/* SET NOCOUNT ON */

RETURN
```

În exemplul următor se construiește procedura stocată **afis**, ce permite afișarea claselor cu un profil anume.

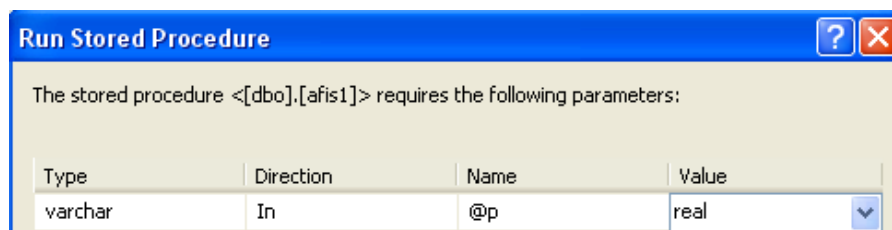


```
dbo.afis: Sto...A\SCOALA.MDF)* Default.aspx.designer.cs
CREATE PROCEDURE afis (@p varchar)
AS
SELECT * FROM CLASE WHERE profil=@p;
RETURN
```

După salvarea procedurii aceasta apare în Solution Explorer și poate fi executată (click dreapta pe numele procedurii și se alege opțiunea )

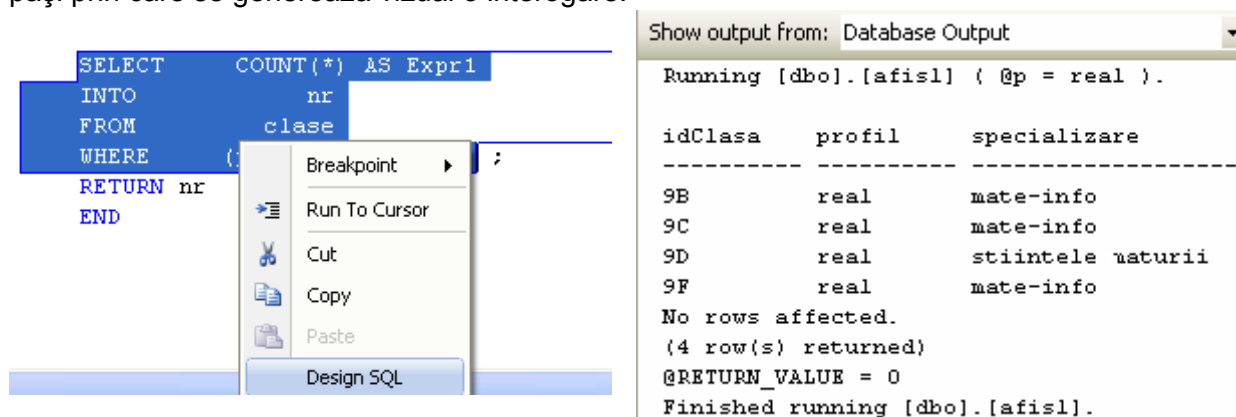


Executarea procedurii stocate **afis** necesită introducerea unei valori pentru parametrul **p**.

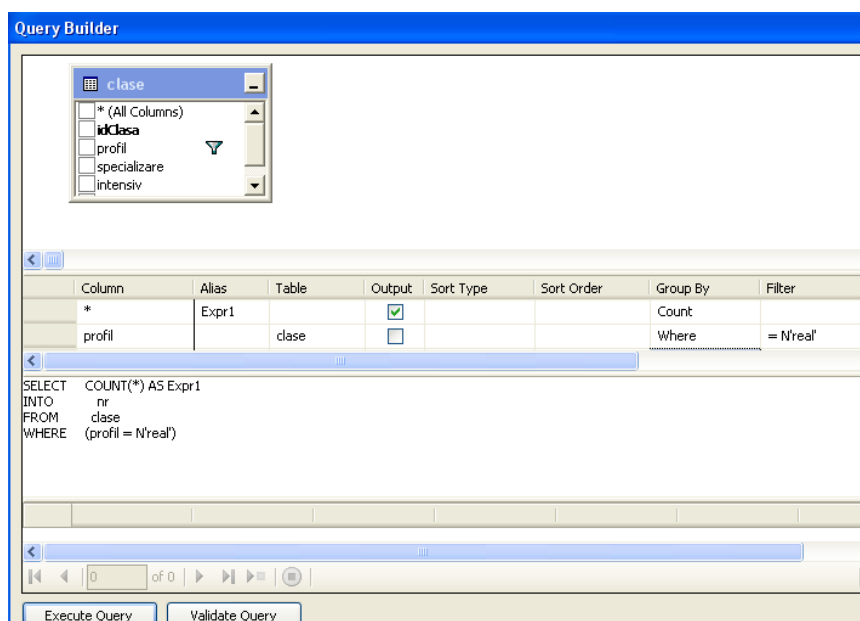


Rezultatul executării este afișat în fereastra Output.

Codul SQL dintr-o procedură stocată poate fi generat automat urmând aceiași pași prin care se generează vizual o interogare.

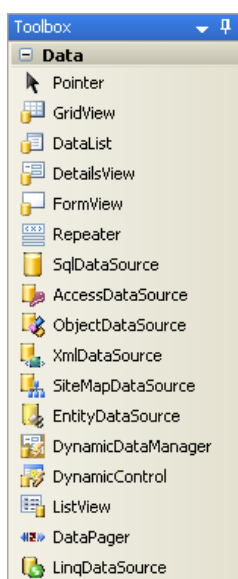


În exemplul următor se generează secvența de cod care atribuie variabilei **nr** numărul de clase de profil real.



Secvențele de cod SQL din procedura stocată pot fi selectate și executate separat.

V.10. Controale .NET legate la date



Majoritatea aplicațiilor software implică, într-un fel sau altul, accesarea și prezentarea datelor. Controalele legate la date joacă un rol cheie în dezvoltarea aplicațiilor .NET deoarece ele permit asocierea controalelor cu unul sau mai multe câmpuri din orice sursă de date. Legarea la date și controalele Web legate la date reprezintă punctul principal al oricărei aplicații ASP.NET.

Legarea la date⁶⁶ este procesul prin care se preiau date de la o sursă și se asociază dinamic unei proprietăți a unui element vizual. În funcție de contextul unde va fi afișat elementul el poate să corespundă unei etichete HTML sau unui control Web.NET.

Putem să construim astfel pagini legate la date fără să scriem secvențe de cod speciale pentru accesarea surselor de date.

V.10.1. Controale pentru sursa de date

Controalele de server Web ASP.NET pot fi legate la o bază de date prin folosirea controlului **DataSource**.



Toate controalele pentru sursa de date includ interfața **IDataSource** și se găsesc în secțiunea **Data** a meniului **Toolbox** din **Visual Studio**.

.NET Framework include următoarele controale pentru sursa de date:



SqlDataSource – permite conectarea la orice sursă de date care are un provider ADO.NET (SQL Server, Oracle, ODBC, OLE DB)⁶⁷;



AccessDataSource – permite conectarea la o bază de date Access .mdb⁶⁸;



ObjectDataSource – permite conectarea la o clasă personalizată definită pentru accesul la date. Această soluție este preferată de cei mai mulți dezvoltatori însă necesită scrierea unui volum mare de cod;




XmlDataSource – permite conectarea la un fișier XML;

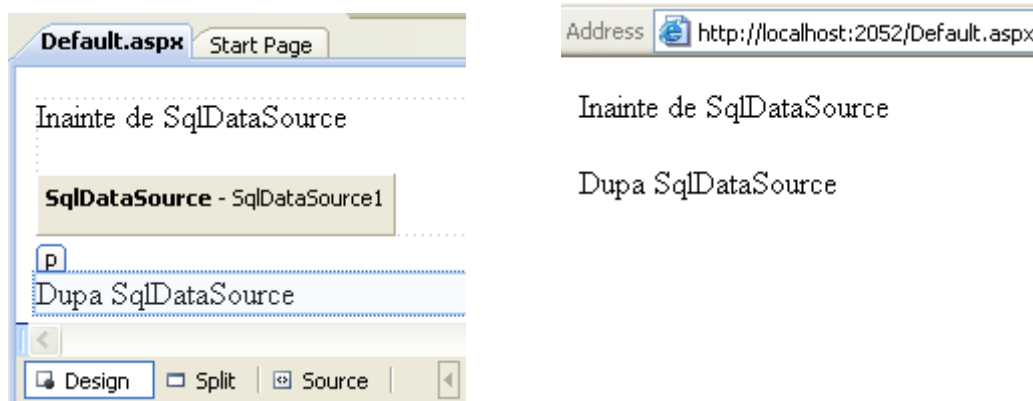
⁶⁶ În limba engleză **data binding**

⁶⁷ Capitolul 5.5.2

⁶⁸ Întrucât bazele de date Access nu au server dedicat (precum SQL Server) care să gestioneze accesul concurrent ele sunt indicate în cazul site-urilor mici pentru care numărul utilizatorilor simultani este redus.

 **SiteMapDataSource** – permite conectarea la un fișier de tip .sitemap care păstrează structura arborescentă a site-ului;

Atunci când adăugăm un control de legare la o sursă de date pentru aplicație el apare sub forma unui dreptunghi gri în modul proiectare (design) dar nu va fi vizibil atunci când se rulează aplicația Web și se încarcă pagina.



Controalele pentru sursele de date au două funcții principale:

- b) preiau datele din sursa de date și le furnizează controlului legat (**DataBind()**). Aceste date pot fi accesate prin intermediul controlului fără utilizarea unor funcții speciale.
- c) actualizează sursele de date atunci când au loc operații de editare.


Pentru a înțelege cum acționează controalele trebuie să înțelegem care este succesiunea de pași pe care-i parcurge o pagină Web care conține controale legate la date:


- a) se crează pagina (**fișier .aspx**);
- b) începe ciclul de viață al paginii cu executarea evenimentelor **Page.Init** și **Page.Load**;
- c) toate controalele de pe pagină sunt activate;
- d) dacă utilizatorul face o operație de modificare a datelor atunci controlul sursei de date execută această operație. Atunci când se modifică o înregistrare din tabelă au loc evenimentele **Updating** și **Updated**, dacă se inserează o linie atunci au loc evenimentele **Inserting** și **Inserted** iar dacă are loc o operație de ștergere atunci se produc evenimentele **Deleting** și **Deleted**;
- e) are loc evenimentul **Page.PreRender**;
- f) controalele pentru surse de date execută operațiile de interogare și furnizează datele controalelor legate la date. Acest pas are loc la primul acces la pagină și de fiecare dată când se revine la pagină astfel încât avem întotdeauna acces la datele cele mai recente. Au loc evenimentele **Selecting** și **Selected**.
- g) pagina este eliminată.


Bazele de date pot fi legate la mai multe tipuri de controale însă unele dintre acestea au fost proiectate **exclusiv pentru bazele de date**.


Printre acestea se numără **GridView**, **DetailsView**, și **FormView**, controale ce pot afișa mai multe câmpuri la un moment dat sub forma unui tabel sau a unei structuri bine definite și permit operații de selectare, editare și ordonare.

Cele mai puternice controale ce pot fi legate la baze de date sunt:

 **GridView** redă o grilă⁶⁹ multi-coloană, complet șablonată și permite afișarea tabelelor mari.

 **DetailsView** este ideal pentru afișarea la un moment dat a unei singure înregistrări dintr-o tabelă, într-un tabel ce are o singură linie pentru fiecare câmp și permite editarea;

 **FormView** afișează o singură înregistrare la un moment dat și permite operații de editare. Față de Details View are avantajul că șablonul de prezentare este flexibil, permițând combinarea mai multor câmpuri.

 **ListView** permite, ca și GridView, afișarea mai multor înregistrări simultan, diferența fiind dată de șablonul de afișare, construit cu mai multă muncă dar cu o mai mare flexibilitate.

V.10.2. Controlul GridView

Controlul **GridView** reprezintă o grilă foarte flexibilă în care sunt prezentate coloanele unei tabele din sursa de date. Fiecare linie din acest tabel corespunde unei înregistrări din sursa de date și fiecare câmp al înregistrării corespunde unei coloane din tabel.

Folosind acest control putem crea coloane simple legate la date, care prezintă datele preluate din sursa de date, coloane șablon, care permit proiectarea structurii grafice a conținutului de celule și coloane bazate pe comenzi, care permit adăugarea unor funcționalități specifice (selectare, editare, modificare, ștergere, paginare,ordonare).

Controlul **GridView** redă conținutul unor tabele organizate pe coloane și permite afișarea mai multor înregistrări în același timp.

După adăugarea unei grile în care fiecare coloană corespunde unui câmp pe pagină

```
<asp:GridView ID="GridView1" runat="server"/>
```

urmează asocierea datelor acestui control.

⁶⁹ în limba engleză **grid**

În următorul exemplu se utilizează obiecte ADO.NET și o interogare pentru a asocia controlul cu datele conținute în DataSet.

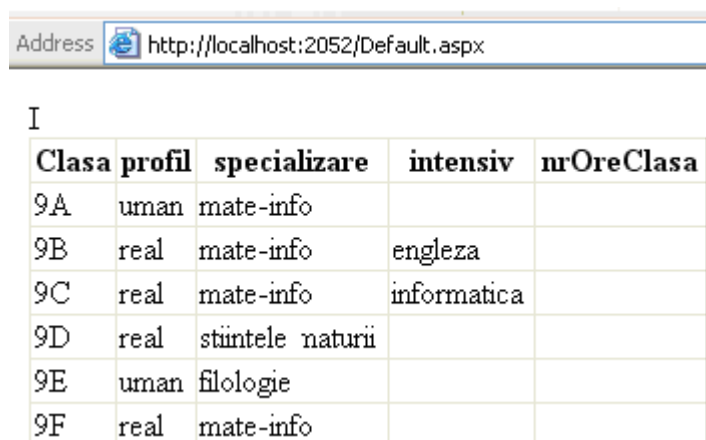
```
string connectionString =
WebConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
string selectSQL = "SELECT * FROM clase";
SqlConnection con = new SqlConnection(connectionString);
SqlCommand cmd = new SqlCommand(selectSQL, con);
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
adapter.Fill(ds, "clase");
GridView1.DataSource = ds;
GridView1.DataBind();
```

Pentru a evita introducerea manuală a acestui cod, putem să utilizăm controlul SqlDataSource pentru a menționa sursa de date

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%"$ ConnectionStrings:ConnectionString %>"
SelectCommand="SELECT * FROM [clase]"></asp:SqlDataSource>
```

și apoi putem lega datele la controlul GridView.

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" />
```




The screenshot shows a web browser window with the address bar displaying 'http://localhost:2052/Default.aspx'. Below the browser, a table is displayed with the following data:







Clasa	profil	specializare	intensiv	nrOreClasa
9A	uman	mate-info		
9B	real	mate-info	engleza	
9C	real	mate-info	informatica	
9D	real	stiintele naturii		
9E	uman	filologie		
9F	real	mate-info		

Definirea coloanelor

În mod implicit, proprietatea **GridView.AutoGenerateColumns** este **True** astfel încât se generează o coloană pentru fiecare câmp al tabeli legate. Această proprietate trebuie setată pe **False** atunci când dorim să afișăm doar o parte dintre coloane sau atunci când vrem să schimbăm ordinea în care sunt afișate sau chiar formatul de afișare. Selectarea coloanelor se va face în secțiunea **Columns** a tag-ului **GridView**.

Coloanele vor fi afișate în ordinea în care sunt scrise în această secțiune.

Coloanele pot avea tipuri diferite, pe lângă cele de bază în care afișează textul dintr-un câmp al unei coloane din tabelă ( BoundField) găsiindu-se coloane de tip:

tip	descriere
 ButtonField	afișează un buton
 CheckBoxField	afișează o casetă de validare și se utilizează pentru câmpurile ce pot avea una dintre valorile True/False
 CommandField	conține butoane pentru selecție sau editare
 HyperLinkField	afișează conținutul ca hyperlink.
 ImageField	afișează imaginea dintr-un câmp ⁷⁰
 TemplateField	permite afișarea câmpurilor folosind șabloane personalizate

Exemplul următor conține secvența ce definește coloanele controlului GridView asociate tabelii clase:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
  DataSourceID="SqlDataSource1" >
  <Columns>
    <asp:BoundField DataField="idClasa" HeaderText="Clasa" />
    <asp:BoundField DataField="profil" HeaderText="profil" />
    <asp:BoundField DataField="specializare" HeaderText="specializare"/>
    <asp:BoundField DataField="intensiv" HeaderText="intensiv" />
    <asp:BoundField DataField="nrOreClasa" HeaderText="nrOreClasa" />
  </Columns>
</asp:GridView>
```

Configurarea coloanelor

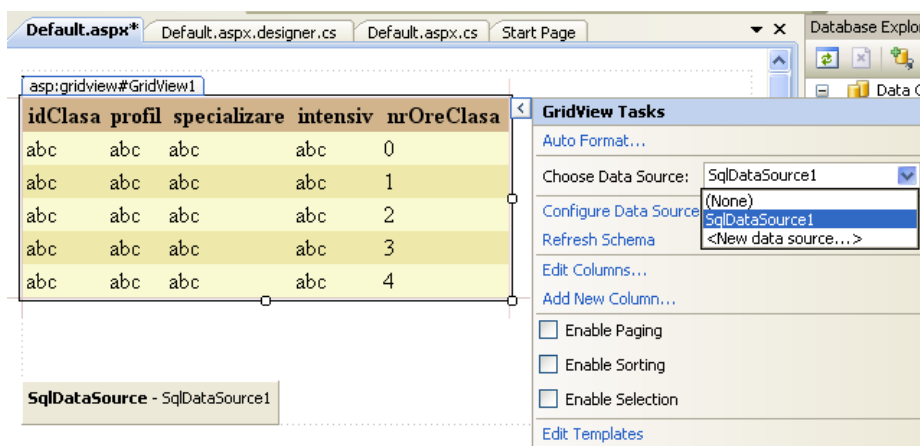
Atunci când declarăm explicit coloanele legate putem seta proprietățile acestora :

- **DataField** – numele câmpului ce va fi afișat în coloană
- **DataFormatString** – indică modul de formatare a datelor de pe coloană
- **FooterText/HeaderText/HeaderImageUrl**-setează textul/imaginea din antet /subsol
- **ReadOnly**- permite/inhibă modificarea conținutului coloanei în mod editare
- **InsertVisible**- permite/inhibă inserarea valorilor în coloană
- **Visible**- permite/inhibă afișarea coloanei
- **SortExpression**- permite ordonarea valorilor de pe această coloană
- **NullDisplayText**- afișează un text în locul valorilor NULL de pe coloană
- **ConvertEmptyStringToNull**- convertește șirurile vide de pe coloană la NULL
- **ControlStyle, HeaderStyle, FooterStyle, and ItemStyle** – stabilește modul în care sunt afișate datele pe acea coloană

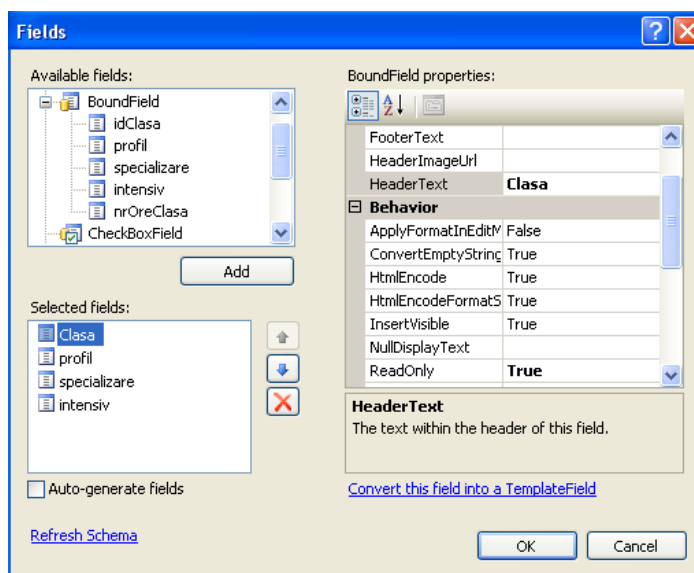
⁷⁰ furnizorul trebuie să suporte formatul în care este memorată imaginea

Generarea coloanelor în mod vizual

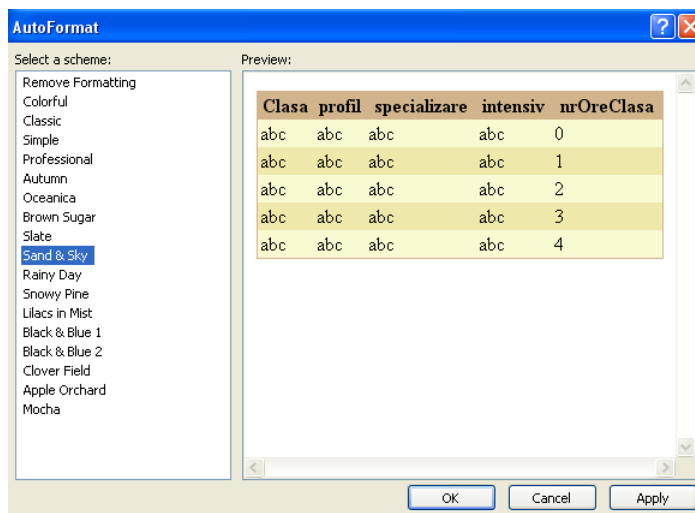
După adăugarea controlului GridView pe pagină (în modul design) și legarea acestuia la sursa de date




alegem opțiunea Edit Column și alegem câmpurile ce completează controlul. Pentru fiecare dintre câmpuri putem să modificăm proprietățile menționate anterior.



Opțiunea AutoFormat permite selectarea formatului de afișare.



Conținutul grilei va fi afișat pe pagină conform formatului stabilit:

Address  http://localhost:2052/Default.aspx

I

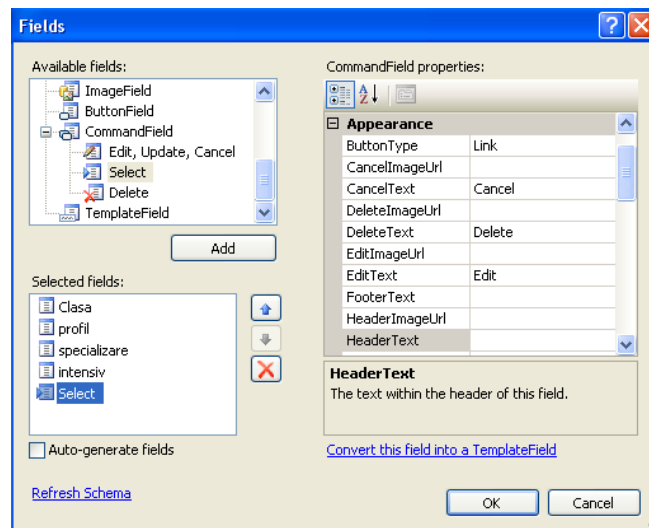
Clasa	profil	specializare	intensiv
9A	uman	mate-info	
9B	real	mate-info	engleza
9C	real	mate-info	informatica
9D	real	stiintele naturii	
9E	uman	filologie	
9F	real	mate-info	

Un control de tip GridView permite selectarea unui rând din tabel atunci când proprietatea **ShowSelectButton** a coloanei de tip **CommandField** este setată la **True**.

```
<asp:CommandField ShowSelectButton="True" />
```

Fiecare element de pe această coloană poate fi afișat ca legătură, ca buton sau ca imagine.


