გ. ღვინეფაძე

ინტერნეტსა და ინტრანეტში მომუშავე გამოყენებების დაპროექტება **DELPHI**-ის მეშვეობით

(კონსპექტი)

ტექნიკური უნივერსიტეტი

ინტერნეტსა და ინტრანეტში მომუშავე გამოყენებების დაპროექტება

შესავალი

დღეისათვის ინტერნეტი ყველაზე პერსპექტიული სივრცეა სხვადასხვა გამოყენებების განსათავსებლად. ქვემოთ მოყვანილ მასალაში აღწერილია ამ მიზნის მისაღწევად განკუთვნილი საშუალებები. აქვე აღვნიშნოთ, რომ მათი მეშვეობით შესაძლებელია საასპარეზოდ ინტრანეტული სივრცის გამოყენებაც.

იმ დისციპლინებიდან, რომლებიც უკვე შესწავლილი გვაქვს, პირველ რიგში, მოგვიხდება HTML, JavaScript, DELPHI 7 და SQL-ის მასალებისათვის ხელახლა თვალის გადავლება, რადგანაც, ძირითადად, სწორედ ამ პროგრამულ საშუალებებს დავეყრდნობით მოცემული დისციპლინის საკითხების გადმოცემისას.

ცნობილია, რომ ინტერნეტში ჩართულია სხვადასხვა პლატფორმებზე მომუშავე და სხვადასხვა ენებზე დაწერილი პროგრამების გამომყენებელი კომპიუტერები და ლოკალური ქსელები, მაგრამ მათ შორის საერთო ენის გამონახვა ხერხდება TCP/IP ოქმების წყვილით. ჩვენ მიერ ასაგები გამოყენებები კი TCP/IP-ს მიმართავენ უფრო მაღალი დონის ოქმების მეშვეობით. ერთ-ერთ ასეთ ყველაზე ცნობილ გამოყენებას წარმოადგენს ჩვენთვის კარგად ნაცნობი WWW – მსოფლიო ქსელი, ხოლო აღნიშნულ ქსელში განთავსებული WWW–სერვერები და საიტების ჩამთვალიერებელი პროგრამა-კლიენტები – ბროუზერები კი ერთმანეთს, ძირითადად, HTTP და HTP ოქმების მეშვეობით მიმართავენ. აქვე შევნიშნოთ, რომ, საერთოდ, HTTP ოქმის შესაძლებლობები უფრო ფართოა, ვიდრე მსოფლიო ქსელში მოგზაურობის სერვისის უზრუნველყოფა – კერძოდ, იგი შეიძლება გამოყენებულ იქნეს ნებისმიერ ორ გამოყენებად პროგრამას შორის კავშირის ორგანიზებისათვის, მაგალითად, კლიენტ-სერვერულ ქსელებში კომპიუტერებს შორის მონაცემების გასაცვლელად. ობიექტის ჩვენება ხდება URL-ის (ოქმი + დომენური მისამართი) მეშვეობით.

HTTP ოქმის დიდი ღირსება არის ის, რომ კლიენტსა და სერვერს შორის მუდმივი კავშირის არსებობა აუცილებელი არ გახლავთ - Web–ფურცლის ჩატვირთვის შემდეგ შეერთება წყდება. ამის მიზეზია ის, რომ HTTP ოქმი ტრანზაქციებზე არის ორიენტირებული: კლიენტი სერვერისაგან ითხოვს მხოლოდ მონაცემების განსაზღვრული, როგორც წესი, არცთუ დიდი, რაოდენობის გადაცემას. მონაცემების როლში შეიძლება გამოვიდნენ არამარტო HTML დოკუმენტები, არამედ ნებისმიერი სახის ისეთი მონაცემები, რომელთა გაგება ორივე მხარეს შეუძლია.

WWW–თან მუშაობისათვის საჭირო პროგრამული უზრუნველყოფა

სერვერული გამოყენებების შესასრულებლად აუცილებელია, კომპიუტერზე დაყენდეს WEB-სერვერ-პროგრამა, კლიენტზე კი, ცხადია, - ბროუზერი. როდესაც მიმდინარეობს სერვერული გამოყენების

დაპროექტება, არ არის საჭირო კავშირი გვქონდეს ინტერნეტთან, თუმცა მოითხოვება, რომ სათანადოდ გავაწყოთ WEB-სერვერ-პროგრამა.

თუ ჩვენს კომპიუტერზე უკვე დაყენებულია FrontPage პროგრამა, მაშინ ...\FrontPage Web კატალოგში უკვე განთავსებული იქნება Personal WEB Server სახელის მქონე WEB-სერვერი. ამასთან, ეს კატალოგი, თავის მხრივ, შეიცავს Content და Server ქვეკატალოგებს, უკანასკნელში კი მოთავსებულია vhttpd.exe ფაილი.

<u>სანამ სერვერს მივმართავდეთ რომელიმე გამოყენებიდან, აუცილებელია შესრულებაზე გავუშვათ vhttpd.exe ფაილი.</u>

Content კატალოგში განლაგებული ქვეკატალოგებიდან ჩვენთვის განსაკუთრებით საინტერესო გახლავთ cgi-bin. სწორედ მასში უნდა განვათავსოთ ჩვენ მიერ შექმნილი, სერვერზე შესასრულებელი .exe და .dll ფაილები.

<u>შენიშვნა</u>: ზოგიერთ სერვერზე ამ ფაილების შესრულებაზე გაშვება აღნიშნული კატალოგიდან დუმილით გათვალისწინებული არ გახლავთ. ასეთ შემთხვევებში cgi-bin კატალოგის თვისებების ფანჯარაში ჩართულ უნდა იქნეს ინდიკატორი: Allow Scripts or Programs to be Run. გასათვალისწინებელია ის ფაქტიც, რომ ზოგიერთ სერვერზე შესრულებადი ფაილების გასაშვებად გამოიყენება არა cgi-bin, არამედ scripts კატალოგი.

FrontPage-ის დაყენებისას გამორიცხული არ არის, პერსონალურ სერვერს არასწორად გადაეცეს კომპიუტერის სახელი. ასეთ შემთხვევაში მივმართავთ Control Panel → Network and Internet Connections → Identification ფანჯარას. ვკრებთ კომპიუტერის სახელს და აქვე Configuration ჩანართზე ვამოწმებთ, დაყენებულია თუ არა TCP/IP კომპონენტი.

თუ ყველაფერი სწორად გავაკეთეთ, მაშინ შესრულებადი ფაილების კატალოგში განლაგებულ ობიექტებს შეიძლება ბროუზერიდან, მაგალითად, ასე მივმართოთ:

<u>http://mycomputer/cgi-bin/ფაილის-სახელი.html</u>

ამასთან, როგორც წესი, დასაშვებია, ოქმი მითითებული არც იყოს.

<u>*HTML–დოკუმენტის მომზადება*</u>

ცნობილია, რომ ამ მიზნით შეიძლება გამოვიყენოთ როგორც მარტივი ტექსტური რედაქტორი Notepad, ასევე – სპეციალიზებული რედაქტორებიც. Delphi-ში HTML–დოკუმენტის მოსამზადებლად გათვალისწინებულია სპეციალური შაბლონი, რომელსაც ასე მივმართავთ:

File → New → Other→ Web Documents

ვირჩევთ HTML document პიქტოგრამას. ვკრებთ სასურველ კოდს და HTML-დოკუმენტს ვიმახსოვრებთ სასურველ საქაღალდეში.

საკუთარი ბროუზერის შექმნა

საინტერესოა, რომ DELPHI 7-ის მეშვეობით ჩვენ შეგვიძლია შევქმნათ საკუთარი ბროუზერიც.

ვაღებთ DELPHI-ის ახალ გამოყენებას. Internet-ჩანართიდან ფორმაზე გადაგვაქვს WebBrowser კომპონენტი და გავშლით მას მთელს ფორმაზე, align თვისებისათვის alClient მნიშვნელობის მინიჭებით.