În exemplul următor butoanele Select sunt afișate ca legături.



Apăsarea unui buton Select de pe pagină duce la producerea următoarelor evenimente:

- **GridView.SelectedIndexChanging** - util atunci când dorim să anulăm operația
- proprietatea **GridView.SelectedIndex** este setată la indicele elementului selectat
- **GridView.SelectedIndexChanged** – permite extragerea de informații din rândul selectat. Pentru aceasta trebuie să selectăm proprietatea **DataKeyNames** la numele câmpului după care se face indexarea tabeli și se accesează datele.

```
< asp:GridView ID="GridView1" runat="server"
    DataKeyNames="idClasa" ... >
```

Address  http://localhost:2052/Default.aspx


I

Clasa	profil	specializare	intensiv	
9A	uman	mate-info		Select
9B	real	mate-info	engleza	Select
9C	real	mate-info	informatica	Select
9D	real	stiintele naturii		Select
9E	uman	filologie		Select
9F	real	mate-info		Select

Pentru a putea șterge sau modifica o înregistrare din tabela legată unui control GridView trebuie să definim comenzile de ștergere și actualizare ca în exemplu:


```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%"$ ConnectionStrings:ConnectionString %">"
  SelectCommand="SELECT * FROM [clase]"
  DeleteCommand="DELETE FROM [clase] WHERE idClasa=@idClasa"
  UpdateCommand="UPDATE clase SET profil=@profil,
  specializare=@specializare, intensiv=@intensiv WHERE idClasa=@idClasa">
</asp:SqlDataSource>
```

Pentru exemplificare, se modifică valoarea din câmpul „intensiv” pentru clasa 9F.

Address  http://localhost:2052/Default.aspx

I


Clasa	profil	specializare	intensiv	
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit
Select 9D	real	stiintele naturii		Delete Edit
Select 9E	uman	filologie		Delete Edit
9F	<input type="text" value="real"/>	<input type="text" value="mate-info"/>	<input type="text" value="intensiv"/>	Update Cancel

Address  http://localhost:2052/Default.aspx

I

Clasa	profil	specializare	intensiv	
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit
Select 9D	real	stiintele naturii		Delete Edit
Select 9E	uman	filologie		Delete Edit
Select 9F	real	mate-info	intensiv	Delete Edit

și se șterge clasa 9E.

Address  http://localhost:2052/Default.aspx


I

Clasa profil specializare intensiv				
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit
Select 9D	real	stiintele naturii		Delete Edit
Select 9F	real	mate-info	intensiv	Delete Edit

Controlul **GridView** permite efectuarea operațiilor de paginare și ordonare atunci când sunt selectate aceste proprietăți din **GridViewTasks**.


Ordonarea este declanșată prin simpla apăsare a numelui⁷¹ coloanei după care vrem să sortăm înregistrările.

În exemplul următor datele sunt ordonate mai întâi după profil și după aceea după specializare.

Address  http://localhost:2052/Default.aspx

I


Clasa profil specializare intensiv				
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit
Select 9D	real	stiintele naturii		Delete Edit
Select 9F	real	mate-info	intensiv	Delete Edit

Address  http://localhost:2052/Default.aspx

I

Clasa profil specializare intensiv				
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit
Select 9F	real	mate-info	intensiv	Delete Edit
Select 9D	real	stiintele naturii		Delete Edit

Numărul de înregistrări vizibile simultan este dat de proprietatea **Page Size**. În exemplul următor această proprietate a fost setată la 3.

Address  http://localhost:2052/Default.aspx

I

Clasa profil specializare intensiv				
Select 9A	uman	mate-info		Delete Edit
Select 9B	real	mate-info	engleza	Delete Edit
Select 9C	real	mate-info	informatica	Delete Edit

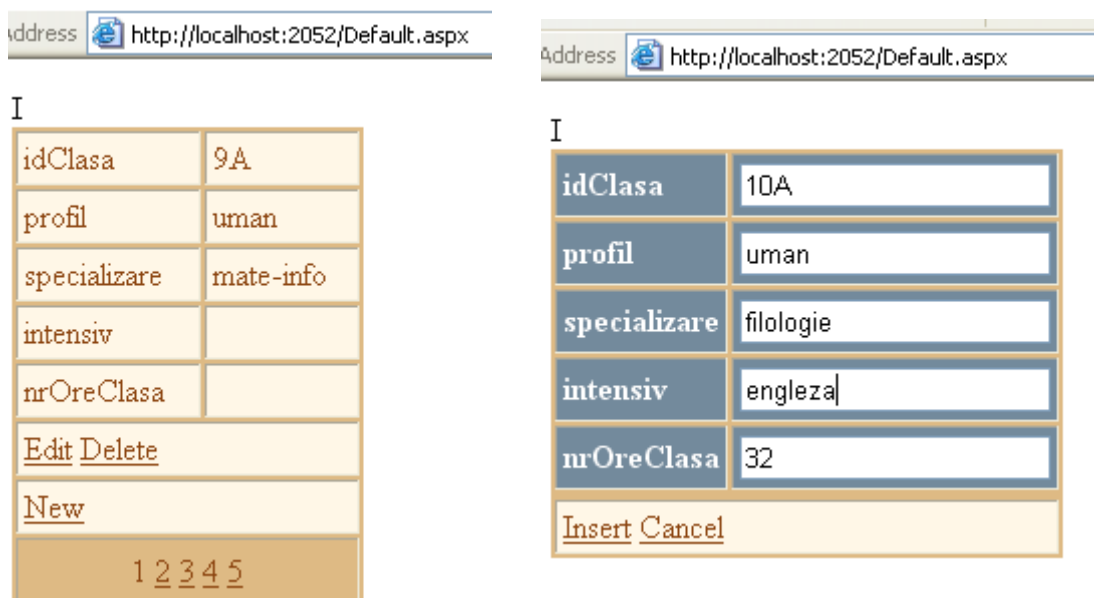
1 2

⁷¹ numele coloanelor sunt subliniate

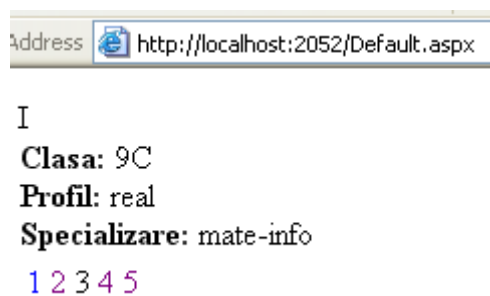
V.10.3. Controalele DetailsView și FormView

Controlul GridView permite afișarea mai multor înregistrări în același timp. Există situații în care este necesară afișarea unei singure înregistrări la un moment dat. Acest lucru poate fi realizat de controalele DetailsView și FormView, controale ce permit și operații de inserare. Diferența dintre cele două tipuri de controale este dată de faptul că Details View permite utilizarea șabloanelor (machtetelor) în timp ce FormView necesită utilizarea unui șablon de afișare și oferă o flexibilitate mai mare. O altă diferență este aceea că DetailsView afișează conținutul unei tabele în timp ce FormView permite și afișarea conținutului nelegat la o tabelă.

Controlul DetailsView afișează câte o înregistrare, fiecare câmp fiind afișat pe câte o linie în tabela de afișare. Ca și în cazul controlului GridView putem alege câmpurile care se afișează, putem insera butoane pentru ștergere și editare, și putem stabili formatul de afișare. Apăsarea butonului New permite inserarea unei noi înregistrări în tabela sursă.



Controlul DetailsView afișează câte o singură înregistrare la un moment dat. Dacă sursa de date are mai multe înregistrări atunci va fi vizibilă doar prima. Acest lucru poate fi evitat dacă se setează **AllowPaging=true**. În acest fel utilizatorul se poate deplasa de la o înregistrare la alta.



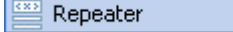
```

<asp:FormView ID="FormView1" runat="server" AllowPaging="True"
  DataSourceID="SqlDataSource1">
  <ItemTemplate>
  <b>Clasa:</b>
  <%# Eval("idClasa") %><br />
  <b>Profil:</b>
  <%# Eval("profil") %><br />
  <b>Specializare:</b>
  <%# Eval("specializare") %><br />
  </ItemTemplate>
</asp:FormView>

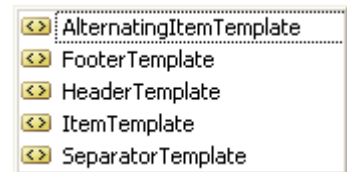
```

V.10.4. Alte controale legate la date

V.10.4.1 Repeater

Controlul  creează o structură grafică personalizată pentru afișarea conținutului controlului prin repetarea unui șablon specificat pentru fiecare element din lista legată. Se definește câte un șablon ASP.NET pentru fiecare categorie de elemente: antet, subsol, separator, etc.) iar controlul îl aplică repetat pe pagină.

Controlul **Repeater** nu conține nicio structură grafică implicită, deci trebuie declarate explicit toate etichetele de formatare și stil.




Pentru fiecare șablon definit, codul ASP.NET este evaluat dinamic în timpul executării metodei DataBind.

În exemplul următor se definește un control Repeater care afișează înregistrările tabelii clase.

```

<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">

```

Address  http://localhost:2052/Default.aspx

Clasa	Profil	Specializare
9A	uman	mate-info
9B	real	mate-info
9C	real	mate-info
9D	real	științele naturii
9F	real	mate-info
Clase		

Șablonul Header este procesat o singură dată și inițiază grafic tabela și titlul.

```
<HeaderTemplate>
  <table style ="border:1px solid black;"class="stdtext">
    <thead bgcolor="#ff6666" style:white">
      <td><b> Clasa </b></td>
      <td><b> Profil </b></td>
      <td><b> Specializare </b></td>
    </thead>
  </HeaderTemplate>
```

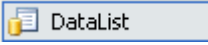
Controlul permite afișarea elementelor/elementelor alternante cu setări grafice diferite .

```
<ItemTemplate>
  <tr>
    <td bgcolor="#ffcc99">
      <b><%# DataBinder.Eval(Container.DataItem,"idClasa") %></b></td>
    <td bgcolor="#ffcc99">
      <%# DataBinder.Eval(Container.DataItem,"profil")%></td>
    <td bgcolor="#ffcc99">
      <%# DataBinder.Eval(Container.DataItem,"specializare") %> </td>
  </tr>
</ItemTemplate>
<AlternatingItemTemplate>
  <tr>
    <td bgcolor="#ffffcc">
      <b><%# DataBinder.Eval(Container.DataItem,"idClasa") %></b></td>
    <td bgcolor="#ffffcc">
      <%# DataBinder.Eval(Container.DataItem,"profil") %> </td>
    <td bgcolor="#ffffcc">
      <%# DataBinder.Eval(Container.DataItem,"specializare") %> </td>
  </tr>
</AlternatingItemTemplate>
```

Șablonul Footer este afișat, ca și șablonul Header, o singură dată. Acest șablon definește rândul final.

```
<FooterTemplate>
  <tfoot>
    <td bgcolor="#ffcc66" colspan="3"><%# " Clasa" %></td>
  </tfoot>
</table>
</FooterTemplate>
```

V.10.4.2 DataList

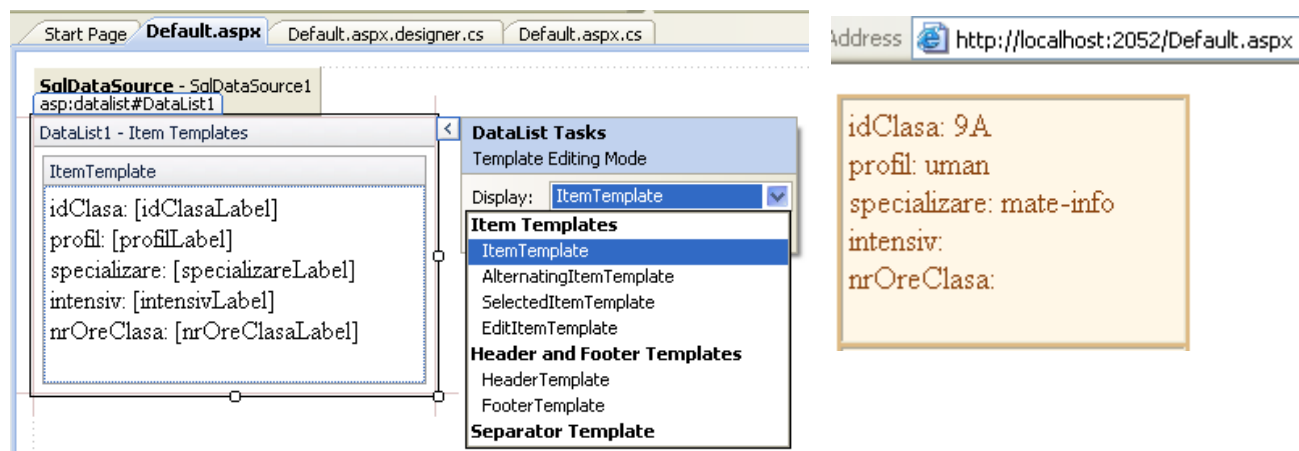
Controlul  afișează conținutul unei liste legate la date prin intermediul șabloanelor ASP.NET. Acest control permite operații de editare și selectare. Aspectul

controlului poate fi personalizat prin setarea de proprietăți stil pentru diferite părți ale controlului.

Controlul **DataList** prezintă structuri grafice predefinite, posibilități avansate de formatare și suport pentru selectare și editare.

Operațiile fundamentale de legare la date care folosesc DataList sunt similare celor care folosesc Repeater. Se folosește proprietatea DataSource pentru a lega controlul la date și metoda DataBind pentru a actualiza interfața utilizator.


Controlul DataList suportă două șabloane speciale **SelectedItemTemplate** și **EditItemTemplate**.



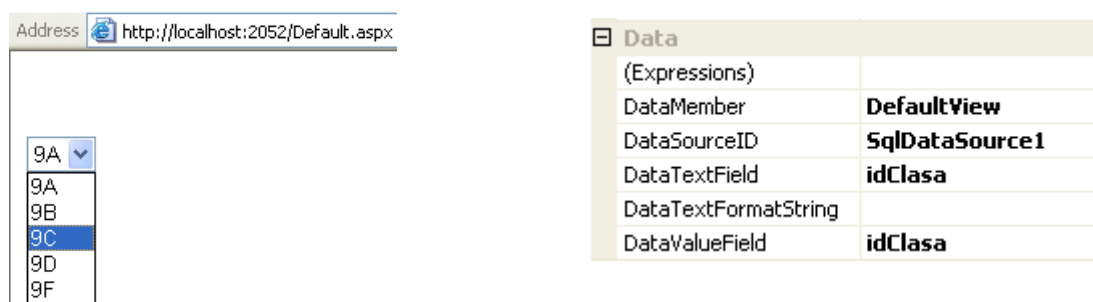
SelectedItemTemplate controlează cum este afișat elementul selectat iar **EditItemTemplate** specifică șablonul grafic pentru rândul editat.

Controlul DataList oferă suport special pentru comenzile predefinite: edit, update, delete, cancel și select. Evenimentele predefinite asociate (**EditCommand**, **UpdateCommand**, **DeleteCommand**, **CancelCommand**, **SelectedIndexChanged**) controlului DataList se declanșează atunci când trebuie să se execute comanda corespunzătoare.

V.10.4.3 DropDownList

Controlul  permite utilizatorului să aleagă un element dintr-o listă derulantă. Proprietățile **SelectedIndex** și **SelectedItem** oferă detalii despre elementul selectat. Legarea la date se realizează prin utilizarea proprietăților: **DataSource**, **DataMember**, **DataTextField**, **DataValueField** și **DataTextFormatString**.

În exemplul următor elementele listei derulante sunt preluate din tabela clase.



The screenshot shows a web browser window with the address bar displaying 'http://localhost:2052/Default.aspx'. Below the address bar is a dropdown menu with the following options: 9A, 9B, 9C (highlighted), 9D, and 9F. To the right of the browser window is a table titled 'Data' with the following content:


(Expressions)	
DataMember	DefaultView
DataSourceID	SqlDataSource1
DataTextField	idClasa
DataTextFormatString	
DataValueField	idClasa

```
<asp:DropDownList ID="DropDownList1" runat="server" DataMember="DefaultView"
DataSourceID="SqlDataSource1"
DataTextField="idClasa"
DataValueField="idClasa">
</asp:DropDownList>
```

Valorile pentru **DataTextField** și **DataValueField** trebuie să corespundă numelui unui câmp din sursa de date. Dacă dorim ca o listă derulantă să afișeze informații din mai multe câmpuri atunci putem cere ca SQL Server să întoarcă o coloană cu formatul dorit și atribuim numele câmpului precalculat proprietății **DataTextField**. De exemplu, dacă dorim ca lista derulantă să afișeze numele clasei și profilul facem atribuirea **DataTextField= "CP"** și selecția

```
SELECT idClasa+ "+ profil AS "CP" FROM clase;
```

V.10.4.4 CheckBoxList

Controlul  poate fi privit ca un control părinte pentru un set de elemente din listă care pot fi bifate, fiecare fiind redat printr-un control **CheckBox** individual. O listă de casete de validare se inserează astfel:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server"
DataSourceID="SqlDataSource1"
DataTextField="idClasa"
DataValueField="idClasa">
</asp:CheckBoxList>
```


Legarea la date se realizează prin utilizarea proprietăților: **DataSource**, **DataMember**, **DataTextField**, **DataValueField** și **DataTextFormatString**.

Spre deosebire de controlul **DropDownList**, controlul **CheckBoxList** nu are proprietăți care furnizează direct informații despre elementele selectate.

La un moment dat , un control CheckBoxList poate avea unul sau mai multe elemente selectate.

Toate controalele listă au o proprietate **Items** (elemente) care conține setul de elemente fii. Proprietatea Items este implementată prin intermediul clasei **ListItemCollection**, iar fiecare element din listă poate fi accesat printr-un obiect **ListItem**.


În exemplul următor se construiește un șir cu etichetele elementelor selectate.

Address  http://localhost:2052/

- 9A
- 9B
- 9C
- 9D
- 9F

```
StringBuilder s = new StringBuilder("");
foreach (ListItem x in CheckBoxList1.Items)
{
    if (x.Selected)
    {
        s.Append(x.Text);
        s.Append(" ");
    }
}
```


V.10.4.5 RadioButtonList

Controlul  gestionează o colecție de elemente de listă de tip butoane radio, redade prin intermediul unor controale de tip RadioButton.

Un control **RadioButtonList** poate avea ori un singur element selectat, ori nici unul. Proprietatea SelectedItem întoarce elementul selectat ca obiect ListItem.

Conținutul unui control RadioButtonList, obținut dintr-o sursă de date, poate fi setat astfel:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server"
DataSourceID="SqlDataSource1"
DataTextField="idClasa"
DataValueField="idClasa">
</asp:RadioButtonList>
```

Address  http://localhost:2052/

Legarea la date se realizează prin utilizarea proprietățile: **DataSource**, **DataMember**, **DataTextField**, **DataValueField** și **DataTextFormatString**.

Aceste proprietăți se comportă în același mod ca proprietățile controalelor descrise anterior.

- 9A
- 9B
- 9C
- 9D
- 9F



V.10.5. Evaluare

1) Pentru toate cerințele de mai jos se consideră următorul scenariu:

Administratorul unei firme trebuie să țină o evidență a departamentelor firmei și a angajaților care lucrează în aceste departamente. Un angajat nu poate lucra în același timp la mai multe departamente. În timp, un angajat poate fi mutat de la un departament la altul. Fiecare departament are un șef .

- a) contruiți un model conceptual care să corespundă cadrului prezentat și să conțină cel puțin două tabele : **angajati**, **departamente**.
- b) adăugați înregistrări în tabele folosind modul vizual
- c) generați diagrama corespunzătoare modelului conceptual ales
- d) proiectați o pagină Web care să vă permită efectuarea următoarelor operații (pentru fiecare operație se utilizează controler-ul indicat)
 - d.1 afișarea angajaților din firmă (**GridView**)
 - d.2 afișarea departamentelor din firmă (**DetailsView**)
 - d.3 afișarea angajaților dintr-un departament (**FormView**)
 - d.4 afișarea șefilor de departament (**Repeater**)
 - d.5 afișarea informațiilor referitoare la unul dintre angajați (**DataList**)
 - d.6 ștergerea unui angajat din baza de date (**DataList**)
 - d.7 mutarea unui angajat de la un departament la altul (**DropDownList**)
 - d.8 ștergerea unui departament din baza de date (implicit a angajaților care lucrează în acest departament) (**RadioButtonList**)
 - d.9 adăugarea unui angajat (**GridView**)
 - d.10 adăugarea unui departament (**DetailsView**)
 - d.11 afișarea numărului de angajați din mai multe departamente (**CheckBoxList**)

2) Construiți o pagină Web care să permită gestiunea unui depozit de materiale, știind că : au loc operații de vânzare și aprovizionare, se pot vizualiza informații referitoare la stocul de materiale, persoanele care cumpără din depozit sunt persoane fizice sau juridice.

3) Construiți o pagină Web care să permită vânzarea de bilete on-line la diferite spectacole.

4) Realizați o aplicație care să implementeze activitatea desfășurată la o grădină zoologică (evidența animalelor, angajaților, etc)

5) Realizați o aplicație care să reprezinte site-ul unui muzeu virtual.

VI. Securitatea aplicațiilor ASP.NET

Pentru o aplicație securizată, avem mai multe posibilități de autentificare, cele mai des întâlnite fiind sintetizate în tabelul următor. Implementarea politicii de securitate se poate face atât din IIS cât și din aplicația ASP.NET.

Tipul aplicației	Modul de autentificare	Descriere
Aplicație web publică pe Internet.	Anonim	Nu avem nevoie de securizare.
Aplicație web pentru Intranet.	Windows Integrated	Acest mod autentifică utilizatorii folosind lista de utilizatori de pe server (Domain Controller). Drepturile utilizatorilor în aplicația web este dat de nivelul de privilegii al contului respectiv.
Aplicație web disponibilă pe Internet, dar cu acces privat.	Windows Integrated	Utilizatorii companiei pot accesa aplicația din afara Intranetului, folosind conturi din lista serverului (Domain Controller).
Aplicații web comerciale.	Forms Authentication	Aplicații care au nevoie de informații confidențiale și eventual în care sunt mai multe tipuri de utilizatori.