ფორმის OnCreate ხდომილობის დამმუშავებელში ჩავწეროთ შემდეგი ოპერატორი:

WebBrowser1.Navigate('ფაილის-მისამართი');

არჩევანის გასაფართოვებლად გამოყენებაში დავამატოთ OpenDialog კომპონენტი და მის Filter თვისებაში ჩავწეროთ:

HTML-ფაილები (*.html, *. htm)	*.html; *. htm
ყველა ფაილი	*.*

გამოყენებაში ასევე ვამატებთ მენიუს Open განყოფილებით, რომლის ხდომილობის დამმუშავებელში შეგვყავს ოპერატორი:

```
If (OpenDialog1.Execute)
  Then  WebBrowser1.Navigate (OpenDialog1.FileName);
```

ვასრულებთ გამოყენებას. თავდაპირველად ბროუზერში ჩაიტვირთება წინასწარ შერჩეული ფაილი. შემდგომ კი შესაძლებელი იქნება სხვა არჩევანის გაკეთებაც. ამასთან, დასაშვებია, გაშვებული იქნეს Word-ის, Excel-ის და ა.შ. ფაილებიც.

ამჯერად ფორმაზე განვათავსოთ CoolBar პანელი-კომპონენტი. იგი საშუალებას იძლევა, ვიმუშავოთ ე.წ. გადაწყობად პანელებთან, რომლებიც, თავის მხრივ, ზოლებისაგან (bands) შედგებიან. ზოლებში განათავსებენ ToolBar ინსტრუმენტულ პანელებს და ნებისმიერ სხვა ფანჯრულ (როგორიცაა, მაგალითად, რედაქტირების ფანჯრები, პანელები და ა.შ.) თუ არაფანჯრულ (მაგალითად, ჭდეები) კომპონენტებს. აქვე შევნიშნავთ, რომ არაფანჯრული კომპონენტები გადაადგილებას არ ექვემდებარება.

CoolBar კომპონენტის განთავსების შემდეგ ფორმაზე გადავიტანოთ სხვა კომპონენტებიც, მაგალითად: ToolBar და Edit. ვხედავთ, რომ თავდაპირველად მათ ეთმობა მთელი ზოლი, მაგრამ კომპონენტის მარცხენა კიდეზე არსებულ ვიწრო ზოლზე ჩაჭიდებით შესაძლებელია მისი სასურველ პოზიციაზე გადატანაც.

ზოლებს შეიძლება თვისებები განვუსაზღვროთ ზოლების რედაქ-ტორის მეშვეობით (მისი გამოძახება შესაძლებელია ზოლის კონტექსტური მენიუდან ან ზოლზე ორჯერ დაწკაპუნებით).

გადაწყობის თვალსაზრისით, მომხმარებელს კიდევ უფრო მეტ შესაძლებლობებს აწვდის ControlBar კომპონენტი.

განვათავსოთ ფორმაზე ControlBar კომპონენტი. დავიტანოთ მასზე აგრეთვე ToolBar და Edit კომპონენტებიც. თითოეული მათგანისათვის დასაშვებია Drag&Doc ტექნოლოგიის გამოყენება. მხოლოდ, ამ მიზნის მისაღწევად, საჭიროა, შესაბამისი მნიშვნელობები მივანიჭოთ კომპონენტის შემდეგ თვისებებს:

DragMode=dmAutomatic

DragKind=dkDock

ფორმაზე კვლავ CoolBar კომპონენტი განვათავსოთ, მასზე კი - ToolBar ინსტრუმენტული პანელი. მის ShowCaptions თვისებას მივანიჭოთ true მნიშვნელობა. თაგვის მარჯვენა ღილაკზე დაწკაპუნებით პანელზე განვათავსოთ ორი ღილაკი. ერთს დავარქვათ TBBack, მეორეს კი – TBForward. შესაბამისი მნიშვნელობები მივანიჭოთ კომპონენტების Caption თვისებებსაც. რადგანაც გამოყენების გაღების მომენტში ამ ღილაკების ფუნქციონირებას აზრი არა აქვს, ამის გამო ორივესათვის Enabled თვისებას მივნიჭოთ false მნიშვნელობა.

ამის შემდეგ CoolBar-ზე გადავიტანოთ ComboBox კომპონენტი. დავარქვათ მას, მაგალითად, CBURL სახელი. ამ კომპონენტის ფანჯარაში შესაძლებელი იქნება სასურველი URL-ის ან HTML ფაილის სახელის ჩაწერა/არჩევა.

CoolBar პანელისათვის true მნიშვნელობა მივანიჭოთ AutoSize და ShowText თვისებებს.

Bands თვისებასთან დაკავშირებულ რედაქტორში შესვლის შემდეგ მეორე ზოლში, რომელზეც განთავსებულია CBURL, ჩავწეროთ “მისამართია: ”.

ფორმაზე გადავიტანოთ StatusBar პანელი. მისი SimplePanel თვისებისათვის მნიშვნელობად ავირჩიოთ true.

ამ მანიპულაციების შემდეგ მოდულში შეგვაქვს კოდი:

```
unit Unit1;
interface
...
type
  Tform1 = class(TForm)
     ...
procedure FormCreate (Sender: TObject);
procedure CBURLClick (Sender: TObject);
procedure CBURLKeyDown (Sender: TObject; var Key: Word;
    Shift: TShiftState);
procedure WebBrowser1BeforeNavigate2 (Sender: TObject;
     const pDisp: IDispatch;
     var URL, Flags, TargetFrameName, PostData,
        Headers: OleVariant; var Cancel: WordBool);
procedure WebBrowser1ProgressChange (Sender: TObject;
                Progress, ProgressMax: Integer);
procedure MOpenClick (Sender: TObject);
procedure TBBackClick (Sender: TObject);
```

```
procedure TBForwardClick (Sender: TObject);

private
{ Private declarations }
procedure Load;
public
{ Public declarations }
end;
var Form1: TForm1;

implementation
{$R * .DFM}

Procedure TForm1.FormCreate(Sender:  TObject);
begin
  // URL-ის ან ფაილის ჩატვირთვა შესრულების დაწყების მომენტში
   CBURL.Text :=  'D:\TESTS\HTML1.html';
   Load;
end;

procedure TForm1.Load;
begin
// CBURL.Text თვისებაში მოცემული URL-ის ან ფაილის ჩატვირთვა
  WebBrowser1.Navigate(CBURL.Text);
end;

procedure TForm1.CBURLClick(Sender :TObject);
begin
//   CBURL-ში არჩევანის გაკეთებისას ჩატვირთვა
  Load;
end;

procedure TForm1.CBURLKeyDown(Sender: TObject; var Key: Word;
                                        Shift: TShiftState);
begin
//  CBURL-ზე Enter-ის დაჭერისას ჩატვირთვა
  If Key = VK_Return
     then Load;
end;

procedure TForm1.WebBrowser1Beforenavigate2(Sender: TObject;
                        const pDisp: IDispatch; var URL, Flags,
                        TargetFrameName, PostData, Headers: OleVariant;
                         Var Cancel: WordBool);
// ჩატვირთვის წინ ხდომილობის დამუშავება
var
     Index: Integer;
begin
```

```
// CBURL სიის მართვა
Index := CBURL. Items. IndexOf (URL);
If Index =  -1
  then
     begin
       CBURL.Items.Insert(0,URL);
       CBURL.ItemIndex := 0;
     end
  else CBURL.ItemIndex := Index;
// TBForward  და TBBack ღილაკების შეღწევადად ქცევა
if CBURL.ItemIndex > 0
   then TBForward.Enabled := true
else TBForward.Enabled := false;
if CBURL.ItemIndex < CBIRL.Items.Count – 1
   then TBBack.Enabled := true
   else TBBack.Enabled := false;
end;

procedure TForm1.WebBrowser1ProgressChange(Sender: Tobject;
                                    Progress, ProgressMax: Integer);
// მდგომარეობის პანელში ტექსტის შეტანა
begin
  if (Progress > 0)
     then StatusBar1.SimpleText :=
       Format ('%s დოკუმენტი: წაკითხულია %d კილობაიტი %d-დან',
                  [WebBrowser.LocationName, Progress div 1024,
                   ProgressMax div 1024] );
end;

procedure TForm1.MOpenClick(Sender: TObject);
begin
// ფაილის გაღების დიალოგის გამოძახება
  If (OpenDialog1.Execute)
   then begin
      CBURL.Text := OpenDialog1.FileName;
      Load;
   end;
end;

procedure TForm1.TBBackClick(Sender: TObject);
begin
   // წინა დოკუმენტზე გადასვლა
   CBURL.Text := CBURL.Items[CBURL.ItemIndex + 1];
   Load;
end;

procedure TForm1.TBForwardClick(Sender: TObject);
begin
```

```
  // შემდეგ დოკუმენტზე გადასვლა
  CBURL.Text := CBURL.Items[CBURL.ItemIndex - 1];
  Load;
end;

end.
```