VI.1. Windows Authentication

În acest mod de autentificare, aplicația ASP .NET are încorporate procedurile de autentificare, dar se bazează pe sistemul de operare Windows pentru autentificarea utilizatorului.

1. Utilizatorul solicită o pagină securizată de la aplicația Web.
2. Cererea ajunge la Serverul Web IIS care compară datele de autentificare ale utilizatorului cu cele ale aplicației (sau ale domeniului)
3. Dacă acestea două nu corespund, IIS refuză cererea utilizatorului.
4. Calculatorul clientului generează o fereastră de autentificare,
5. Clientul introduce datele de autentificare, după care retrimite cererea către IIS
6. IIS verifică datele de autentificare, și în cazul în care sunt corecte, direcționează cererea către aplicația Web.
7. Pagina securizată este returnată utilizatorului.

VI.2.Forms-Based Authentication

Atunci când se utilizează autentificarea bazată pe formulare, IIS nu realizează autentificarea, deci este necesar ca în setările acestuia să fie permis accesul anonim.

1. În momentul în care un utilizator solicită o pagină securizată, IIS autentifică clientul ca fiind un utilizator anonim, după care trimite cererea către ASP.NET.
2. Acesta verifică pe calculatorul clientului prezența unui anumit cookie¹
3. Dacă cookie-ul nu este prezent sau este invalid, ASP.NET refuză cererea clientului și returnează o pagină de autentificare (Login.aspx).
4. Clientul completează informațiile cerute în pagina de autentificare și apoi trimite informațiile.
5. Din nou, IIS autentifică clientul ca fiind un utilizator anonim și trimite cererea către ASP.NET
6. ASP.NET autentifică clientul pe baza informațiilor furnizate. De asemenea generează și un cookie. Cookie reprezintă un mic fișier text ce păstrează diverse informații despre utilizatorul respectiv, informații folosite la următoarea vizită a sa pe site-ul respectiv, la autentificare, sau în diverse alte scopuri.
7. Pagina securizată cerută și noul cookie sunt returnate clientului. Atâta timp cât acest cookie rămâne valid, clientul poate solicita și vizualiza orice pagină securizată ce utilizează aceleași informații de autentificare.

VI.3.Securizarea din aplicația web

Securizarea unei aplicații web presupune realizarea a două obiective:(1) autentificarea și (2) autorizarea.

1. **Autentificarea** presupune introducerea de către utilizator a unor credențiale, de exemplu nume de utilizator și parolă, iar apoi verificarea în sistem că acestea există și sunt valide.
2. **Autorizarea** este procesul prin care un utilizator autentificat primește acces pe anumite resurse, numai pe resursele pe care are dreptul să le acceseze.

Aceste obiective pot fi atinse foarte ușor utilizând funcționalitățile și uneltele din ASP.NET respectiv Visual Studio, anume clasa *Membership* și unealta *ASP.NET Configuration* (din meniul Website al Visual Studio Web Developer Express). Configurarea autentificării și autorizării se poate realiza după cum se vede în acest tutorial: [http://msdn2.microsoft.com/en-us/library/879kf95c\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/879kf95c(VS.80).aspx) .

VII. Proiectarea și realizarea unei aplicații Web

Pentru realizarea unui site web dinamic, trebuie parcurse următoarele etape:

- Proiectarea interfeței
- Proiectarea bazei de date
- Realizarea logicii aplicației (așa numitul „business logic”)

În continuare vom prezenta, pas cu pas, realizarea unui site web dinamic care oferă informații despre filme.

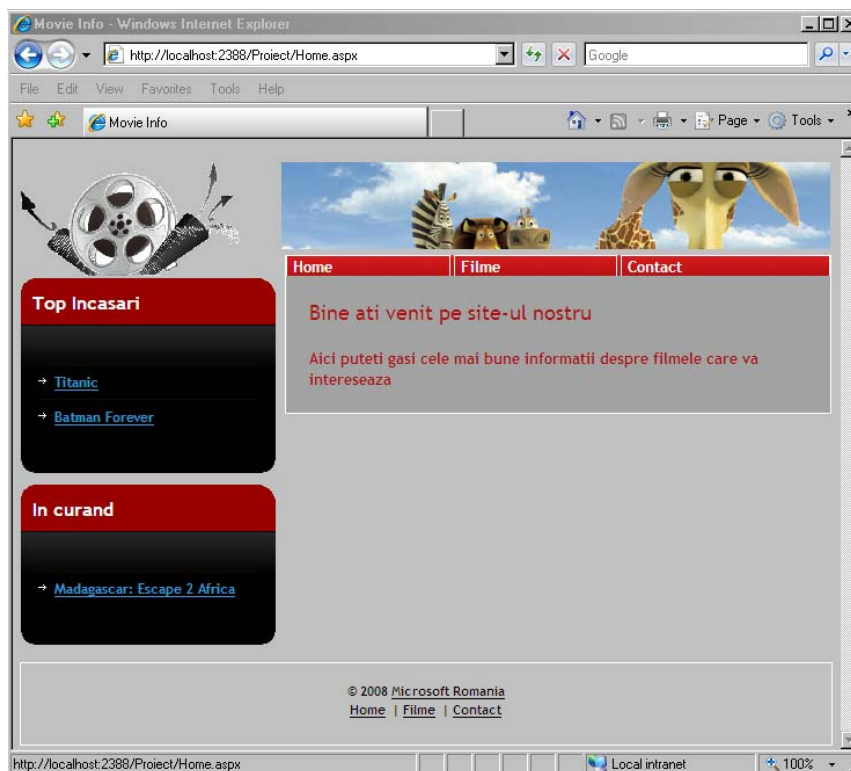


Figura 7-1 Interfața site-ului web dinamic

VII.1. Realizarea interfeței

ASP.NET pune la dispoziția programatorului câteva unelte puternice pentru dezvoltarea interfeței unui site web:

1. MasterPages
2. Foi de stil (StyleSheets)

VII.1.1. MasterPages

Pentru a obține un aspect unitar al site-ului, se folosește o pagină specială numită MasterPage. Aceasta este o machetă pe baza căreia vor fi realizate toate paginile din site-ul web. În afara conținutului moștenit din MasterPage, fiecare pagină .aspx va avea și un conținut propriu.

Pentru a adăuga o pagină MasterPage, din meniul *Website* se alege opțiunea *Add New Item*, și apoi *MasterPage*:

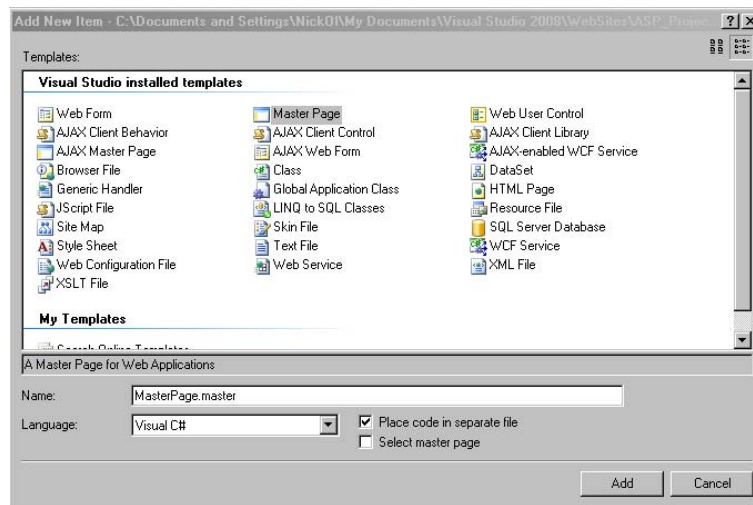


Figura 7.2 Adăugarea unei pagini MasterPage

MasterPage conține o zonă numită *ContentPlaceHolder* unde va fi inserat conținutul paginilor .aspx construite pe baza acestui template.

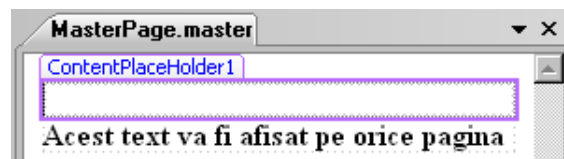


Figura 7.3 Zona ContentPlaceHolder va reține conținutul fiecărei pagini .aspx

Codul asp asociat este:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Pagina de Index</title>
<asp:ContentPlaceHolder id="head" runat="server"></asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
<asp:Label ID="Label1" runat="server" Font-Bold="True"
Text="Acest text va fi afisat pe orice pagina">
</asp:Label>
</div>
</form>
</body>
</html>
```

La adăugarea în proiect a unei noi pagini .aspx, se poate selecta pagina MasterPage care conține layout-ul dorit pentru site-ul web.

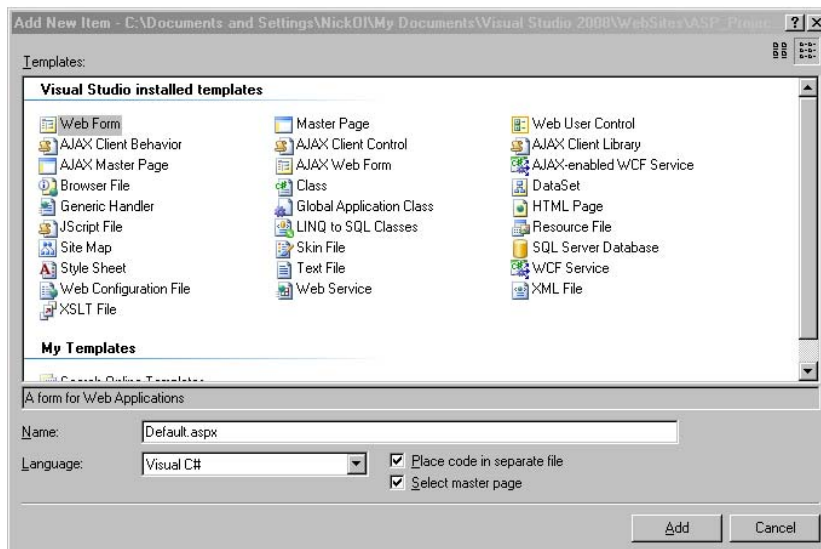


Figura 7.4 La adăugarea unei noi pagini .aspx se bifează opțiunea Select master page

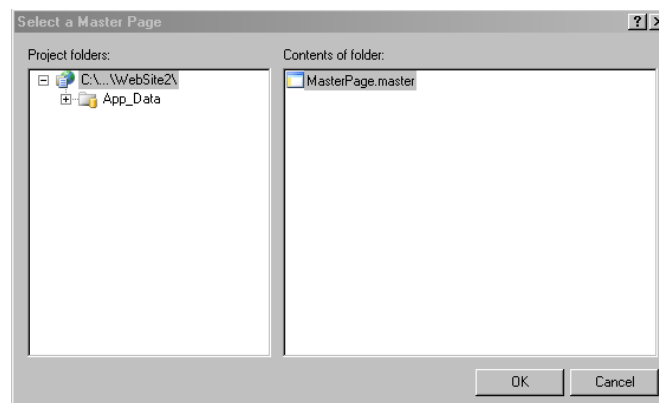


Figura 7.5 Selectarea MasterPage pentru pagina web nou adăugată

Pagina .aspx adăugată are următoarea structură:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
Title="Untitled Page" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
<p>
Acesta este continutul paginii adaugate
</p>
</asp:Content>
```

În zona de directive a apărut o nouă clauză care specifică numele fișierului MasterPage: `MasterPageFile = "~/MasterPage.master"`.

Fiecare pagină aspx va avea un control de tip *Content*, al cărui id va avea aceeași valoare cu cel din MasterPage:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
    <!--Continutul paginii .aspx -->
</asp:Content>
```

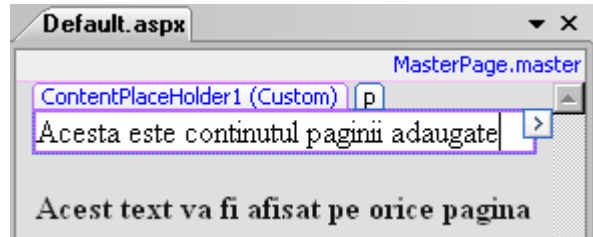


Figura 7.6 Pagina .aspx construită pe baza unei MasterPage

În cazul proiectului nostru, fișierul MasterPage va avea următorul design:

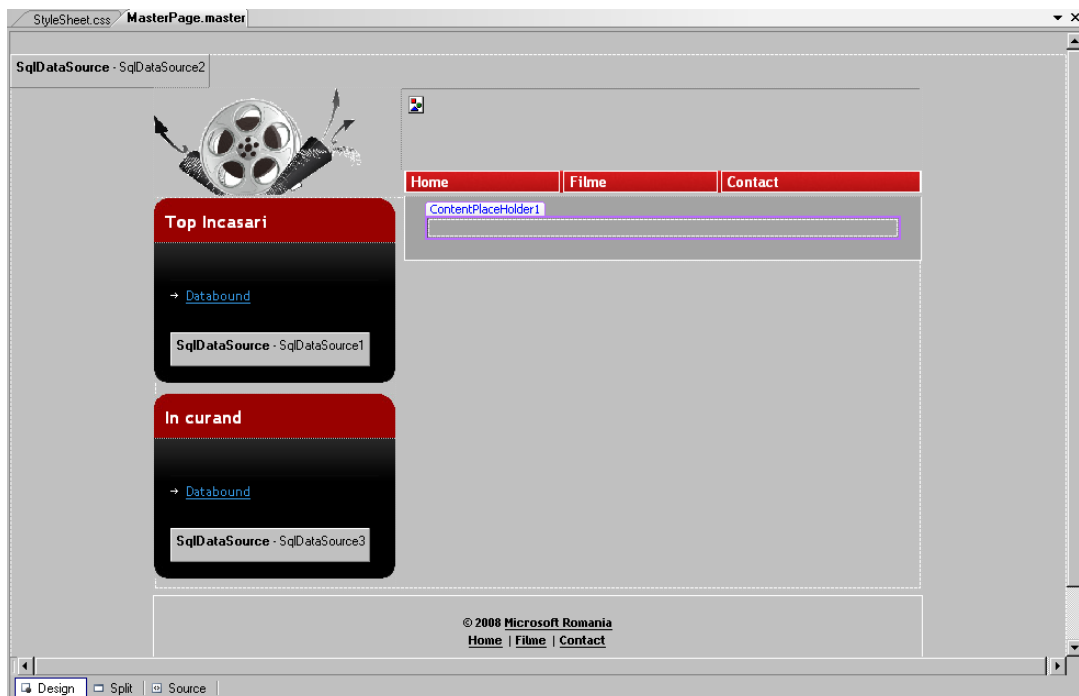


Figura 7.7 MasterPage în modul design

Pe lângă fișierul MasterPage.master, vom adăuga următoarele pagini .aspx:

- **Home.aspx** – prima pagină a site-ului, care va conține un mesaj de întâmpinare.
- **Movie.aspx** – va afișa o listă cu toate filmele din baza de date, sau anumite filme conform unui criteriu de filtrare.
- **Detalii.aspx** – permite utilizatorului modificarea informațiilor despre un film selectat.
- **AdaugăFilm.aspx** – permite utilizatorului adăugarea unui nou film în baza de date.
- **Upload.aspx** – este un formular de upload a unui fișier imagine, ce reprezintă posterul unui film.
- **Contact.aspx** – este un formular de contact, ce permite trimiterea unui e-mail cu datele introduse în formular.

Layout-ului paginii MasterPage poate fi realizat prin intermediul tabelelor sau prin intermediul tag-urilor html <div> formatate prin declarații css. În exemplul de mai sus, pentru definirea layout-ului s-au folosit tag-uri <div>:

Zona de header:

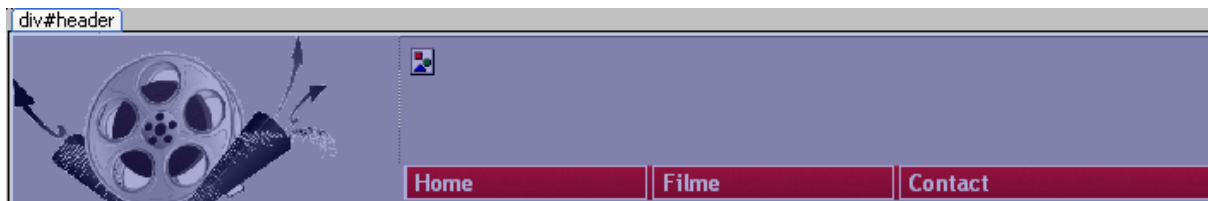


Figura 7.8 Div-ul care conține zona de header a paginii

Header-ul are la rândul său două secțiuni:

- Secțiunea de logo

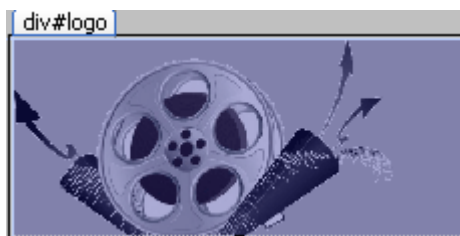


Figura 7.9 Div-ul care conține secțiunea de logo a header-ului

- Secțiunea pentru banner

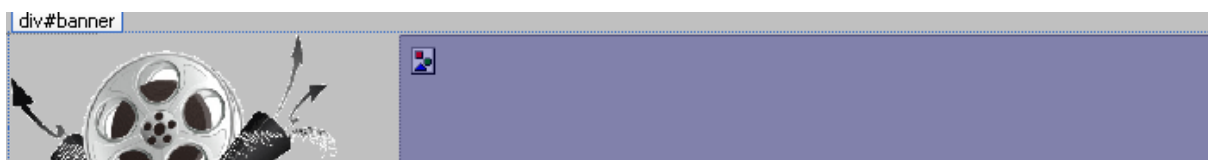


Figura 7.10 Div-ul care conține secțiunea pentru banner a header-ului

Zona pentru conținutul paginii:



Figura 7.11 Div-ul pentru conținutul paginii

Pagina este formată la rândul ei din:

- a) Coloana din stânga, ce conține două casete cu informații

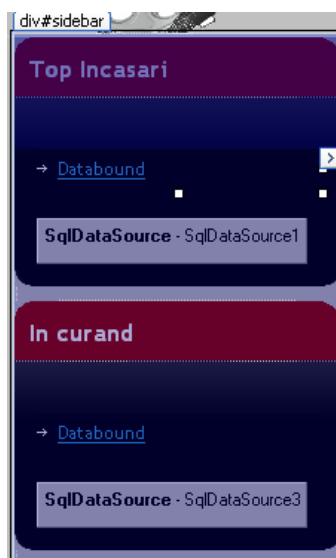


Figura 7.12 Div-ul pentru coloana din stânga paginii

Coloana este formată din două casete de informații, fiecare fiind definită de un div.

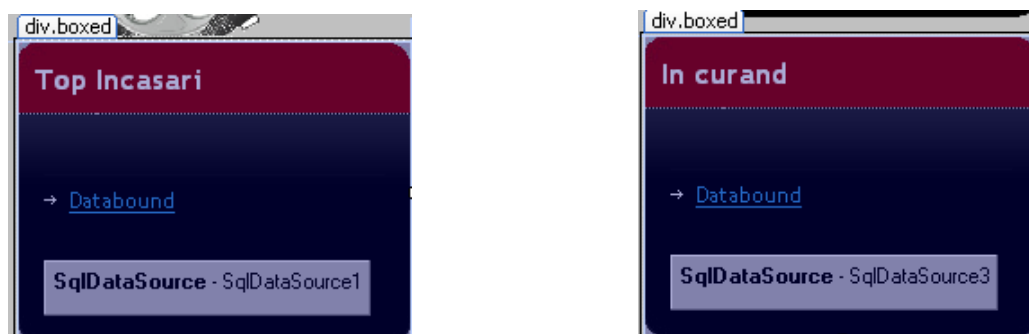


Figura 7.13 Div-ul pentru casetele din stânga paginii

b) Zona de conținut a paginii, care cuprinde și controlul *ContentPlaceHolder*



Figura 7.14 Div-ul pentru conținutul paginii

Zona de subsol a paginii web:

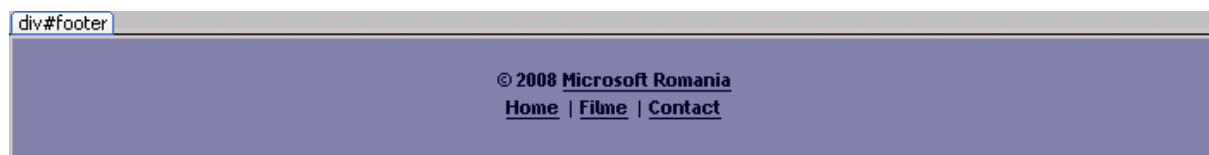


Figura 7.15 Div-ul pentru zona de subsol a paginii

Codul pentru definirea layout-ului din fișierul MasterPage.master este următorul:

```

<%@ Master Language="C#" AutoEventWireup="true"
    CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