განვიხილოთ ზემოთ მოყვანილი კოდი.

FormCreate ფუნქცია წარმოადგენს ფორმის შექმნის OnCreate ხდომილობის დამმუშავებელს. მას CBURL სიის Text თვისებაში შეაქვს იმ ფაილის სახელი ან WEB-ფურცლის URL, რომელიც ეკრანზე უნდა აისახოს მოცემული გამოყენების გაშვებისთანავე. ეს კი მოხდება Load ფუნქცია-უტილიტის გამოძახებით, რის შედეგადაც WebBrowser1 ბროუზერში Navigate მეთოდით ჩაიტვირთება CBURL.Text-ში მითითებული ფაილი თუ WEB-ფურცელი.

აღვნიშნოთ, რომ Load უტილიტის გამოძახება ხდება აგრეთვე CBURL სიაში ელემენტის არჩევისას (იხ. CBURLClick ფუნქცია) და შერჩეული ელემენტისათვის ENTER ღილაკზე დაჭერის მომენტში (იხ. CBURLKeyDown ფუნქცია).

WebBrowser1Before1Navigate2 ფუნქცია გამოიძახება ახალ დოკუმენტზე გადასვლის მომენტში, როგორც ამ ხდომილობის დამმუშავებელი. Cancel პარამეტრი იძლევა გადაფიქრების საშუალებას. IndexOf მეთოდით მოწმდება, არსებობს თუ არა მოთხოვნილი ელემენტის შესახებ სიაში ინფორმაცია. უარყოფითი პასუხის შემთხვევაში მეთოდი აბრუნებს -1 მნიშვნელობას, შესაბამისი ინფორმაცია ჩაისმება სიის თავში Insert მეთოდით და, ცხადია, ინდექსს მიენიჭება ნულოვანი მნიშვნელობა. სხვა შემთხვევაში, ანუ სიაში ელემენტის არსებობისას, ინდექსი მიიღებს ამ ელემენტის პოზიციის შესატყვის მნიშვნელობას.

შემდეგ, იმის და მიხედვით, არსებობს თუ არა მოცემული ელემენტისათვის სიაში წინ და უკან მყოფი ელემენტები, განისაზღვრება TBBack და TBForward ღილაკების ქმედითუნარიანობა მოცემული მომენტისათვის.

WebBrowser1ProgressChange ფუნქცია დიდი მოცულობის დოკუმენტების ჩატვირთვის შემთხვევაში იძლევა ინფორმაციას ამ პროცესის მიმდინარეობის შესახებ სტატუსის სტრიქონში.

MOpenClick ფუნქცია წარმოადგენს მენიუში Open პუნქტზე დაწკაპუნების ხდომილობის დამმუშავებელს.

მსგავსი დანიშნულება აქვთ TBBackClick და TBForwardClick ფუნქციებსაც.

დაბოლოს, აღვნიშნოთ, რომ საკუთარი ბროუზერის შექმნა საშუალებას იძლევა, გვქონდეს ინსტრუმენტი ჩვენი კონკრეტული მიზნების განსახორციელებლად. ხშირ შემთხვევაში ეს არის ბროუზერი, რომელიც ამა თუ იმ დაწესებულების თანამშრომლებს აწვდის მხოლოდ მათთვის საჭირო რესურსებთან შეღწევის საშუალებას.

CGI ინტერფეისი

დინამიკური WEB-ფურცლებისათვის

სტატიკური ინფორმაციის შემცველი WEB-ფურცლის გამოძახებისაგან განსხვავებით, დინამიკური HTML-დოკუმენტი WEB-სერვერზე ფორმირდება მომხმარებლის კონკრეტული მოთხოვნიდან გამომდინარე და მხოლოდ ამის შემდეგ გადაეგზავნება იგი კლიენტს.

ასეთი ტიპის დოკუმენტების ფორმირებისათვის ძირითადად გამოიყენება ორი გზა:

- სერვერის მიერ შესრულებადი .exe ფაილის (პროგრამის) გამოძახება.
- სერვერის მიმართვა დინამიკურად მიერთებადი DLL ბიბლიოთეკისადმი.

პირველი გზა გულისხმობს CGI ან WIN-CGI, მეორე კი - ISAPI ან NSAPI ინტერფეისის გამოყენებას. თითოეულ მიდგომას გააჩნია თავისი დადებითი და უარყოფითი მხარეები.

CGI-გამოყენებები სერვერისაგან ინფორმაციას იღებენ სტანდარტული შეტანის საშუალებებისა და გარემოს ამსახველი ცვლადების მეშვეობით. მათ მიერ ფორმირებული HTML-დოკუმენტები ასევე სტანდარტული გამოსასვლელით გადაეცემა კლიენტს.

ყველაზე პრიმიტიულ შემთხვევაში CGI-გამოყენება შეიძლება რეალიზებული იქნეს, როგორც კონსოლ-პროგრამა. ასეთი პროგრამის შესაქმნელად უნდა გავცეთ შემდეგი ბრძანება:

File → New → Other

New ფურცელზე მივმართავთ Console Aplication პიქტოგრამას.

ვკრებთ კოდს:

```
program Ptime;
{ $APPTYPE CONSOLE }
uses
  SysUtils;

var Days:  array [1..7]  of  string =
      (‘კვირა’, ‘ორშაბათი’, ‘სამშაბათი’, ‘ოთხშაბათი’, ‘ხუთშაბათი’,
       ‘პარასკევი’, ‘შაბათი’);
begin

writeln (‘<HTML>’);
writeln (‘<H1> მოგესალმებით!</H1><HR>’);
writeln ('დღეს არის '  +  DateToStr(Date)  +  ' ('  +
            Days[DayOfWeek(Date)]  +  ')<HR>');
writeln (FormatDateTime('თბილისის დროით h საათი და m წუთია. ',  Time));
writeln (‘</HTML>’);

end.
```

პროგრამას შევინახავთ და გავუშვებთ კომპილაციაზე. თუ ამ პროგრამის შესრულებას ჩვეულებრივი გზით შევეცდებით, მაშინ ეკრანზე გამოგვივა DOS ფანჯარა.

საჭიროა, ეს შესრულებადი მოდული მოვათავსოთ ...\FrontPage Web\Content\cgi-bin კატალოგში. ბროუზერის სამისამართო ველში კი ვკრებთ:

mycomputer/cgi-bin/ ფაილის-სახელი.exe

(ანუ მოცემულ შემთხვევაში PTime.exe)

CGI-გამოყენებების უდავო ღირსებაა არის ის, რომ ისინი შეიძლება განთავსდნენ პრაქტიკულად ნებისმიერ WEB-სერვერზე, მაგრამ გამოყენებისადმი ყოველი მოთხოვნის გადაგზავნისას ხელახლა ხდება რესურსების მობილიზება-გათავისუფლება, რაც დროის საკმაოდ დიდ დანახარჯებთან არის დაკავშირებული.

ამ ნაკლისაგან თავისუფალია ISAPI და NSAPI პროგრამული ინტერფეისები. (იხ. ქვემოთ).