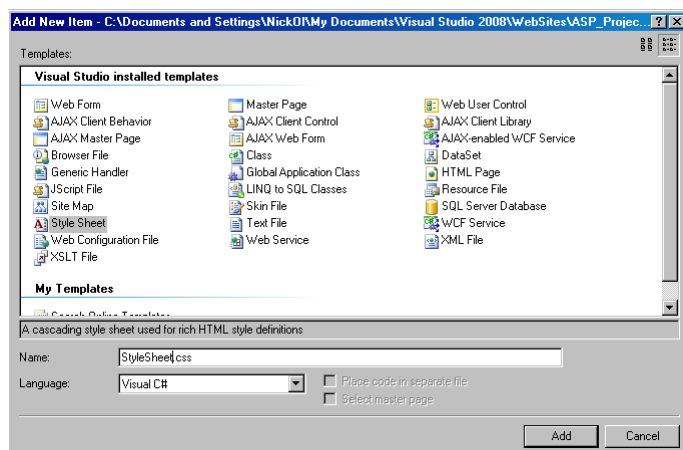
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
<title>Movie Info</title>
<asp:ContentPlaceHolder id="head" runat="server"></asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div id="header">
<div id="logo"></div>
<div id="banner"></div>
</div>
<div id="page">
<div id="content">
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
</div>
<div id="sidebar">
<div class="boxed">
<h2 class="title">Top Incasari</h2>
<div class="content"></div>
</div>
<div class="boxed">
<h2 class="title">In curand</h2>
<div class="content"></div>
</div>
</div>
</div>
<div id="footer"></div>
</form>
</body>
</html>

```

VII.1.2. Foi de stiluri

Pentru a adăuga un fișier css în cadrul site-ului web, se alege opțiunea *Add New Item* și apoi *Style Sheet*:

Figura 7.16 Adăugarea unui fișier ce conține declarații css



Formatarea tag-urilor html din paginile .aspx va fi realizată prin adăugarea în secțiunea head din MasterPage a liniei

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```

Observație: Acest lucru se poate realiza și prin drag and drop a fișierului StyleSheet.css (din fereastra Solution Explorer) în secțiunea de head a fișierului MasterPage.master.

Visual Web Developer pune la dispoziție un editor css foarte puternic. Având deschis fișierul StyleSheet.css puteți avea acces la opțiunile *Add Style Rule*, respectiv *Build Style*.

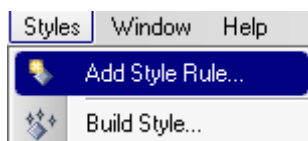


Figura 7.17 Opțiunile din meniul Styles pentru construirea foilor de stil

Cu ajutorul opțiunii Add Style Rule puteți defini o nouă regulă de stil, o nouă clasă sau o regulă de formatare pentru un tag html identificat printr-un id.

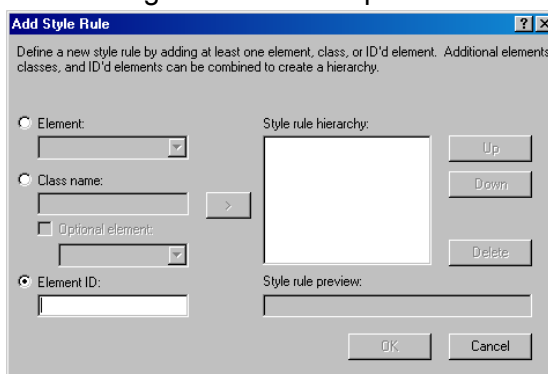


Figura 7.18 Definirea unei reguli de stil

Odată definită o declarație css, puteți stabili formatarea prin intermediul opțiunii Build Style.

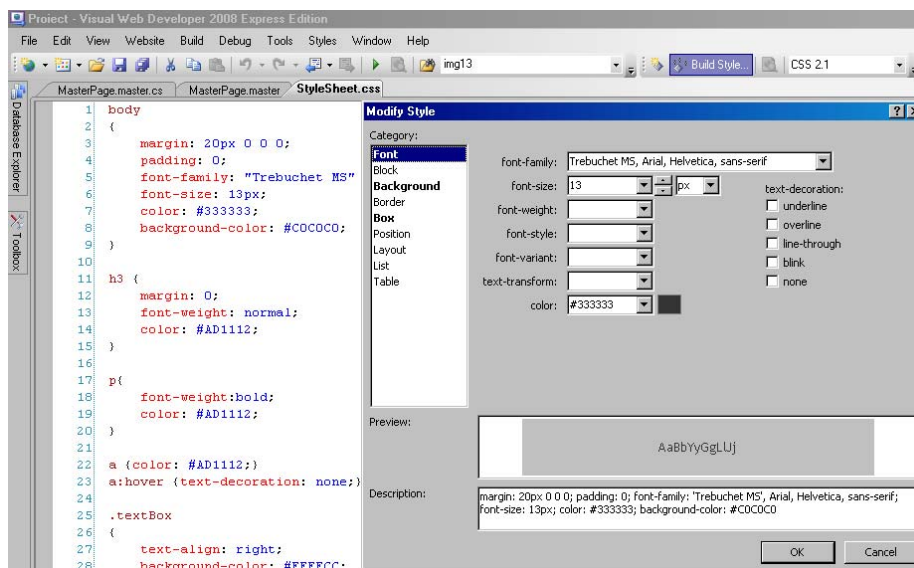


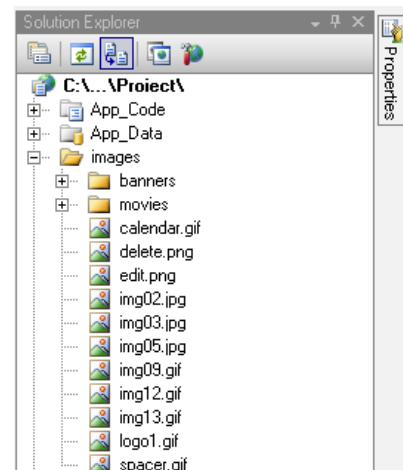
Figura 7.19 Definirea stilurilor de formatare

Pentru proiectul nostru, fișierul **StyleSheet.css** are următorul conținut:

```
body{margin: 20px 0 0 0;padding: 0;
font-family: "Trebuchet MS" , Arial, Helvetica, sans-serif;
font-size: 13px; color: #333333; background-color: #C0C0C0;}
h3 {margin: 0; font-weight: normal; color: #AD1112;}
p {font-weight:bold; color: #AD1112;}
a {color: #AD1112;}
a:hover {text-decoration: none;}
.textBox { text-align: right;background-color:#FFFFCC;border:1.5px solid #000000;}
.button {background-color: #990000; font-weight: bold; color: #FFFFFF;}
/* Header */ #header {width: 700px; height: 98px; margin: 0 auto;}
/* Logo */ #logo {float: left;width: 225px;height: 98px;}
/* Meniu */
.menu {float: right; width: 472px; height: 1px; background: url(images/img05.jpg)
repeat-x;border-left: 1px solid #FFFFFF; border-right: 1px solid #FFFFFF; }
.menu a{color: White;}
/* Pagina */ #page {width: 700px; margin: 0 auto; }
/* Continutul paginii */
#content { border-left: 1px solid #FFFFFF; border-right: 1px solid #FFFFFF;
border-bottom: 1px solid #FFFFFF; float: right; width: 430px; padding: 20px;
background: #a3a3a3;}
/* Coloana din stânga */
#sidebar {float: left;width: 220px;padding-top: 2px;}
#sidebar a {color: #3699E4;}
#sidebar ul {margin-left: 0;padding-left: 0;list-style: none;font-size: 92%;}
#sidebar ul li {padding: 5px 14px;background: url(images/img13.gif) no-repeat 0px
12px;border-top: 1px solid #0B0B0B;color: #3699E4;}
#sidebar ul li h2 {margin: 0; font-size: 100%;}
#sidebar ul li h3 {margin: 0; font-size: 92%;}
#sidebar ul li p {margin: 0;}
.boxed {margin-bottom: 10px;
background: #000000 url(images/img03.jpg) no-repeat left bottom;}
.boxed .title { height: 30px; margin: 0; padding: 10px 0 0 10px;
background: #5EB2ED url(images/img02.jpg) no-repeat;font-size: 16px;
font-weight: bold; color: #FFFFFF;}
.boxed .content { padding: 15px; background: url(images/img09.gif) repeat-x;}
/* Footer */
#footer{ text-align: center; width: 700px; height: 70px; margin: 0 auto;
border: 1px solid #FFFFFF; margin-top: 5px; font-weight: bold;}
#footer p { margin: 0;line-height: normal; font-size: 85%; color: #000000;font-
weight: bold;}
#footer a {color: #000000;font-weight: bold;}
```

Imaginile folosite ca fundal pentru diferitele secțiuni, precum și alte imagini folosite în site, au fost adăugate în prealabil în cadrul proiectului, într-un director *images*.

Figura 7.20 Directorul images din cadrul site-ului web



Pentru un control server web puteți folosi proprietatea `CssClass` pentru a-i asocia o clasă css definită în fișierul `StyleSheet.css`. De exemplu, controlul `Menu1` din `MasterPage` are asociată clasa css `menu`.

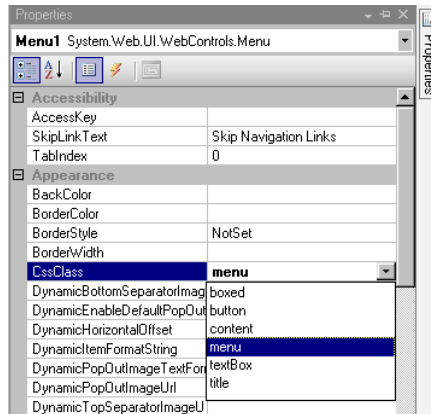


Figura 7.21 Adăugarea unei clase css pentru un control web

VII.1.3. Controalele web server din MasterPage

Pentru a finaliza layout-ul paginii master, vom adăuga următoarele controale web server:

Pentru secțiunea de logo, un obiect *hyperlink*.

```
<div id="logo">
  <asp:HyperLink ID="HyperLink5" runat="server"
    ImageUrl="~/images/logo1.gif" NavigateUrl="~/Home.aspx">Movie Info
  </asp:HyperLink>
</div>
```

În secțiunea de banner vom folosi un control de tip *AdRotator*, care va afișa bannere de filme. Controlul *AdRotator* va fi conectat la o sursă de date. De asemenea, pentru navigarea între paginile site-ului, vom utiliza un control de tip *Menu*.

```
<div id="header">
  <div id="logo">
    <asp:HyperLink ID="HyperLink5" runat="server" ImageUrl="~/images/logo1.gif"
      NavigateUrl="~/Home.aspx"> Movie Info
    </asp:HyperLink>
  </div>
  <div id = "banner">
    <asp:AdRotator ID="AdRotator1" runat="server" Height="75px"
      Target="_blank" Width="474px" />
    <asp:Menu ID="Menu1" runat="server" Orientation="Horizontal"
      CssClass="menu">
      <StaticMenuStyle VerticalPadding="0px" />
      <StaticMenuItemStyle Font-Bold="True" BorderColor="White"
        BorderStyle="Solid" BorderWidth="1px" HorizontalPadding="5px" />
      <StaticHoverStyle BackColor="#990000" ForeColor="Blue" />
    <Items>
      <asp:MenuItem Text="Home" Value="Home" NavigateUrl="~/Home.aspx">
      </asp:MenuItem>
      <asp:MenuItem Text="Filme" Value="Filme" NavigateUrl="~/Movie.aspx">
      </asp:MenuItem>
    </Items>
  </div>
</div>
```

```

<asp:MenuItem Text="Contact"Value="Contact"NavigateUrl="~/Contact.aspx">
</asp:MenuItem>
</Items>
</asp:Menu>
</div>
</div>

```

Prima casetă de informații va afișa primele 5 filme în ordinea descrescătoare a încasărilor. A doua casetă de informații va conține primele 5 filme care urmează să apară, în ordinea crescătoare a datei. Informațiile din ambele casete sunt afișate prin intermediul unor controale de tip `BulletedList`, conectate la o sursă de date.

```

<div id="page">
  <div id="content">
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>
  </div>
  <div id="sidebar">
    <div class="boxed"><h2 class="title">Top Incasari</h2>
    <div class="content">
      <asp:BulletedList ID="BulletedList1" runat="server"DisplayMode="LinkButton"
        DataValueField="id" onclick="BulletedList1_Click" Target="_parent">
      </asp:BulletedList>
    </div>
  </div>
  <div class="boxed"><h2 class="title">In curand</h2>
  <div class="content">
    <asp:BulletedList ID="BulletedList2" runat="server"DisplayMode="LinkButton"
      onclick="BulletedList2_Click">
    </asp:BulletedList>
  </div>
</div>
</div>
</div>

```

În zona de subsol a paginii vom folosi controale *Hyperlink*:

```

<div id="footer">
  <p>&nbsp;</p>
  <p>&copy; 2008
    <asp:HyperLink ID="HyperLink6" runat="server" Font-Bold="True"
      NavigateUrl="http://www.microsoft.com/romania/educatie/cursnet/default.mspx"
      Target="_blank">Microsoft Romania</asp:HyperLink>
  </p>
  <p id="links">
    <asp:HyperLink ID="HyperLink1" runat="server"NavigateUrl="~/Home.aspx">Home
    </asp:HyperLink> &nbsp;|
    <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/Movie.aspx">Filme
    </asp:HyperLink> &nbsp;|
    <asp:HyperLink ID="HyperLink4" runat="server"NavigateUrl="~/Contact.aspx">Contact
    </asp:HyperLink>
  </p>
</div>

```

Proiectarea bazei de date

Pentru a adăuga o bază de date în cadrul site-ului web, în fereastra *Solution Explorer* apăsați click dreapta pe directorul *App_Data* și apoi alegeți opțiunea *Add New Item*. În noua fereastră selectați *Sql Server Database*. Noua bază de date va fi denumită *filme.mdf*.

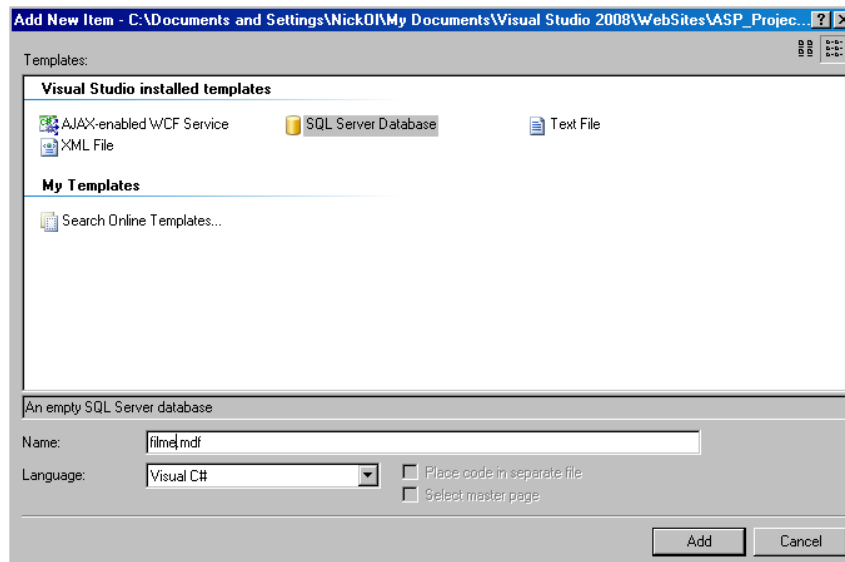


Figura 7.22 Adăugarea unei baze de date sql server în cadrul site-ului web

Pentru a păstra simplitatea site-ului web, informațiile despre filme le vom reține într-o singură tabelă, deși aceasta nu se găsește în forma normală 3. Structura tabelii *film* este următoarea:

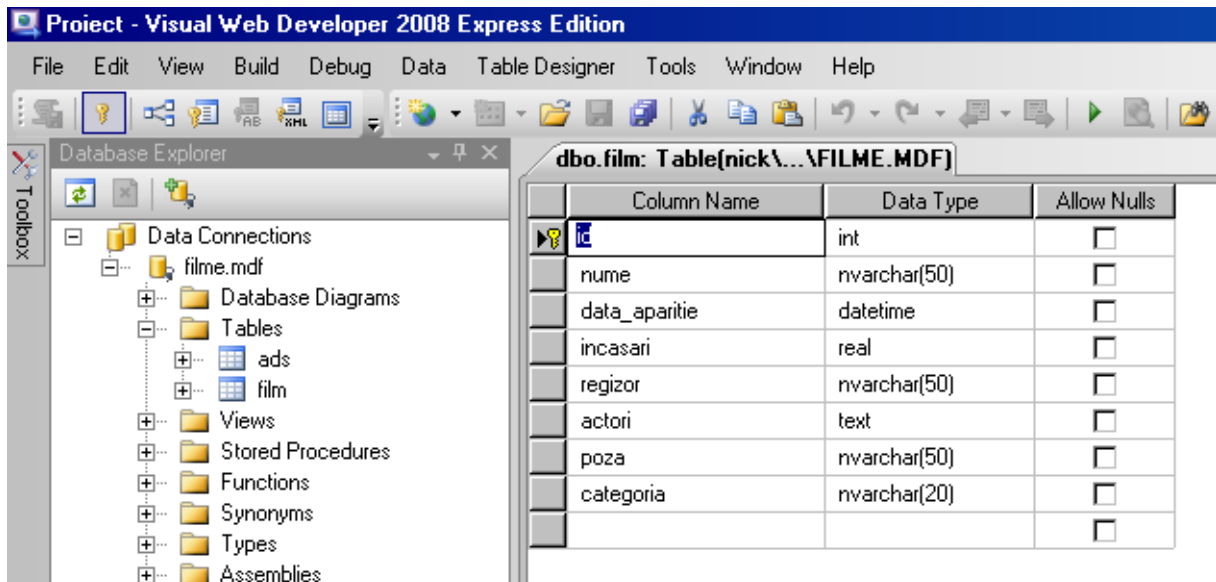


Figura 7.23 Structura bazei de date film

Câmpul *id* este cheie primară. Ca și în Access, MySQL sau Oracle, și în Sql Server puteți defini un câmp pentru care valorile se autoincrementează. Pentru aceasta, trebuie setată proprietatea *Is Identity* cu valoarea *yes*.

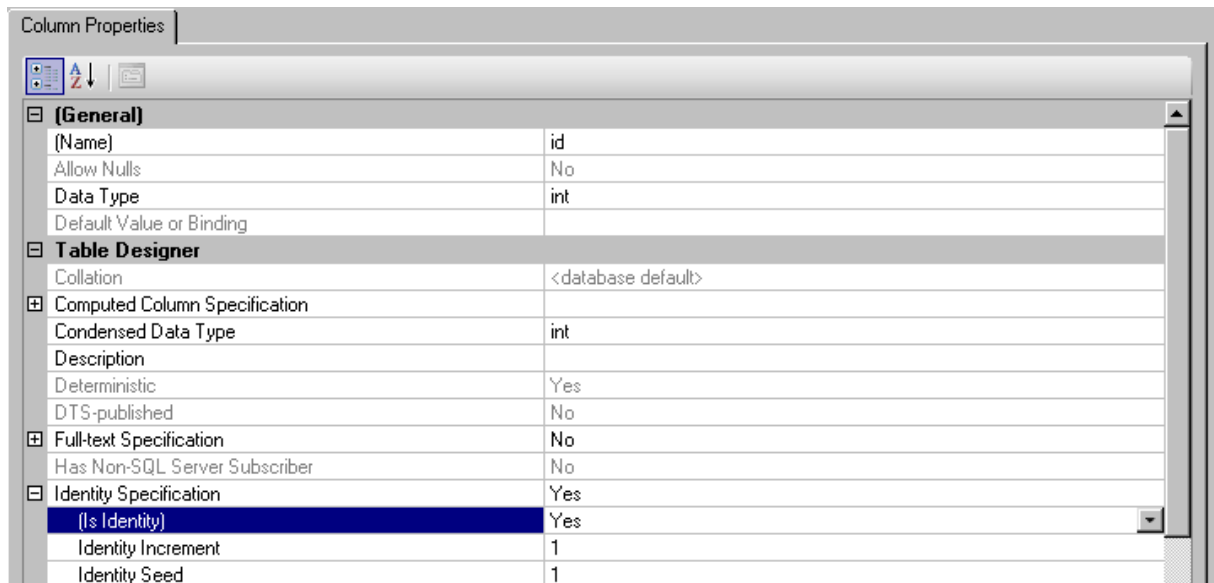


Figura 7.24 Setarea proprietății Is Identity

În baza de date filme vom mai adăuga o nouă tabelă numită *ads*, care va constitui sursa de date pentru controlul *AdRotator* din *MasterPage*. Această tabelă trebuie să aibă structura:

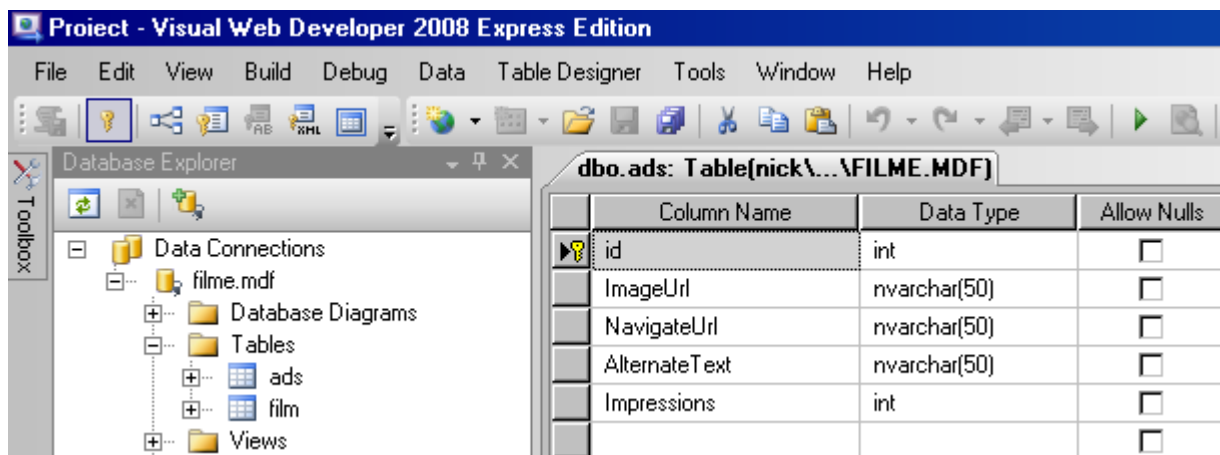


Figura 7.25 Structura tabelii ads

De asemenea, pentru câmpul *id* trebuie setată cu valoarea *yes* proprietatea *Is Identity*. Câmpul *ImageUrl* va conține calea către fișierul ce conține imaginea afișată în controlul *AdRotator*. În cadrul proiectului a fost creat directorul *banners* în directorul *images*, aici fiind salvate fișierele imagine.

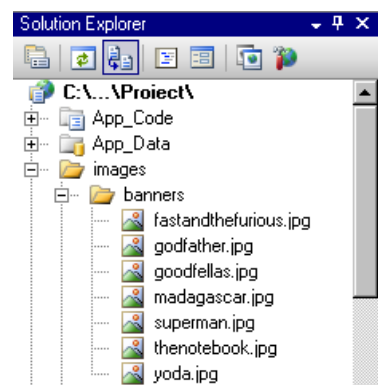


Figura 7.26 Fișierele imagine pentru controlul AdRotator.

În tabela *ads* sunt inserate următoarele înregistrări:

ads: Query(nick\...\FILME.MDF)					
	id	ImageUrl	NavigateUrl	AlternateText	Impressions
▶	2	~/images/banners/yoda.jpg	http://www.starwars.com/	Star Wars	20
	5	~/images/banners/fastandthefurious.jpg	http://www.thefastandthefurious.com/	Fast and Furious	10
	6	~/images/banners/godfather.jpg	http://en.wikipedia.org/wiki/The_Godfather	The Godfather	30
	10	~/images/banners/goodfellas.jpg	http://en.wikipedia.org/wiki/Goodfellas	Goodfellas	30
	13	~/images/banners/madagascar.jpg	http://www.madagascar-themovie.com/	Madagascar	20
	14	~/images/banners/superman.jpg	http://www.supermanhomepage.com	Superman	20
	15	~/images/banners/thenotebook.jpg	http://en.wikipedia.org/wiki/The_Notebook_(film)	The notebook	30
*	NULL	NULL	NULL	NULL	NULL

Figura 7.27 Datele din tabela ads

VII.1.4. Conectarea la sursa de date a controalelor din MasterPage

Pentru controlul AdRotator selectăm ca sursă de date o baza de date Sql Server.

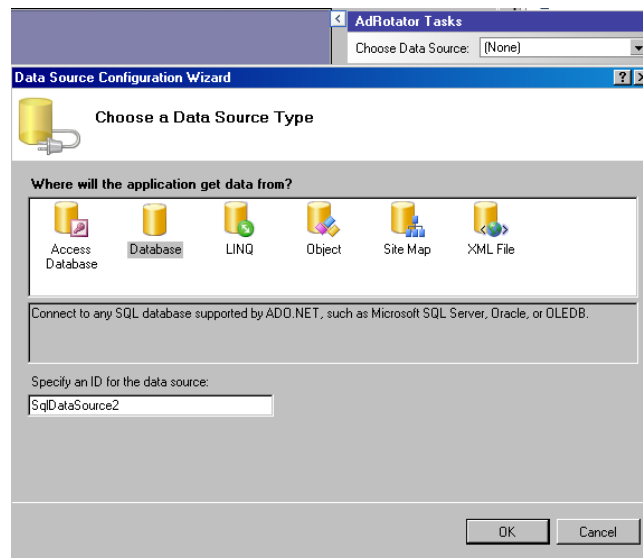


Figura 7.28 Alegerea tipului sursei de date pentru controlul AdRotator

La pasul următor, alegem opțiunea New Connection, și selectăm baza de date filme.

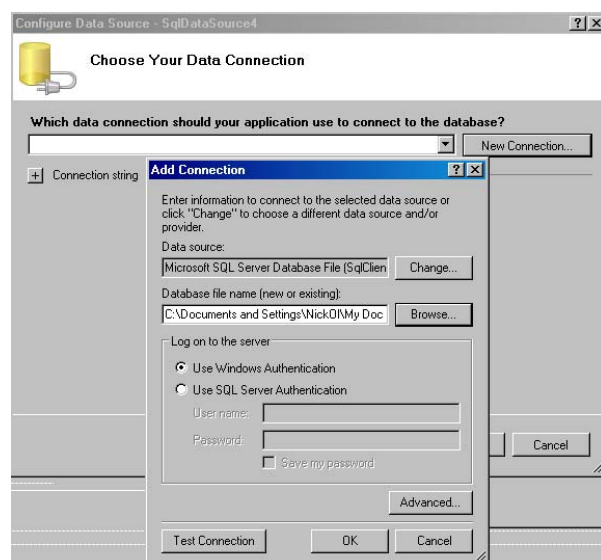


Figura 7.29 Selectarea bazei de date filme ca sursă de date

La pasul următor salvăm *connection string* pentru a-l putea reutiliza și la celelalte surse de date. Acest șir de caractere va fi salvat în fișierul *web.config*.

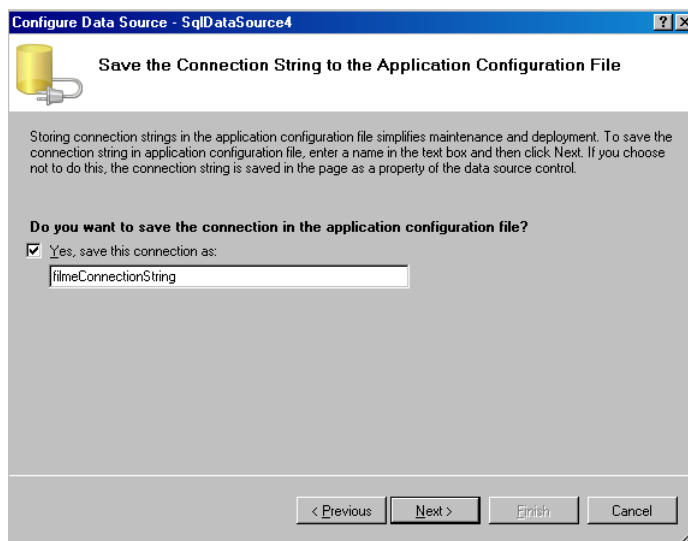


Figura 7.30 Salvarea Connection String

Pentru controlul AdRotator avem nevoie de toate înregistrările din tabelă, deci vom folosi interogarea: **SELECT * FROM ads**

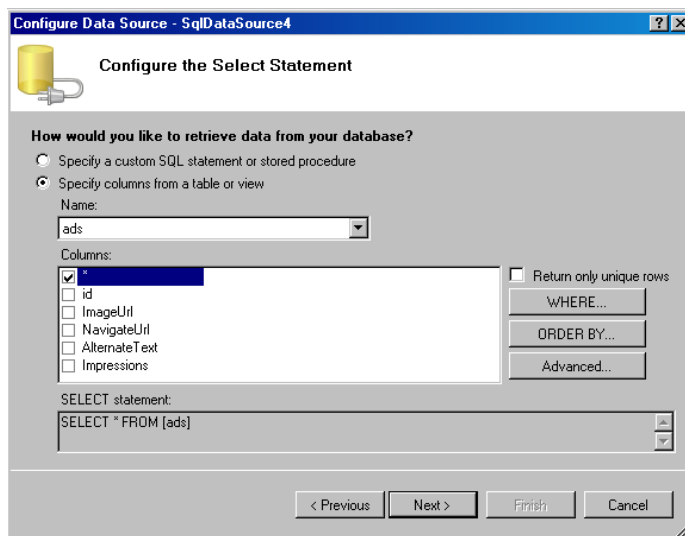


Figura 7.31 Selectarea tuturor înregistrărilor din tabela ads

Pentru a modifica interogarea sql, puteți alege opțiunea *Configure Data Source* a controlului *SqlDataSource* generat.

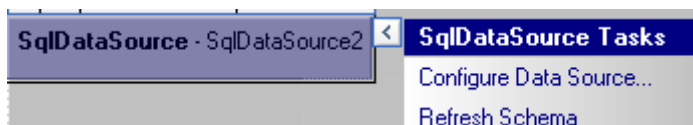


Figura 7.32 Modificarea proprietăților controlului *SqlDataSource*

Pentru caseta de informații „Top încasări” parcurgem aceeași pași. La pasul doi putem folosi *connection string* –ul generat anterior. Interogarea pentru extragerea interogărilor este:

```
SELECT TOP (5) nume, id FROM film
```

```
WHERE (data_aparitie < GETDATE()) ORDER BY incasari DESC
```

Sunt selectate primele 5 filme care au apărut înaintea datei curente, în ordinea descrescătoare a încasărilor.

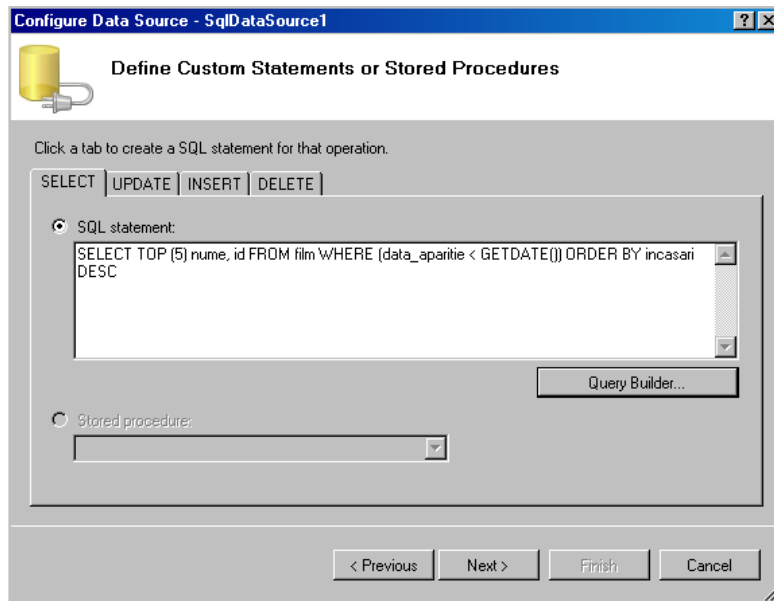


Figura 7.33 Interogarea Sql folosită pentru caseta de informații „Top încasări”

Câmpul *nume* va fi folosit pentru afișarea în controlul BulletedList, iar câmpul *id* va corespunde atributului value al controlului.

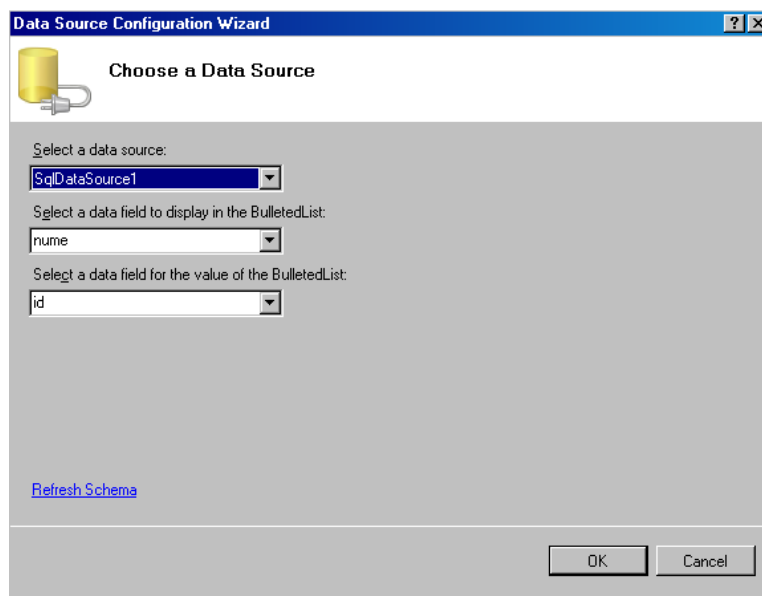


Figura 7.34 Selectarea câmpurilor nume și id pentru controlul BulletedList

Caseta de informații „În curând” trebuie să afișeze primele 5 filme care au data apariției mai mare decât data curentă. Pentru clauza Where putem utiliza funcția **GETDATE()** care returnează data curentă, la fel ca în cazul casetei „Top încasări”. Pentru a exemplifica însă diversele posibilități de construcție a interogărilor, vom folosi o altă abordare.

Interogarea de extragere a filmelor cu data apariției mai mare ca data curentă, va avea un parametru.

SELECT TOP 5 nume, id FROM film

WHERE (data_aparitie > @dataAparitie) ORDER BY data_aparitie

Numele parametrului este precedat de caracterul @.

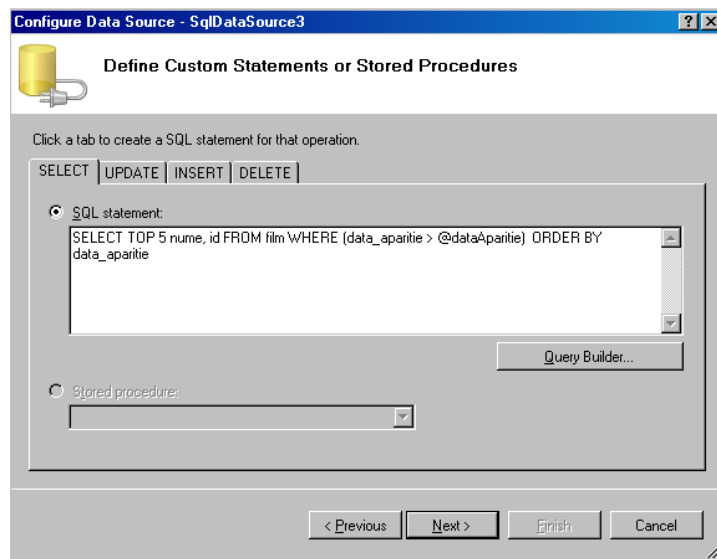


Figura 7.35 Interogarea parametrizată pentru selectarea filmelor cu data apariției mai mare ca data trimisă ca parametru.

La următorul pas vom selecta sursa de unde va fi preluată valoarea parametrului.

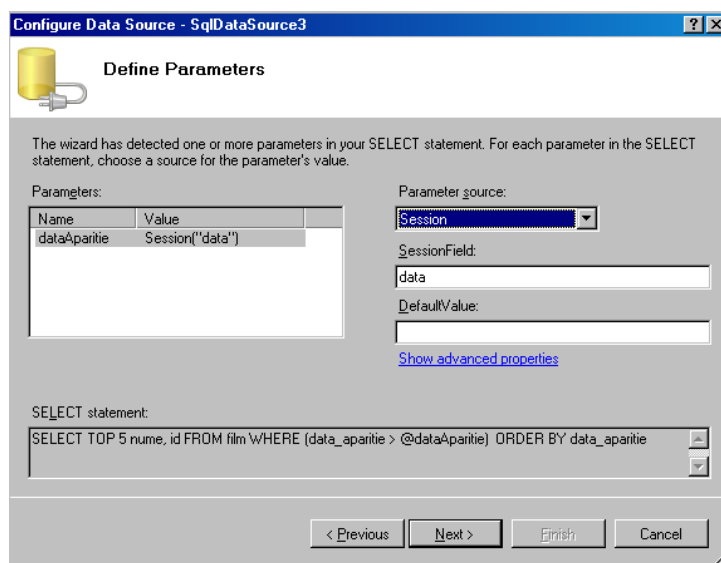


Figura 7.36 Parametrul interogării va fi preluat din sesiune, câmpul numit „data”

Valorile posibile pentru sursa parametrului sunt:

- Cookie
- Control (un alt control web din pagină, de exemplu un control textBox)
- Form
- Profile
- QueryString
- Session

Fișierul `MasterPage.master.cs` conține codul C# asociat paginii master. Pentru a reține în sesiune data curentă, folosim evenimentul `Page_Load`:

```
protected void Page_Load(object sender, EventArgs e)
{
    Session["data"] = DateTime.Now;
}
```

Pentru controalele `BulletedList` din cele două casete de informații, proprietatea `DisplayMode` are valoarea `LinkButton`. Dorim ca în momentul în care vizitatorul selectează un film din listă să-l redirectăm la pagina `Movie.aspx`, unde vor fi afișate informații despre film. Pentru aceasta vom trimite prin `QueryString` id-ul filmului selectat.

```
protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
{
    string x = BulletedList1.Items[e.Index].Value;
    Response.Redirect("movie.aspx?id=" + x);
}

protected void BulletedList2_Click(object sender, BulletedListEventArgs e)
{
    string x = BulletedList2.Items[e.Index].Value;
    Response.Redirect("movie.aspx?id=" + x);
}
```

VII.2. Home.aspx

Pagina `Home.aspx` prima pagină a proiectului și cea mai simplă ca structură. Conținând un mesaj de întâmpinare a utilizatorului.

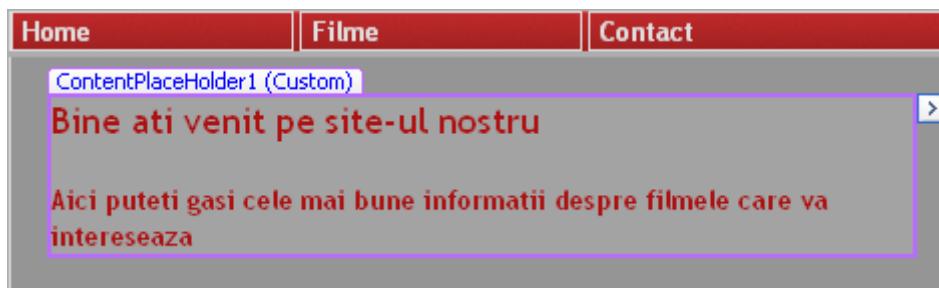


Figura 7.37 Pagina `Home.aspx` în modul design

Codul asp al paginii este:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Home.aspx.cs" Inherits="Home" Title="Movie
Info" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
<h3>Bine ati venit pe site-ul nostru</h3>
<p>Aici puteti gasi cele mai bune informatii despre filmele care va intereseaza</p>
</asp:Content>
```

VII.3. Movie.aspx

În această pagină sunt afișate filmele din baza de date și se poate realiza filtrarea acestora după anumite criterii.

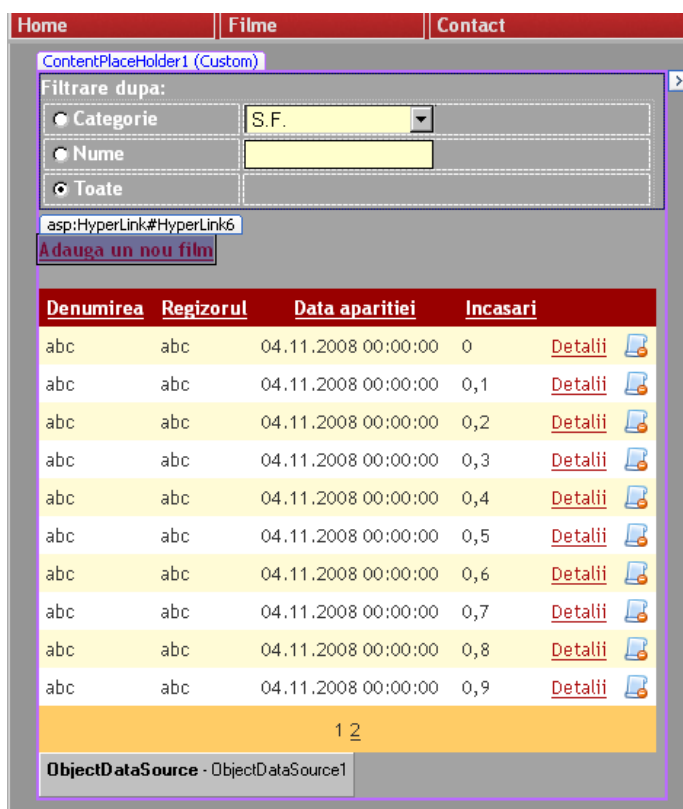


Figura 7.38 Pagina Movie.aspx în modul design

Afișarea filmelor se realizează prin intermediul unui control GridView.

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
AllowSorting="True" AutoGenerateColumns="False" CellPadding="4"
DataKeyNames="id" DataSourceID="ObjectDataSource1" GridLines="None"
ForeColor="#333333" Width="424px">
```