WEB DELPHI სერვერი

ზემოთ განხილული პრიმიტიული კონსოლური პროგრამა მისი თითოეული გამოძახებისას ქმნიდა დინამიკურ WEB-ფურცელს (დროისა და თარიღის ახალი მნიშვნელობებით). ადვილი შესამჩნევია, რომ მასში გათვალისწინებული არ გახლდათ დროის რეალურ რეჟიმში მომხმარებლის ახალ-ახალ მოთხოვნებზე რეაგირების შესაძლებლობა.

DELPHI 7-ში ჩადებულია რიგი საშუალებებისა კლიენტსა და სერვერს შორის სწორედ ამგვარი ურთიერთობების დასამყარებლად.

სერვერული გამოყენებების დასაპროექტებლად გათვალისწინებულია TWebModule ტიპის WEB-კომპონენტი (არსებობს სხვა კომპონენტებიც, რომელთაც შემდგომ განვიხილავთ). ხაზგასმით აღვნიშნავთ, რომ ეს კომპონენტი უშუალოდ კომპონენტების პალიტრაში არ ფიგურირებს, რის გამოც გამოყენებაში მისი ჩართვისათვის უნდა გავცეთ შემდეგი ბრძანება:

File → New → Other

ამასთან, New items დიალოგურ ფანჯარაში New ჩანართზე მივმართავთ Web Server Aplication პიქტოგრამას.

გამოდის ფანჯარა, რომელშიც ვირჩევთ სერვერის ტიპს (მოცემულ შემთხვევაში ეს გახლავთ CGI Stand-alone executable - CGI-ში შესრულებადი მოდული).

OK და გამოდის ცარიელი ფანჯარა, რომელშიც დასაშვებია სხვადასხვა კომპონენტების განთავსება (ბაზასთან კავშირის, ფურცლების შემქმნელთა და სხვ.). ჯერჯერობით თავი შევიკავოთ ასეთი ქმედებებისაგან და ფანჯრის კონტექსტური მენიუდან Action Editor პუნქტის არჩევის შედეგად გამოვიძახოთ ე.წ. *მოქმედებების რედაქტორი.*

ობიექტების ინსპექტორის ფანჯარაში შეიძლება ჩავათვალიეროთ-განვსაზღვროთ WEB-მოდულის თვისებები. WEB-მოდულის ძირითადი თვისება არის Actions, რომლის ელემენტები ასახავენ ცალკეულ შესაძლო მოქმედებებს, ესენი კი, კლიენტის მოთხოვნის საფუძველზე, შეიძლება

განახორციელოს სერვერულმა პროგრამამ. თითოეული მოქმედების ძირითადი თვისებაა PathInfo. მის არსში უკეთ გასარკვევად მოვიყვანოთ ადრე განხილულთან შედარებით უფრო რთული სახით წარმოდგენილი URL-ის მაგალითი:

http://www.myserv.com/p1/serv1.exe/action1?value=5&Resp=Yes

ვხედავთ, რომ ოქმს მოსდევს ჰოსტის სახელი (ანუ სერვერის მისამართი). მოცემულ შემთხვევაში ეს არის

www.myserv.com

შემდეგ ფიგურირებს სერვერული გამოყენების სახელი (მანამდე გზის ჩვენებით):

/p1/serv1.exe

ამის შემდეგ მითითებულია URL-ის არააუცილებელი PathInfo ნაწილი:

/action1

სწორედ ეს ინფორმაცია განსაზღვრავს სერვერული გამოყენების მიერ ინფორმაციის დამუშავების წესს.

მაგალითად, შეიძლება მიეთითოს პროცედურა, რომელიც მოახდენს მოცემული შეტყობინების დამუშავებას. "?" სიმბოლოს შემდეგ ჩაიწერება მოთხოვნა. ჩვეულებრივ, იგი წარმოადგენს ერთმანეთთან "&" სიმბოლოებით დაკავშირებული შემდეგი ფრაგმენტების მიმდევრობას:

იდენტიფიკატორი=მნიშვნელობა & იდენტიფიკატორი=მნიშვნელობა...

ამ ინფორმაციას სერვერული გამოყენება ამუშავებს.