```

<FooterStyle BackColor="#990000" ForeColor="White" Font-Bold="True" />
<RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
<Columns>
<asp:BoundField DataField="nume" HeaderText="Denumirea" SortExpression="nume" />
<asp:BoundField DataField="regizor" HeaderText="Regizorul" SortExpression="regizor" />
<asp:BoundField DataField="data_aparitie" HeaderText="Data aparitiei"
SortExpression="data_aparitie"></asp:BoundField>
<asp:BoundField DataField="incasari" HeaderText="Incasari" SortExpression="incasari" />
<asp:HyperLinkField DataNavigateUrlFields="id"
DataNavigateUrlFormatString="Detalii.aspx?id={0}" Text="Detalii" />
<asp:CommandField ButtonType="Image" DeleteImageUrl="~/images/delete.png"
ShowDeleteButton="True" />
</Columns>
<PagerStyle BackColor="#FFCC66" ForeColor="#333333" HorizontalAlign="Center" />
<SelectedRowStyle BackColor="#FFCC66" Font-Bold="True" ForeColor="Navy" />
<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
<AlternatingRowStyle BackColor="White" />
</asp:GridView>

```

Sursa de date din GridView va fi un obiect DataSet. Adăugarea unui obiect DataSet se poate realiza prin intermediul opțiunii *Add New Item*.

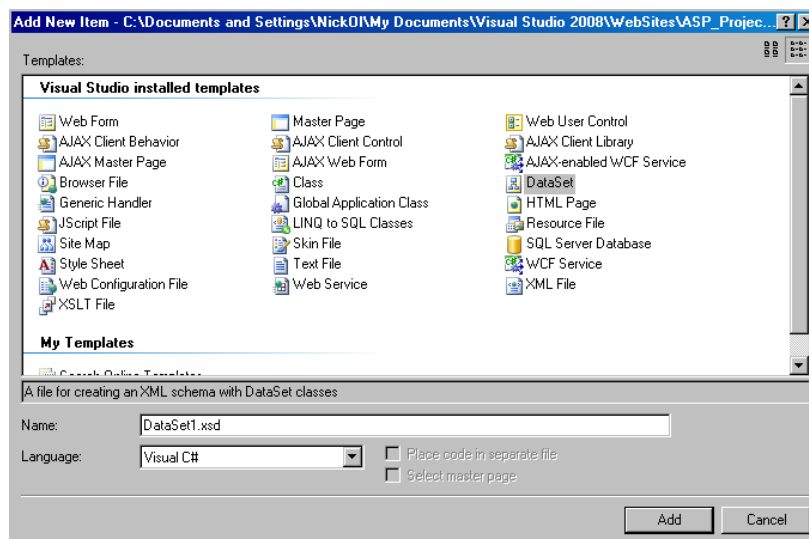


Figura 7.39 Adăugarea unui obiect DataSet

La crearea obiectului DataSet, în cadrul proiectului este adăugat un director nou, numit App_Code.

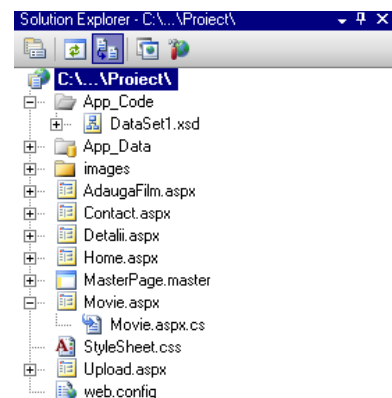


Figura 7.40 Directorul App_Code

Obiectul DataSet poate conține unul sau mai multe obiecte TableAdapter. Fiecare TableAdapter poate avea mai multe interogări care populează obiectul DataSet. Cea mai simplă metodă pentru a adăuga un obiect TableAdapter pentru o tabelă, este cu drag & drop din fereastra Database Explorer în fereastra DataSet. În cazul proiectului nostru, a fost creat un obiect TableAdapter corespunzător tabelii film.

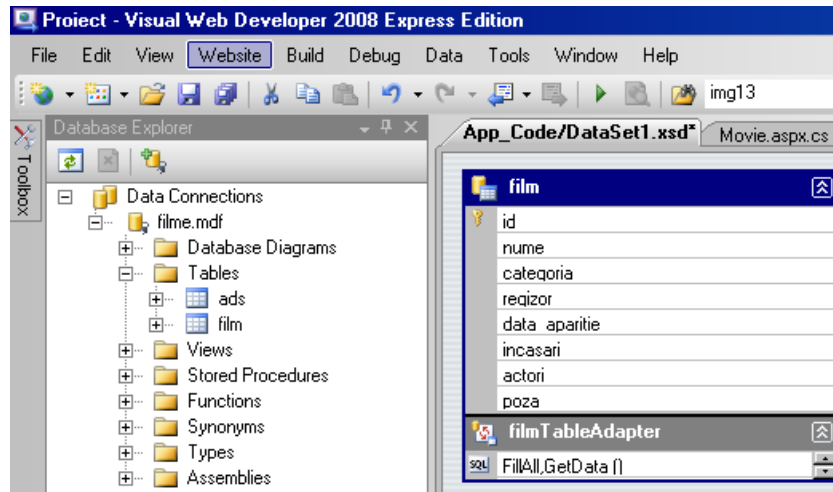


Figura 7.41 Adăugarea unui obiect TableAdapter prin Drag&Drop din fereastra Database Explorer

Implicit este creată metoda *FillAll*, care returnează toate înregistrările din tabelă. Pentru a vedea rezultatul interogării, se poate alege opțiunea *Preview Data*.

Adăugarea unei noi interogări este realizată cu ajutorul opțiunii *Add Query*, iar modificarea unei interogări existente cu ajutorul opțiunii *Configure*.

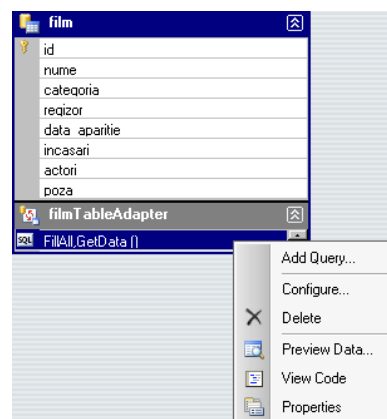


Figura 7.42 Opțiuni pentru prelucrarea interogărilor dintr-un obiect TableAdapter

Pentru legarea controlului GridView la sursa de date reprezentată de obiectul DataSet, se alege opțiunea *Object* la definirea tipului sursei de date.

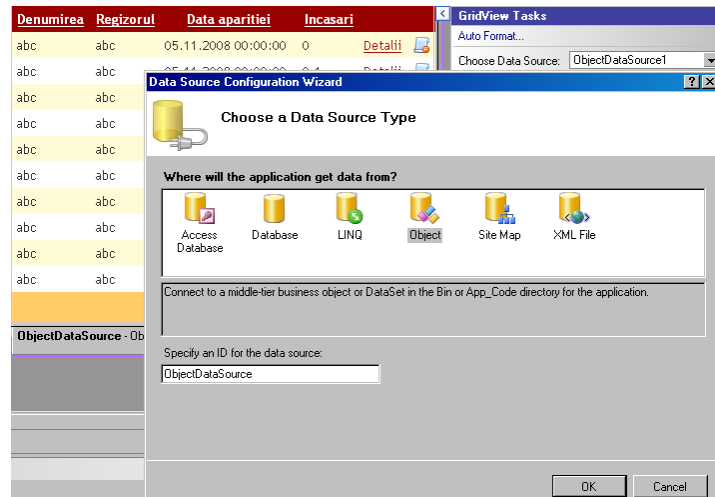


Figura 7.43 Tipul sursei de date pentru controlul GridView este Object

La următorul pas se alege obiectul TableAdapter dorit din DataSet.

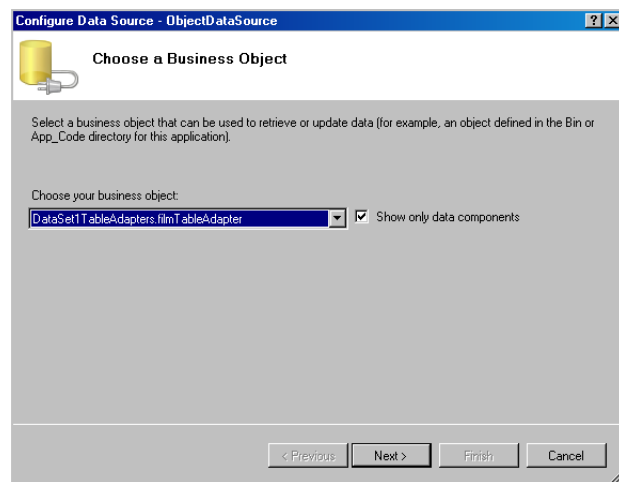


Figura 7.44 Selectarea obiectului TableAdapter

La ultimul pas se alege interogarea pentru popularea controlului GridView.

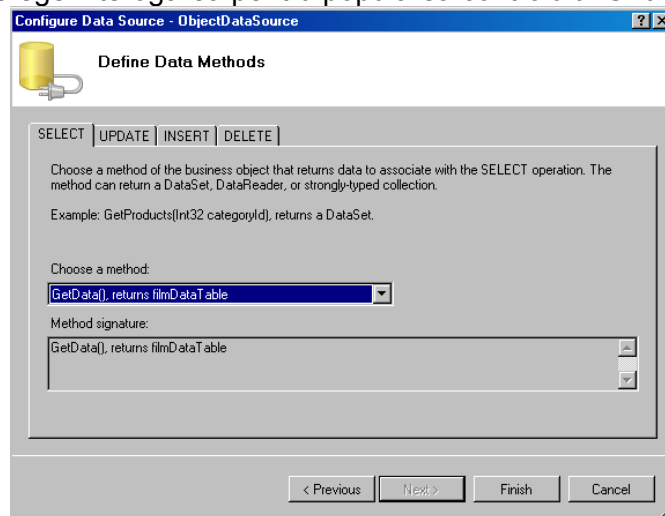


Figura 7.45 Selectarea metodei folosite pentru popularea obiectului DataSet

În lista de câmpuri a controlului GridView, vom adăuga un câmp de tip Hyperlink, numit Detalii. La selectarea acestui link pentru un anumit film, vom redirecta utilizatorul către pagina Detalii.aspx, trimițând prin QueryString id-ul filmului selectat. Pentru aceasta, proprietatea *DataNavigateUrlFields* a câmpului va avea valoarea *id*, iar proprietatea *DataNavigateUrlFormatString* va avea valoarea *Detalii.aspx?id={0}*. Parametrul dintre acolade va fi înlocuit automat cu valoarea id-ului filmului selectat.

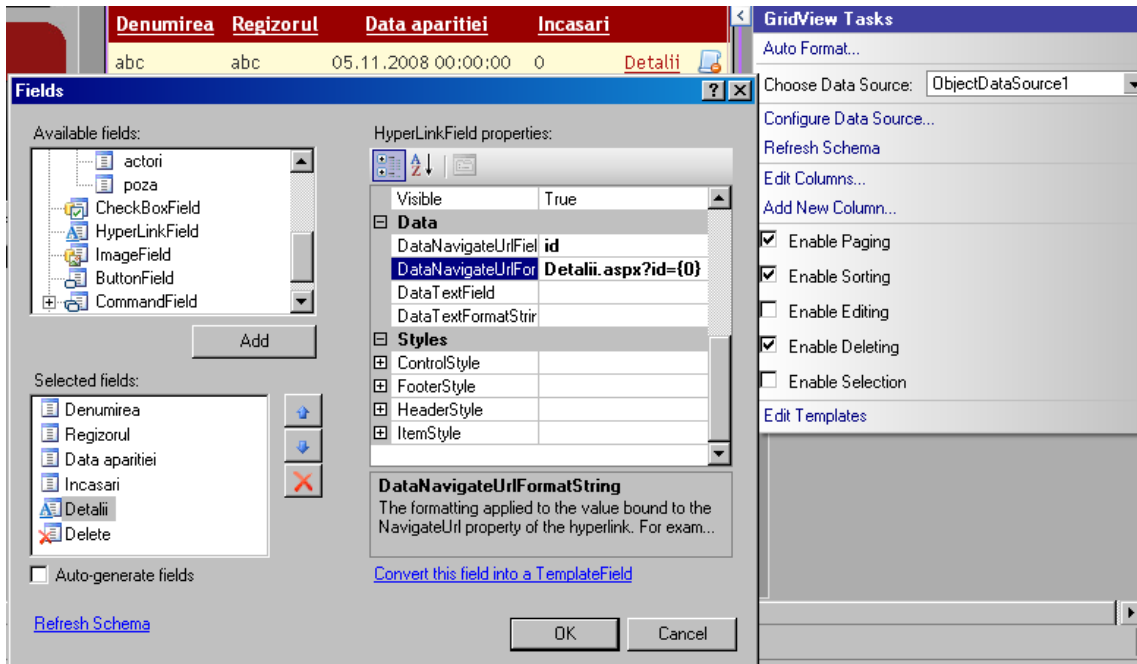


Figura 7.46 Modificarea proprietăților câmpului HyperLink Detalii

Filtrarea datelor din controlul GridView se realizează prin intermediul criteriilor definite de controalele de tip RadioButton. Există 3 modalități de filtrare: după categoria filmului, după numele acestuia sau afișarea tuturor filmelor. Categoria filmului căutat poate fi selectată printr-un control DropDownList.

```
<asp:Panel ID="Panel1" runat="server" BorderStyle="Solid" BorderWidth="1px">
<asp:Label ID="Label1" runat="server" Font-Bold="True" ForeColor="White"
Text="Filtrare dupa:"></asp:Label><br />
<table>
<tr>
<td>
<asp:RadioButton ID="RadioButton1" runat="server" AutoPostBack="True"
Font-Bold="True" ForeColor="White" GroupName="filtru" oncheckedchanged =
"RadioButton1_CheckedChanged" Text="Categorie" />
</td>
<td>
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
CssClass="textBox" Enabled="False"
onselectedindexchanged="DropDownList1_SelectedIndexChanged"
TabIndex="6" Width="131px">
```

```

        <asp:ListItem>S.F.</asp:ListItem>
        <asp:ListItem>Romance</asp:ListItem>
        <asp:ListItem>Drama</asp:ListItem>
        <asp:ListItem>Razboi</asp:ListItem>
        <asp:ListItem>Actiune</asp:ListItem>
        <asp:ListItem>Aventura</asp:ListItem>
        <asp:ListItem>Desene animate</asp:ListItem>
    </asp:DropDownList>
</td>
</tr>
<tr>
<td class="style2">
    <asp:RadioButton ID="RadioButton2" runat="server" AutoPostBack="True"
        Font-Bold="True" ForeColor="White" GroupName="filtru"
        oncheckedchanged = "RadioButton2_CheckedChanged" Text="Nume" />
</td>
<td>
    <asp:TextBox ID="TextBox1" runat="server" CssClass="textBox"
        Enabled="False" Width="130px" AutoPostBack="True"
        ontextchanged="TextBox1_TextChanged"></asp:TextBox>
</td>
</tr>
<tr>
<td class="style2">
    <asp:RadioButton ID="RadioButton3" runat="server" AutoPostBack="True"
        Checked="True" Font-Bold="True" ForeColor="White" GroupName="filtru"
        Text="Toate" oncheckedchanged = "RadioButton3_CheckedChanged" />
</td>
<td>&nbsp;</td>
</tr>
</table>
</asp:Panel>

```

Pentru a putea filtra datele din GridView, vom adauga obiectului TableAdapter 3 metode, ce conțin interogări sql parametrizate. Metodele se vor numi GetDataById(), GetDataByCategory(), GetDataByName().

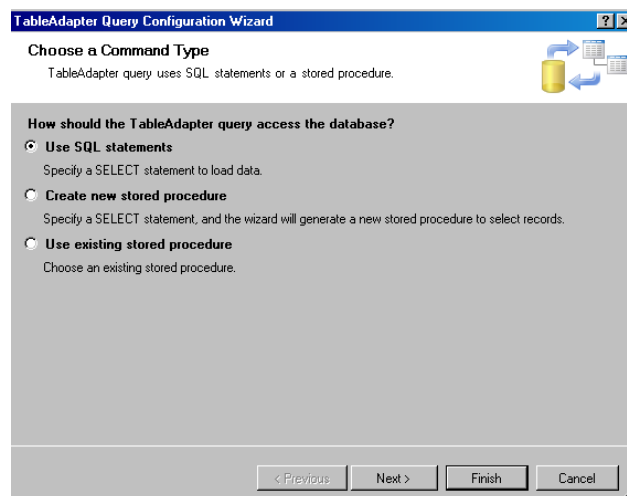


Figura 7.47 Pentru extragerea datelor se folosește o nouă interogare

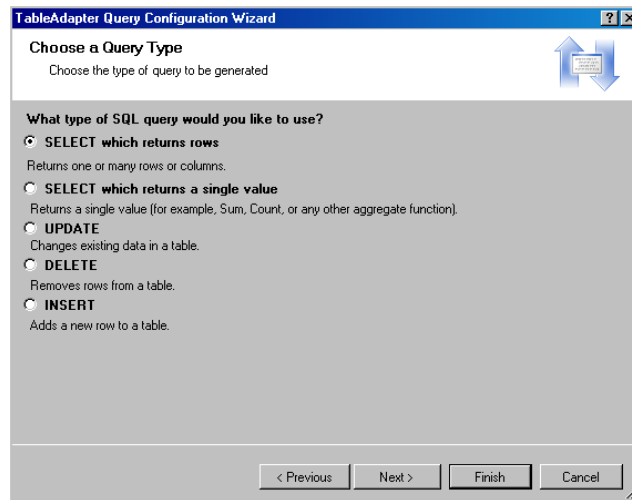


Figura 7.48 Alegerea tipului interogării

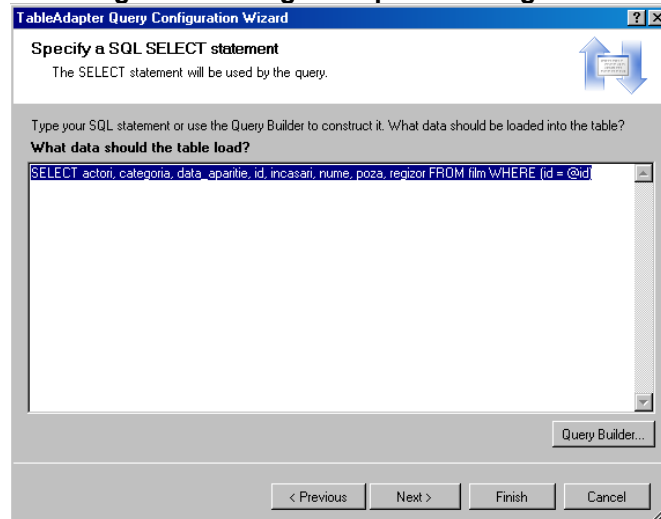


Figura 7.49 Construirea interogării

Interogarea va returna acea înregistrare din tabela film, care are valoarea câmpului id egală cu valoarea trimisă ca parametru:

```
SELECT actori, categoria, data_aparitie, id, incasari, nume, poza, regizor
FROM film WHERE (id = @id)
```

La ultimul pas vom alege numele metodei.

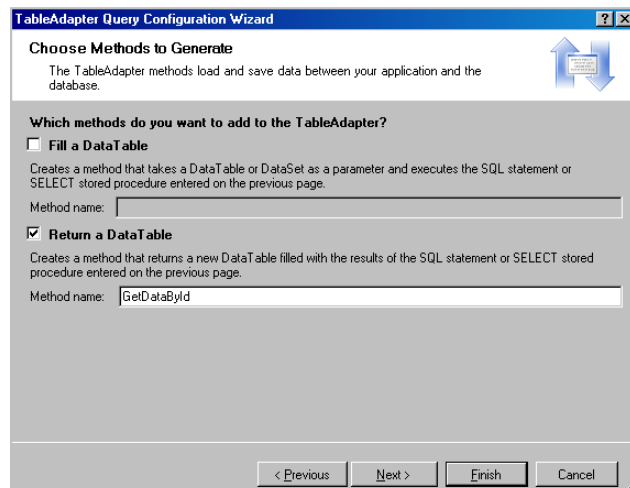


Figura 7.50 Introducerea denumirii metodei

Pentru metoda `GetDataByCategory()` interogarea sql este:

```
SELECT id, nume, data_aparitie, incasari, regizor, actori, poza, categoria  
FROM film WHERE (categoria = @categoria)
```

Interogarea conținută de metoda `GetDataByName()` este:

```
SELECT id, nume, data_aparitie, incasari, regizor, actori, poza, categoria  
FROM film WHERE (nume LIKE '%' + @nume + '%')
```

Metoda `GetDataById()` va fi apelată atunci când utilizatorul ajunge în pagina `movie.aspx` din casetele de informații „Top încasări”, respectiv „În curând” din `MasterPage`. Când utilizatorul selectează un film în caseta de informații, se realizează un redirect către pagina `movie.aspx`, id-ul filmului selectat fiind trimis folosind `QueryString`. În pagina `movie.aspx` evenimentul `Page_Load` vom prelua acest id și om apela modifica interogarea folosită pentru popularea controlului `GridView`:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.QueryString["id"] != null)
    {
        string id = Request.QueryString["id"];
        ObjectDataSource1.SelectMethod = "GetDataById";
        ObjectDataSource1.SelectParameters.Clear();
        ObjectDataSource1.SelectParameters.Add(new Parameter("id", TypeCode.Int32, id));
    }
}
```

Același procedeu îl vom folosi și pentru filtrarea după criteriile nume și categorie:

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    ObjectDataSource1.SelectMethod = "GetDataByCategory";
    ObjectDataSource1.SelectParameters.Clear();
    ObjectDataSource1.SelectParameters.Add(new Parameter("categoria",
        TypeCode.String, DropDownList1.SelectedValue));
}
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    if (TextBox1.Text.Length > 0)
    {
        ObjectDataSource1.SelectMethod = "GetDataByName";
        ObjectDataSource1.SelectParameters.Clear();
        ObjectDataSource1.SelectParameters.Add(new Parameter("nume",
            TypeCode.String, TextBox1.Text));
    }
}
```

Evenimentul `SelectedIndexChanged` va fi generat atunci când utilizatorul selectează o categorie în controlul de tip `DropDownList`, iar evenimentul `TextChanged` va fi generat când utilizatorul apasă tasta `Enter` în controlul `TextBox`.

Pentru afișarea tuturor înregistrărilor în GridView, se folosește un control de tip RadioButton.

```
protected void RadioButton3_CheckedChanged(object sender, EventArgs e)
{
    TextBox1.Text = "";
    TextBox1.Enabled = false;
    DropDownList1.Enabled = false;

    ObjectDataSource1.SelectMethod = "GetData";
    ObjectDataSource1.SelectParameters.Clear();
}
```

Pentru ca la selectarea unui criteriu de filtrare să nu rămână activ criteriul precedent, vom utiliza proprietatea *Enabled* a controalelor DropDownList, respectiv TextBox:

```
protected void RadioButton1_CheckedChanged(object sender, EventArgs e)
{
    TextBox1.Text = "";
    TextBox1.Enabled = false;
    DropDownList1.Enabled = true;
}

protected void RadioButton2_CheckedChanged(object sender, EventArgs e)
{
    TextBox1.Enabled = true;
    DropDownList1.Enabled = false;
    TextBox1.Focus();
}
```

În pagina movie.aspx am adăugat și un control de tip hyperlink pentru adăugarea unui nou film în baza de date.

```
<asp:HyperLink ID="HyperLink6" runat="server" Font-Bold="True"
    NavigateUrl="~/AdaugaFilm.aspx">Adauga un nou film </asp:HyperLink>
```

VII.4. Detalii.aspx

În această pagină putem modifica informațiile asociate unui film, modificările putând fi salvate în baza de date.

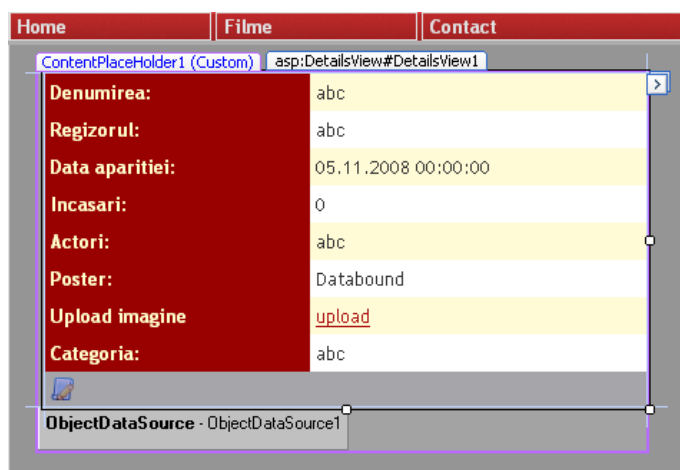


Figura 7.51 Detalii.aspx în modul design

Pentru afișarea și modificarea informațiilor pentru un film, vom folosi un control de tip DetailsView

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
    CellPadding="4" DataKeyNames="id" DataSourceID="ObjectDataSource1"
    ForeColor="#333333" GridLines="None" Height="50px"
    HorizontalAlign="Center" Width="420px">
<FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
    CommandRowStyle BackColor="#FFFC0" Font-Bold="True" />
<RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
<FieldHeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="#FFFC0" />
<PagerStyle BackColor="#FFCC66" ForeColor="#333333" HorizontalAlign="Center" />
<Fields>
<asp:BoundField DataField="nume" HeaderText="Denumirea:" SortExpression="nume" />
<asp:BoundField DataField="regizor" HeaderText="Regizorul:" SortExpression="regizor"/>
<asp:BoundField DataField="data_aparitie" HeaderText="Data aparitiei:"
    SortExpression="data_aparitie" />
<asp:BoundField DataField="incasari" HeaderText="Incasari:" SortExpression="incasari"/>
<asp:BoundField DataField="actori" HeaderText="Actori:" SortExpression="actori" />
<asp:ImageField DataImageUrlField="poza"
    DataImageUrlFormatString="images/movies/{0}" HeaderText="Poster:">
</asp:ImageField>
<asp:HyperLinkField HeaderText="Upload imagine" Text="upload"
    DataNavigateUrlFields="id" DataNavigateUrlFormatString="Upload.aspx?id={0}" />
<asp:BoundField DataField="categoria" HeaderText="Categoria:" SortExpression="categoria"/>
<asp:CommandField ButtonType="Image" CancelImageUrl="~/images/delete.png"
    EditImageUrl="~/images/edit.png" ShowEditButton="True"
    UpdateImageUrl="~/images/edit.png" />
</Fields>
<HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
<AlternatingRowStyle BackColor="White" />
</asp:DetailsView>
```

Sursa de date pentru acest control va fi tot obiectul DataSet. Metoda care va popula controlul DetailsView va fi GetDataById(). Valoarea parametrului @id din interogare, este preluat prin QueryString, deoarece in această pagină se ajunge din movie.aspx prin selectarea link-ului *Detalii* din controlul GridView. Pașii pentru selectarea sursei de date sunt:

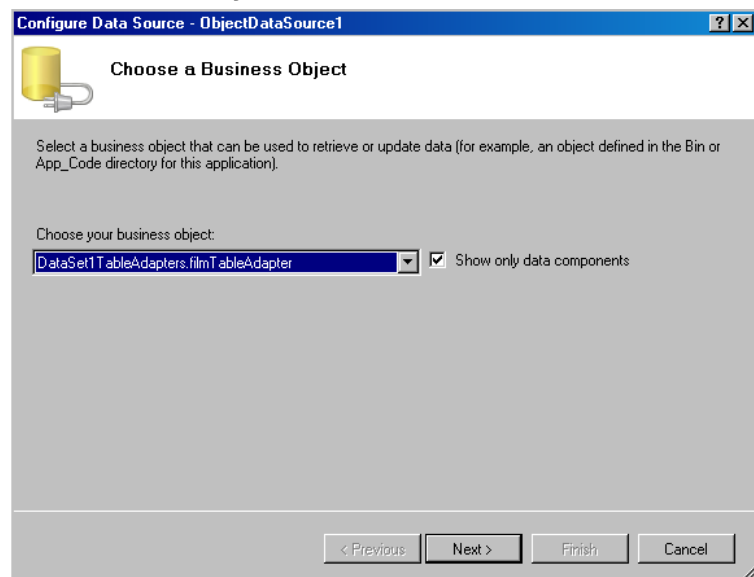


Figura 7.52 Selectarea obiectului DataSet ca sursă de date

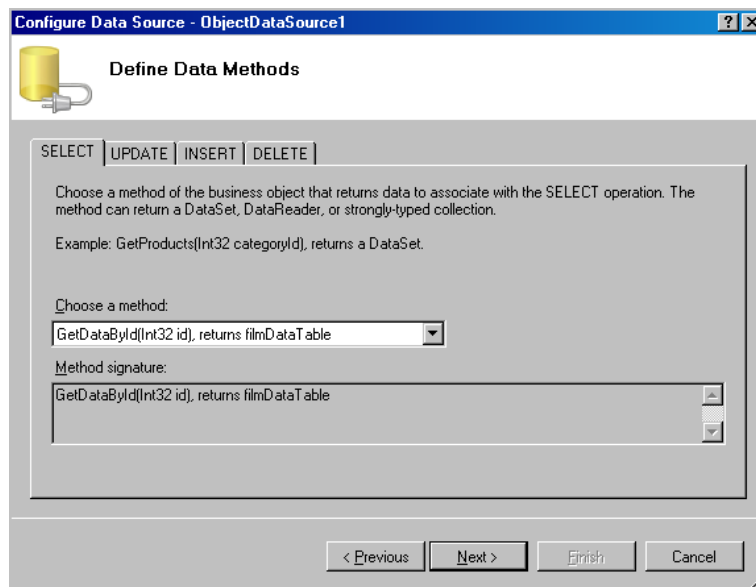


Figura 7.53 Selectarea metodei GetDataById

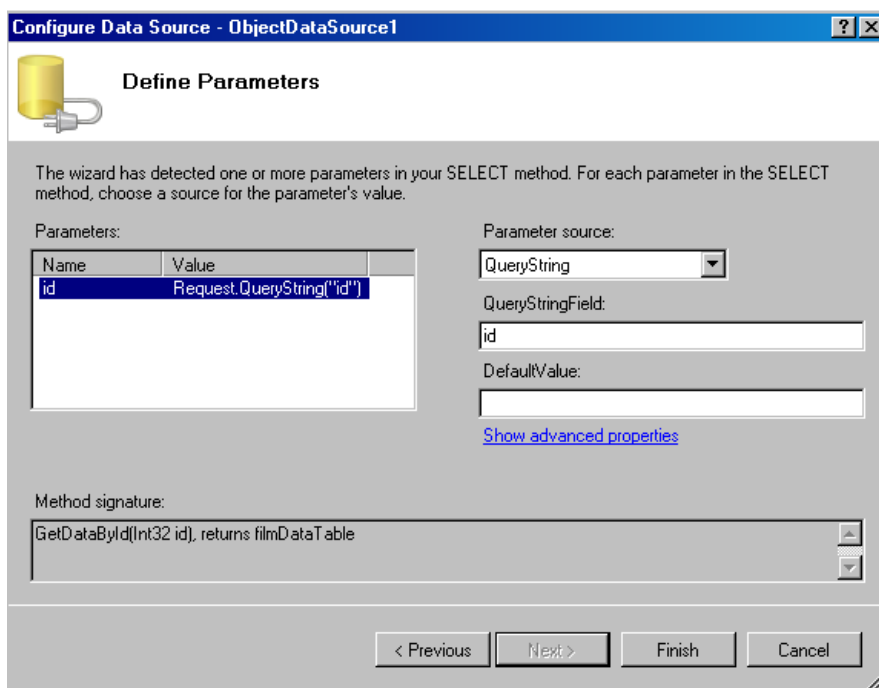


Figura 7.54 Valoarea parametrului id este preluată din QueryString

În lista de câmpuri ale controlului DetailsView, vrem să apară un link către pagina care permite upload-ul unui fișier, respectiv un câmp de tip imagine, care să afișeze imaginea upload-ată. Imaginea reprezintă posterul filmului, iar câmpul *poza* din tabela *film* conține numele fișierului imagine. Toate posterele sunt upload-ate în directorul images/movies.

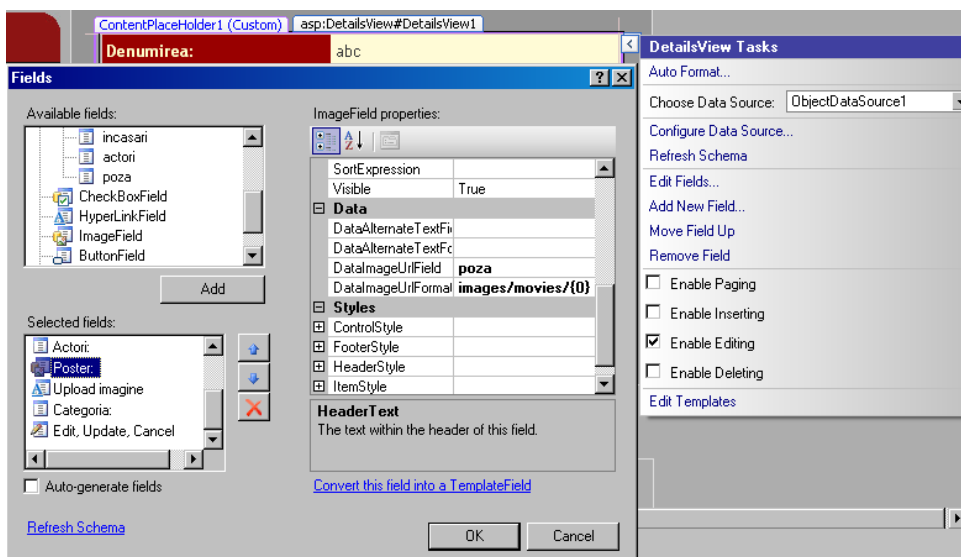


Figura 7.55 Modificarea proprietăților câmpului ImageField

În controlul DetailsView adăugăm un nou câmp de tip ImageField. Pentru a afișa posterul, modificăm proprietățile :

- *DataImageUrlField* are valoarea numelui câmpului din baza de date care conține numele fișierului imagine: *poza*
- *DataImageUrlFormatString* conține un șir de caractere care reprezintă calea către imagine: *images/movies/{0}*. Parametrul dintre acolade va fi înlocuit de numele fișierului imagine din câmpul bazei de date

Pentru a afișa un link pentru upload-ul unui fișier imagine, controlul DetailsView conține o coloană de tip *HyperLinkField*. Pentru acest câmp modificăm proprietățile:

- *DataNavigateUrlFields* conține id-ul care va fi trimis prin QueryString
- *DataImageUrlFormatString* conține un șir de caractere care reprezintă adresa paginii către care se realizează redirect-ul: **Upload.aspx?id={0}**. Parametrul dintre acolade va fi înlocuit cu id-ul înregistrării modificate.

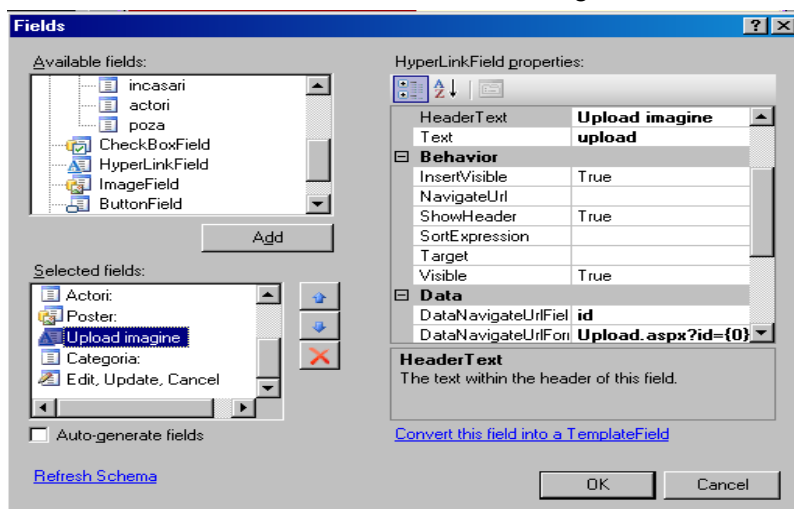


Figura 7.56 Modificarea proprietăților câmpului de tip HyperLinkField

VII.5. Upload.aspx

În această pagină utilizatorul poate upload-a un fișier, acesta fiind salvat în directorul **/images/movies/**. Controlul de tip FileUpload are asociat un control de validare de tip RequiredFieldValidator. De asemenea, pentru a afișa erorile vom utiliza un control de tip ValidationSummary.

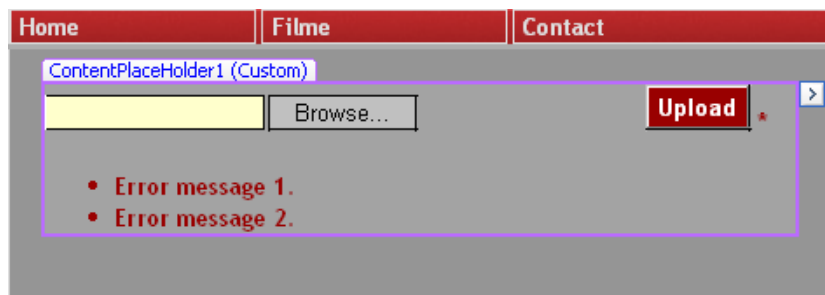


Figura 7.57 Pagina upload.aspx în design view

Codul asp asociat paginii este:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Upload.aspx.cs" Inherits="Upload" Title="Upload Poster"%>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:ContentID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"Runat="Server">
<asp:FileUpload ID="FileUpload1" runat="server" Width="340px" CssClass="textBox" />
&nbsp;
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Upload"
Width="60px" CssClass="button" />

&nbsp;
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="FileUpload1" ErrorMessage="Nu ati selectat un fisier pentru upload!"
Font-Bold="True" ForeColor="#990000">*
</asp:RequiredFieldValidator>
<br />
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
Font-Bold="True" ForeColor="#990000" />
</asp:Content>
```

Utilizatorul poate ajunge în pagina `upload.aspx` fie din pagina de adăugare a unui nou film (`AdaugăFilm.aspx`), fie din pagina de detalii (`Detalii.aspx`). Deoarece în pagina de detalii posterul filmului este afișat printr-un câmp `ImageField` al controlului `DetailsView`, după upload-ul fișierului pe server se salvează numele fișierului uploadat în tabela `film` din baza de date filme.

Pentru conectarea la baza de date se folosește șirul de conectare salvat în fișierul `web.config`. Astfel, în această pagină trebuie inclus spațiul de nume `System.Web.Configuration`:

```
using System.Web.Configuration;
```

Dacă utilizatorul ajunge în pagina `Upload.aspx` din pagina `AdaugăFilm.aspx`, numele fișierului uploadat va fi salvat în sesiune.

Codul executat la apăsarea butonului *upload* este următorul:

```

protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    { FileUpload1.SaveAs(Server.MapPath(".") + "/images/movies/" + FileUpload1.FileName);
      int id = Convert.ToInt32(Request.QueryString["id"]);
      if (id > 0)
      { SqlConnection conn = new SqlConnection
        (WebConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();
        string updateString = @"UPDATE film SET poza = @fileName WHERE id = @id";
        SqlCommand cmd = new SqlCommand(updateString, conn);
        cmd.Parameters.AddWithValue("@id", id.ToString());
        cmd.Parameters.AddWithValue("@fileName", FileUpload1.FileName);
        cmd.ExecuteNonQuery();
        Server.Transfer("Detalii.aspx?id=" + id);
      }
    else
    { Session["uploadedFileName"] = FileUpload1.FileName;
      Server.Transfer("AaugăFilm.aspx");
    }
  }
}

```

VII.6. AaugăFilm.aspx

Această pagină este un formular de adăugare a datelor introduse de utilizator în tabela film a bazei de date.

Toate controalele de tip `TextBox` sau `DropDownList` au asociată clasa css `textBox` prin intermediul proprietății `CssClass`.

Figura 7.58 AaugăFilm.aspx în modul design

Deoarece toate câmpurile din baza de date sunt obligatorii, fiecărui control pentru introducerea datelor (cu excepția celui pentru introducerea încasărilor) i s-a asociat un control de tip `RequiredFieldValidator`. Controlul `txtÎncasări` are asociat un control de tip `RangeValidator`.

The screenshot shows a web form titled 'ContentPlaceHolder1 (Custom)'. It features a dark red header and several input fields with a light yellow background. The fields are: 'Denumirea:' (text box), 'Regizorul:' (text box), 'Actori:' (text area), 'Data aparitiei:' (calendar for 'noiembrie 2008'), 'Incasari:' (text box with value '0'), 'Categoria:' (dropdown menu with 'S.F.' selected), and 'Poster:' (text box with an 'Upload' button). At the bottom, there is a red button labeled 'Aauga'. Below the button, there are two error messages: 'Error message 1.' and 'Error message 2.'


```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<table cellpadding="5" cellspacing="1" class="style1">
<tr>
<td bgcolor="#990000">
<asp:Label ID="Label1" runat="server" Font-Bold="True" ForeColor="White"
Text="Denumirea:"></asp:Label>
</td>
<td>
<asp:TextBox ID="txtDenumire" runat="server" Width="213px" TabIndex="1"
BorderStyle="Groove" CssClass="textBox" MaxLength="50"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="txtDenumire" ErrorMessage="Nu ati introdus denumirea"
Font-Bold="True" ForeColor="#990000">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td bgcolor="#990000">
<asp:Label ID="Label2" runat="server" Font-Bold="True" ForeColor="White"
Text="Regizorul:"></asp:Label>
</td>
<td>
<asp:TextBox ID="txtRegizor" runat="server" Width="214px" TabIndex="2"
BorderStyle="Groove" CssClass="textBox" MaxLength="50"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="txtRegizor" ErrorMessage="Nu ati introdus regizorul"
Font-Bold="True" ForeColor="#990000">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td bgcolor="#990000">
<asp:Label ID="Label3" runat="server" Font-Bold="True" ForeColor="White"
Text="Actori:"></asp:Label>
</td>
<td>
<asp:TextBox ID="txtActori" runat="server" Rows="5" TextMode="MultiLine"
Width="212px" TabIndex="3" BorderStyle="Groove" CssClass="textBox"
MaxLength="50"></asp:TextBox>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="txtActori" ErrorMessage="Nu ati introdus actorii"
Font-Bold="True" ForeColor="#990000">*</asp:RequiredFieldValidator>
</td>
</tr>
</table>
```

```


<tr>
<td bgcolor="#990000">
<asp:Label ID="Label4" runat="server" Font-Bold="True" ForeColor="White"
Text="Data aparitiei:"></asp:Label>
</td>
<td>
<asp:TextBox ID="txtData" runat="server" Width="100px" TabIndex="4"
BorderStyle="Groove" CssClass="textBox"></asp:TextBox>
<asp:ImageButton ID="ImageButton1" runat="server"
ImageUrl="~/images/calendar.gif" onclick="ImageButton1_Click"
ToolTip="Calendar" CausesValidation="False" />
<asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC"
BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest"
Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399" Height="200px"
ShowGridLines="True" Visible="False" Width="220px"
onselectionchanged="Calendar1_SelectionChanged">
<SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
<SelectorStyle BackColor="#FFCC66" />
<TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
<OtherMonthDayStyle ForeColor="#CC9966" />
<NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
<DayHeaderStyle BackColor="#FFCC66"Font-Bold="True"Height="1px" />
<TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
ForeColor="#FFFFCC" />
</asp:Calendar>
</td>
<td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
ControlToValidate="txtData" ErrorMessage="Nu ati introdus data aparitiei"
Font-Bold="True" ForeColor="#990000">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td bgcolor="#990000">
<asp:Label ID="Label7" runat="server" Font-Bold="True" ForeColor="White"
Text="Incasari:"></asp:Label>
</td>
<td>
<asp:TextBox ID="txtIncasari" runat="server" TabIndex="5" BorderStyle="Groove"
CssClass="textBox">0</asp:TextBox>
</td>
<td>
<asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="txtIncasari" ErrorMessage="Incasarile intre 0 si 1000000 $"
Font-Bold="True" ForeColor="#990000" MaximumValue="1000000"
MinimumValue="0">*</asp:RangeValidator>
</td>
</tr>
<tr>
<td bgcolor="#990000" class="style5">
<asp:Label ID="Label5" runat="server" Font-Bold="True" ForeColor="White"
Text="Categoria:"></asp:Label>
</td>
<td>
<asp:DropDownList ID="DropDownList1" runat="server" TabIndex="6"
CssClass="textBox">

```

```

        <asp:ListItem>S.F.</asp:ListItem>
        <asp:ListItem>Romance</asp:ListItem>
        <asp:ListItem>Drama</asp:ListItem>
        <asp:ListItem>Razboi</asp:ListItem>
        <asp:ListItem>Actiune</asp:ListItem>
        <asp:ListItem>Aventura</asp:ListItem>
        <asp:ListItem>Desene animate</asp:ListItem>
    </asp:DropDownList>
</td>
<td>&nbsp;</td>
</tr>
<tr>
<td bgcolor="#990000">
    <asp:Label ID="Label6" runat="server" Font-Bold="True" ForeColor="White"
        Text="Poster:"></asp:Label>
</td>
<td >
    <asp:TextBox ID="txtPoster" runat="server" CssClass="textBox" Height="22px"
        MaxLength="50" Width="173px"></asp:TextBox>
    <asp:LinkButton ID="LinkButton1" runat="server" CausesValidation="False"
        onclick="LinkButton1_Click">Upload </asp:LinkButton>
</td>
<td>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server"
        ControlToValidate="txtPoster" ErrorMessage="Nu ati uploadat posterul"
        ForeColor="#990000">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td align="center" colspan="3">
    <asp:Button ID="Button1" runat="server" Text="Adauga" CssClass="button"
        onclick="Button1_Click" />
</td>
</tr>
<tr>
<td colspan="3">
    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
        Font-Bold="True" ForeColor="#990000" />
</td>
</tr>
</table>
</asp:Content>

```

Afișarea sau ascunderea calendarului se realizează prin intermediul unui control de tip ImageButton .

```

protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    if (Calendar1.Visible == true)
    {Calendar1.Visible = false;}
    else
    {Calendar1.Visible = true;}
}

```

La apăsarea acestui buton se generează un PostBack, rezultând în validarea controalelor din pagină, deși nu s-a apăsât încă butonul Adaugă. Pentru a nu se realiza această validare, controlul ImageButton1 are setată proprietatea *CausesValidation* la valoarea *False*.

Când utilizatorul selectează o dată calendaristică, se completează controlul textBox corespunzător:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    txtData.Text = Calendar1.SelectedDate.ToShortDateString();
}
```

Link-ul pentru upload-ul unui fișier este un control de tip LinkButton. El va redirecta către pagina upload.aspx, unde utilizatorul va putea upload-a un fișier. Ca și la controlul ImageButton, proprietatea *CausesValidation* are valoarea *False*. În momentul în care utilizatorul se întoarce în pagina AdaugăFișier.aspx după ce a realizat upload-ul, datele introduse până atunci se pierd, deoarece server-ul reinițializează pagina. Pentru a nu pierde datele introduse, acestea vor fi salvate în sesiune la apăsarea butonului upload.

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
    if (txtDenumire.Text.Length > 0)
    {
        Session["denumire"] = txtDenumire.Text;
    }
    if (txtRegizor.Text.Length > 0)
    {
        Session["regizor"] = txtRegizor.Text;
    }
    if (txtActori.Text.Length > 0)
    {
        Session["actori"] = txtActori.Text;
    }
    if (txtData.Text.Length > 0)
    {
        Session["dataAparitiei"] = txtData.Text;
    }
    if (txtIncasari.Text.Length > 0)
    {
        Session["incasari"] = txtIncasari.Text;
    }
    Session["categoria"] = DropDownList1.SelectedIndex;

    Server.Transfer("Upload.aspx?id=0");
}
```

În evenimentul Page_Load al paginii vom verifica dacă sunt date în sesiune, și vom inițializa controalele din pagină.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        txtDenumire.Focus();
    }
    if (Session["denumire"] != null)
    {
        txtDenumire.Text = Session["denumire"].ToString();
    }
    if (Session["regizor"] != null)
    {
        txtRegizor.Text = Session["regizor"].ToString();
    }
    if (Session["actori"] != null)
    {
        txtActori.Text = Session["actori"].ToString();
    }
    if (Session["dataAparitiei"] != null)
    {
        txtData.Text = Session["dataAparitiei"].ToString();
    }
    if (Session["incasari"] != null)
    {
        txtIncasari.Text = Session["incasari"].ToString();
    }
    if (Session["uploadedFileName"] != null)
    {
        txtPoster.Text = Session["uploadedFileName"].ToString();
    }
    if (Session["categoria"] != null)
    {
        DropDownList1.SelectedIndex = Convert.ToInt32(Session["categoria"]);
    }
}
```

La apăsarea butonului Adaugă, datele vor fi salvate în baza de date și șterse din sesiune:

```
protected void Button1_Click(object sender, EventArgs e)
{
    SqlConnection conn = new SqlConnection
    (WebConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
    conn.Open();
    string insertString = "INSERT INTO film (nume, data_aparitie, incasari, regizor,
    actori, poza, categoria)";
    insertString += " VALUES (@nume, @data, @incasari, @regizor, @actori, @poza,
    @categoria)";
    SqlCommand cmd = new SqlCommand(insertString, conn);
    cmd.Parameters.AddWithValue("@nume", txtDenumire.Text.Trim());
    cmd.Parameters.AddWithValue("@data", Convert.ToDateTime(txtData.Text));
    cmd.Parameters.AddWithValue("@incasari", Convert.ToInt32(txtIncasari.Text.Trim()));
    cmd.Parameters.AddWithValue("@regizor", txtRegizor.Text.Trim());
    cmd.Parameters.AddWithValue("@actori", txtActori.Text.Trim());
}
```

```

cmd.Parameters.AddWithValue("@poza", txtPoster.Text.Trim());
cmd.Parameters.AddWithValue("@categoria", DropDownList1.SelectedValue);
cmd.ExecuteNonQuery();

    Session.Remove("denumire");
    Session.Remove("regizor");
    Session.Remove("actori");
    Session.Remove("dataAparitiei");
    Session.Remove("incasari");
    Session.Remove("denumire");
    Session.Remove("categoria");
    Session.Remove("uploadedFileName");

    Server.Transfer("Movie.aspx");
}

```

VII.7. Contact.aspx

Datele introduse de utilizator în acest formular, vor fi trimise prin email către deținătorul paginii. Pentru realizarea formularului, s-a folosit un control de tip Wizard, care conține trei pași.

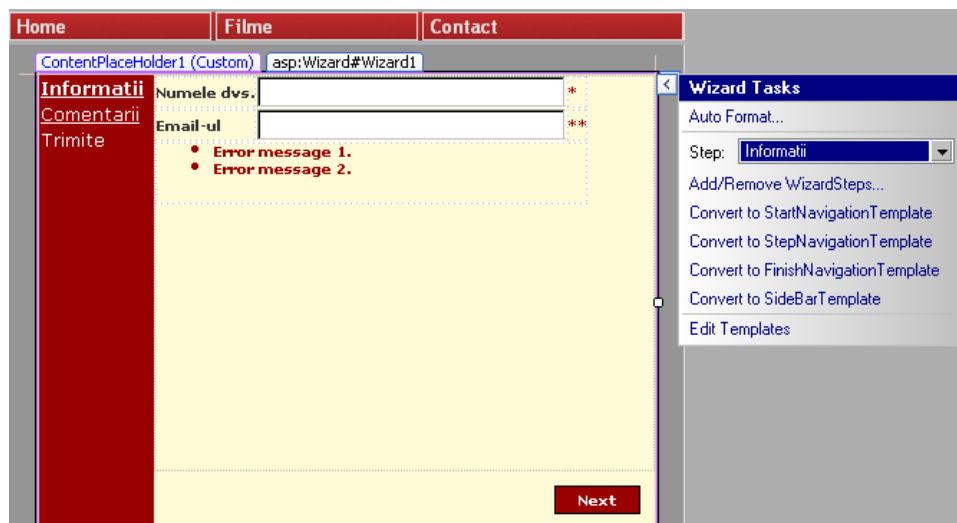


Figura 7.59
Pasul 1 de
completare al
formularului de
contact

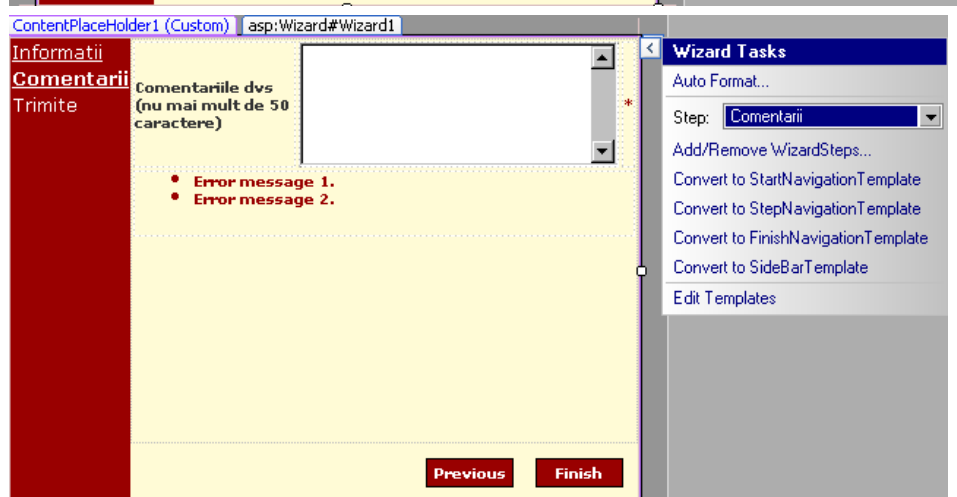


Figura 7.60 Pasul 2

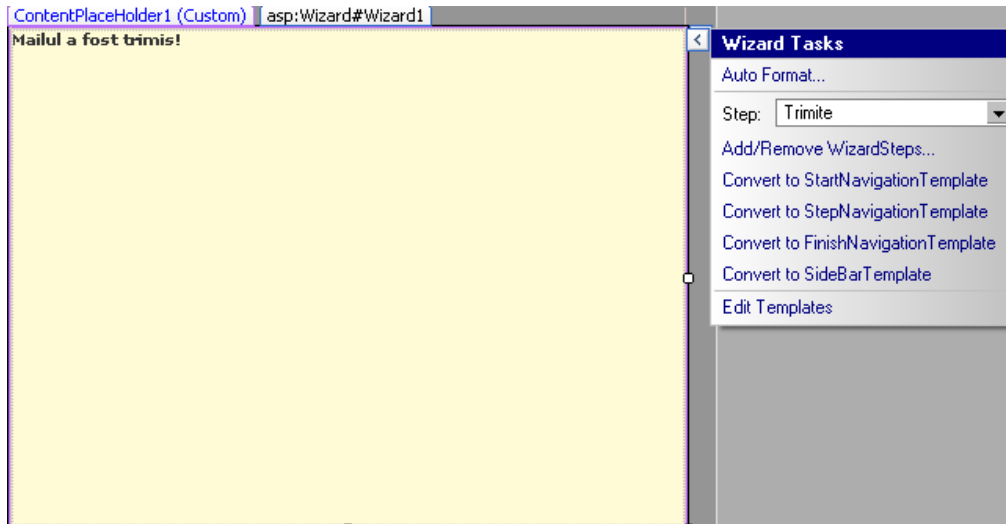


Figura 7.61 Ultimul pas al formularului.

La adăugarea controlului Wizard, puteți folosi opțiunea Add/Remove WizardSteps pentru a specifica numărul de pași ai controlului.

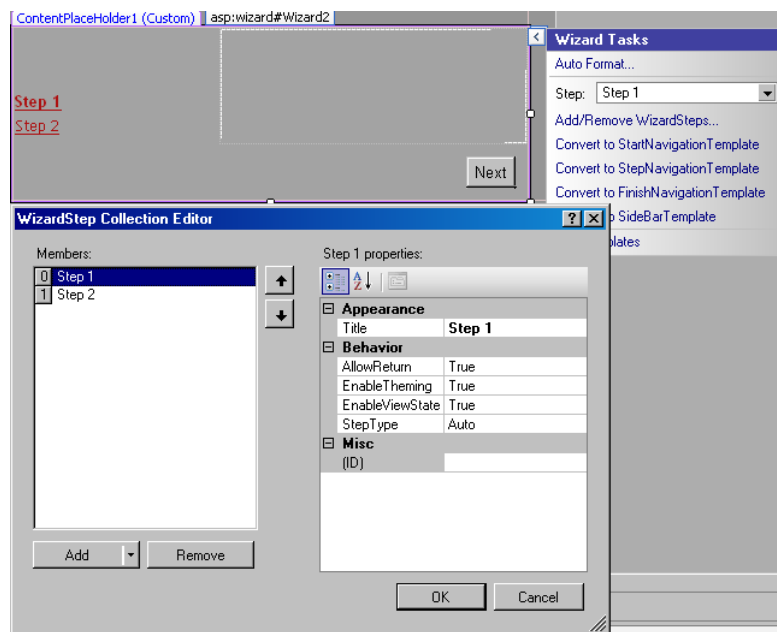


Figura 7.62 Adăugarea de pași pentru controlul wizard.

Butoanele afișate automat sunt determinate de tipul pasului.

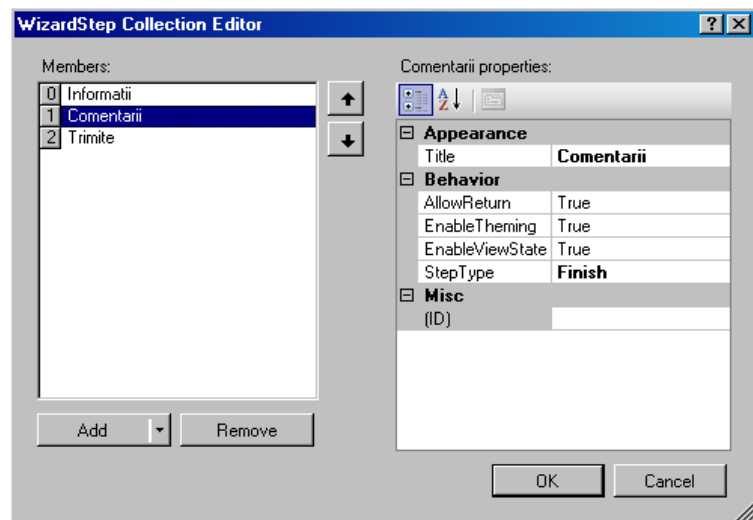


Figura 7.63 Al doilea pas este de tip Finish

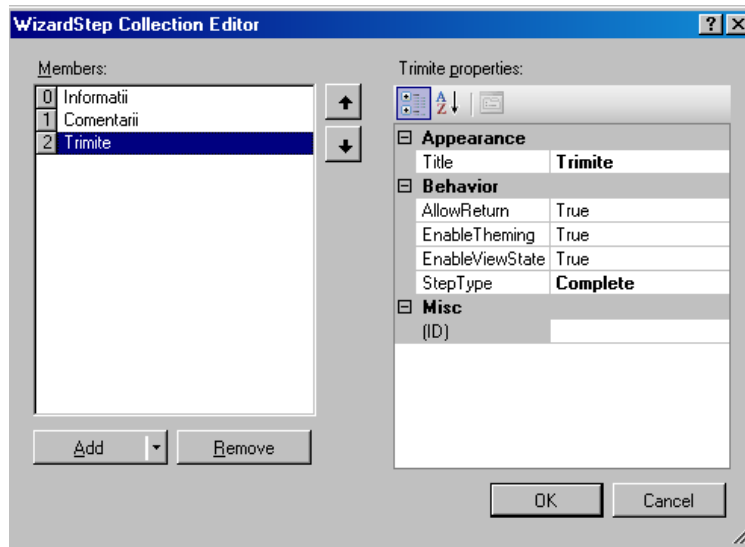


Figura 7.64 Ultimul pas este de tip Complete

Codul asp asociat controlului wizard este:

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<asp:Wizard ID="Wizard1" runat="server" BackColor="#FFFBD6"
  BorderColor="#FFDFAD" BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em"
  Height="318px" Width="431px" ActiveStepIndex="0"
  onfinishbuttonclick="Wizard1_FinishButtonClick">
<StepStyle VerticalAlign="Top" />
  <StartNextButtonStyle BackColor="#990000" BorderColor="Black" BorderStyle="Solid"
    Font-Bold="True" Font-Size="X-Small" ForeColor="White" Height="20px" Width="60px" />
  <FinishCompleteButtonStyle BackColor="#990000" BorderColor="Black" BorderStyle="Solid"
    Font-Bold="True" Font-Size="X-Small" ForeColor="White" Height="20px" Width="60px" />
  <StepNextButtonStyle BackColor="#990000" BorderColor="Black" BorderStyle="Solid"
    Font-Bold="True" Font-Size="X-Small" ForeColor="White" Height="20px" Width="60px" />
  <FinishPreviousButtonStyle BackColor="#990000" BorderColor="Black" BorderStyle="Solid"
    Font-Bold="True" Font-Size="X-Small" ForeColor="White" Height="20px" Width="60px" />
<WizardSteps>
  <asp:WizardStep runat="server" title="Informatii">
    <table cellpadding="0" cellspacing="1" class="style1">
      <tr>
        <td class="style4">
<asp:Label ID="Label1" runat="server" Font-Bold="True" Text="Numele dvs."></asp:Label>
          <td class="style3">
<asp:TextBox ID="txtNum" runat="server" Width="215px"></asp:TextBox>
          <td>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
            ControlToValidate = "txtNum" ErrorMessage="Nu ati introdus numele"
            Font-Bold="True" ForeColor="#990000">*</asp:RequiredFieldValidator>
          </td>
        </tr>
      <tr><td class="style4">
<asp:Label ID="Label2" runat="server" Font-Bold="True" Text="Email-ul">
</asp:Label>
      </td>
    </table>
  </asp:WizardStep>
</WizardSteps>
</asp:Content>
```



```

<td class="style3">
  <asp:TextBox ID="txtEmail" runat="server" Width="215px"></asp:TextBox>
</td>
<td>
  <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
    ControlToValidate="txtEmail" ErrorMessage="Nu ati introdus email-ul" Font-Bold="True"
    ForeColor="#990000">*</asp:RequiredFieldValidator>
  <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
    ControlToValidate="txtEmail" ErrorMessage="Nu ati scris corect email-ul"
    Font-Bold="True" ForeColor="#990000"
    ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*">*</asp:RegularExpressionValidator>
</td></tr>
<tr>
  <td class="style2" colspan="3">
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" Font-Bold="True"
      ForeColor="#990000" />
  </td></tr></table>
</asp:WizardStep>
<asp:WizardStep runat="server" title="Comentarii" StepType="Finish">
  <table class="style1">
    <tr>
      <td class="style4">
        <asp:Label ID="Label3" runat="server" Font-Bold="True"
          Text="Comentariile dvs (nu mai mult de 50 caractere)"></asp:Label>
      </td>
      <td class="style3">
        <asp:TextBox ID="txtComments" runat="server" Rows="5"
          TextMode="MultiLine" Width="217px"></asp:TextBox>
      </td>
      <td>
        <asp:CustomValidator ID="CustomValidator1" runat="server"
          ControlToValidate="txtComments" ErrorMessage="Nu mai mult de 50 caractere"
          Font-Bold="True" ForeColor="#990000" OnServerValidate=
"CustomValidator1_ServerValidate" ValidateEmptyText="True">*</asp:CustomValidator>
      </td></tr>
      <tr>
        <td class="style2" colspan="3">
          <asp:ValidationSummary ID="ValidationSummary2" runat="server"
            Font-Bold="True" ForeColor="#990000" />
        </td></tr></table></asp:WizardStep>
<asp:WizardStep runat="server" StepType="Complete" Title="Trimite">
<asp:Label ID="Label4" runat="server" Font-Bold="True"
  Text="Mailul a fost trimis!"></asp:Label></asp:WizardStep></WizardSteps>
<SideBarButtonStyle ForeColor="White" />
<NavigationButtonStyle BackColor="White" BorderColor="#CC9966" BorderStyle="Solid"
  BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em" ForeColor="#990000" />
<SideBarButtonStyle BackColor="#990000" Font-Size="Small" VerticalAlign="Top" Width="80px" />
<HeaderStyle BackColor="#FFCC66" BorderColor="#FFFBD6" BorderStyle="Solid"
  BorderWidth="2px" Font-Bold="True" Font-Size="0.9em" ForeColor="#333333"
  HorizontalAlign="Center" />
<StepPreviousButtonStyle BackColor="#990000" BorderColor="Black"
  BorderStyle="Solid" Font-Bold="True" Font-Size="X-Small" ForeColor="White"
  Height="20px" Width="60px" />
</asp:Wizard>
</asp:Content>

```

Controalele pentru introducerea numelui și e-mail-ului sunt validate prin intermediul unui control `RequiredFieldValidator`. Câmpul e-mail este validat și din punctul de vedere al formei șirului de caractere introdus, folosind un control `RegularExpressionValidator`.

Controlul de la pasul 2 este utilizat de utilizator pentru introducerea comentariilor. De aceea, valoarea proprietății `TextMode` este `MultiLine`. Dorim să limităm numărul maxim al caracterelor introduse la 50. Pentru a valida numărul de caractere introdus, folosim un control de tip `CustomValidator`, pentru care codul se va executa pe server.

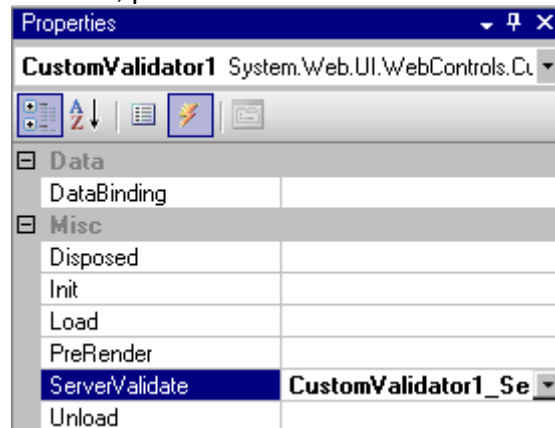


Figura 7.65 Evenimentul asociat controlului `CustomValidator`

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (txtComments.Text.Length > 50)
    {args.IsValid = false;}
    else
    {args.IsValid = true;}
}
```

La apăsarea butonului Finish de la pasul 2, se va trimite un e-mail. Mesajul va fi un obiect de tip `MailMessage`. Constructorul clasei are parametrii: from, to, subject, body. Pentru a trimite efectiv mail-ul, folosim clasa `SmtpClient`. Trimiterea unui mail presupune existența unui server Smp. În exemplul de mai jos, folosim server-ul smtp oferit gratis utilizatorilor Gmail. Parametrii `username@gmail.com` respectiv `username@gmail.com` trebuie înlocuiți cu datele contului de mail.

```
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    MailMessage message = new MailMessage(txtEmail.Text, "username@yahoo.com", "Feedback
de la " + txtNume.Text, txtComments.Text);
    SmtpClient mailClient = new SmtpClient();
    mailClient.Host = "smtp.gmail.com";
    mailClient.EnableSsl = true;
    mailClient.Credentials = new System.Net.NetworkCredential
("username@gmail.com", "password");

    mailClient.Port = 587;
    mailClient.Send(message);
    message.Dispose();
}
```



VII.8. Evaluare

1. Adăugați bazei de date o tabelă numită *Categorii*, care va conține toate categoriile posibile ale unui film. Modificați apoi controalele de tip DropDownList din paginile Movie.aspx respectiv AdaugăFilm.aspx astfel încât să preia datele direct din tabela nouă introdusă. Veți folosi un obiect de tip TableAdapter adăugat în DataSet-ul existent.
2. Adăugați o nouă tabelă numită Actori. Realizați un formular de introducere a datelor pentru noua tabelă. Modificați formularul de introducere a informațiilor despre film, astfel încât actorii să fie preluați din noua tabelă.
3. Modificați pagina upload.aspx astfel încât să realizeze doar upload-ul unui fișier, fără a mai salva în baza de date numele acestuia. Apoi modificați pagina detalii.aspx astfel încât să permită vizualizarea posterului upload-at în pagina upload.aspx.
4. Presupunând că firma dvs. are două locații într-un singur oraș, adăugați o nouă pagină numită Despre.aspx, care să conțină un control de tip ImageMap. Imaginea trebuie să fie o hartă cu cele două locații, iar la click pe fiecare dintre ele trebuie să se afișeze informațiile despre locația respectivă într-o nouă pagină.

VIII. Test de verificare a cunoștințelor

1. Care din următoarele obiecte poate fi folosit pentru a menține starea într-o aplicație ASP.NET ?
 - a. Session
 - b. Application
 - c. ViewState
 - d. Toate de mai sus
2. Care dintre următoarele modalități realizează un redirect către o pagină externă ?
 - a. Server.Transfer
 - b. Response.Redirect
 - c. a. și b.
 - d. Nici una din modalitățile de mai sus
3. Care din următoarele limbaje poate fi folosit pentru a scrie cod server side în ASP.NET ?
 - a. C#

- b. C
 - c. Visual Basic .Net
4. Pentru a delimita codul C# sau VB.Net scris în cadrul unei pagini se folosește tag-ul:
- a. < Body >
 - b. < Head >
 - c. < Script >
5. Controlul ASP.Net folosit pentru afișarea unui text într-o pagină web este:
- a. < asp:label >
 - b. < asp:listitem >
 - c. < asp:button >
6. Controlul ASP <asp:dropdownlist> corespunde cărui tag Html ?
- a. < Option >
 - b. < Select >
 - c. < List >
7. Controlul ASP < asp : listitem > corespunde cărui tag Html ?
- a. < Option >
 - b. < UL >
 - c. < List >
8. Proprietatea "EnableViewState" permite salvarea datelor introduse de utilizator?
- a. Da
 - b. Nu
9. Care din următoarele posibilități nu este o modalitate de menținere a stării ?
- b. Querystate
 - c. HiddenField
 - d. Application
 - e. Cookies
10. Într-o pagină web trebuie salvate informații de dimensiuni mici între două accesări ale paginii. Securitatea datelor nu este importantă. Care dintre următoarele posibilități este cea mai potrivită?
- a. cookie
 - b. query string
 - c. url
 - d. javascript function
 - e. hidden field
11. Care dintre următoarele evenimente este ultimul în cadrul ciclului de viață a unei pagini web?
- a. Page_Load
 - b. Validate
 - c. Page_Init
12. Care dintre următoarele proprietăți este comună tuturor controalelor de validare?
- a. ValueToCompare
 - b. ControlToValidate
 - c. InitialValue
 - d. ValidationExpression
 - e. ControlToCompare
13. Ce tip de dată are valoarea returnată de proprietatea IsPostBack
- a. bit
 - b. boolean
 - c. int
 - d. object
 - e. string
14. Într-o pagină web există un control de tip DropDownList pentru a reține nume de persoane. Controlul are două valori: Ion și Marius. La apăsarea unui buton de submit, se generează o acțiune de post back. În loc de două valori, controlul DropDownList conține 4: 2 valori „Ion” și 2 valori „Marius”. De ce?

- a. eroare la încărcarea paginii web
 - b. Proprietatea AutoPostBack pentru controlul DropDownList are valoarea false.
 - c. Directiva AutoEventWireUp are valoarea false
 - d. Nu este verificată proprietatea IsPostBack la încărcarea paginii.
15. Care este durata de viață a unui element reținut în obiectul View State?
- a. Elementul va exista cât timp există pagina curentă
 - b. Elementul va exista cât timp există sesiunea curentă
 - c. Elementul va exista cât timp există obiectul Application