დავუბრუნდეთ მოქმედების მთავარ PathInfo თვისებას. ცხადია, იგი ორგანულ კავშირშია URL-ის ზემოთ განხილულ PathInfo ელემენტთან. მათი თანხვდენის შემთხვევაში მუშაობას იწყებს შესაბამისი ხდომილობის დამმუშავებელი, რომლის სათაურს ასეთი სახე აქვს:

```
procedure TWebModule1.WebModule1WebActionItem1Action (
                    Sender: Tobject; Request: TWebRequest;
                    Response: TWebResponse; var Handled: Boolean);
```

აქ Request პარამეტრი წარმოადგენს TWebRequest ტიპის მოთხოვნის ობიექტს, რომლის string ტიპის Query თვისება არის სწორედ URL-ის Query უბანი. მოცემული შემთხვევისათვის მას ექნება სახე:

Value=5 & Resp=Yes

რამდენადმე განსხვავებული - სიის სახით წარმოგვიდგენს იმავე ინფორმაციას Request ობიექტის სხვა - QueryFields თვისება (მისი ტიპია TStrings). მოცემული შემთხვევისათვის ეს თვისება შემდეგი ორი სტრიქონის სახით წარმოგვიდგება:

"Value=5" და "Resp=Yes"

პარამეტრის მნიშვნელობებისადმი მიმართვა შესაძლებელია შემდეგი ნოტაციით:

Request. QueryFields.Values['Resp']

განსახილველ შემთხვევაში ნოტაცია დაგვიბრუნებს 'Yes' სტრიქონს.

Response პარამეტრი კი მომხმარებელს პასუხს უბრუნებს string ტიპის Content თვისების მეშვეობით. უმეტესწილად, ეს პასუხი გახლავთ HTML ტექსტი.

Handled პარამეტრი უჩვენებს, დამთავრდა თუ არა Response ობიექტის ფორმირება. ზოგჯერ საჭიროა მის შექმნაზე მუშაობა გააგრძელონ სხვა მოქმედებებმა. ასეთ შემთხვევაში Handled პარამეტრს false მნიშვნელობა მიენიჭება და Response ობიექტი გადაეცემა შესაბამის მოქმედებას მის Content თვისებაში ჩაწერილი, წინა მოქმედებების მიერ გამომუშავებული ინფორმაციით.

ამრიგად, ჩვენ განვიხილეთ მოქმედების ობიექტის PathInfo თვისება და OnAction ხდომილობის დამმუშავებელი.

აღვნიშნოთ, რომ მოქმედების ობიექტს გააჩნია აგრეთვე Default თვისება. ცხადია, იგი მხოლოდ ერთი მოქმედებისათვის შეიძლება იყოს დაყენებული true მდგომარეობაში. მოქმედების ობიექტის Enabled თვისება გვამცნობს, შეღწევადია თუ არა ეს ობიექტი.

თვით სერვერს (რომელიც TWebModule ტიპისაა) შეიძლება განვუსაზღვროთ დამმუშავებელნი BeforeDispatch და AfterDispatch ხდომილობებისათვის, ანუ მოთხოვნის დამუშავების დაწყებისა და დამთავრების სიტუაციებისათვის. გაანალიზების საფუძველზე, BeforeDispatch დამმუშავებელი მოთხოვნას მაშინვე გასცემს პასუხს ანდა, Enabled თვისებისადმი მნიშვნელობების მინიჭებით, ჩართავს-გამორთავს შესაბამის მოქმედებებს. მეორე დამმუშავებლის მეშვეობით კი შესაძლებელი ხდება, დაზუსტებულ იქნეს ხდომილობების მიერ მომზადებული პასუხი.

<u>სერვერული გამოყენების მაგალითი</u>

დავაპროექტოთ შემდეგი მარტივი სახის გამოყენება:

კლიენტს სერვერი უგზავნის შეკითხვას «რამდენია ორჯერ ორი»? ამასთან, ეკრანზე აისახება ორი ალტერნატიული პასუხი: სწორი და მცდარი. არჩევანის სერვერზე გადაგზავნის შემდეგ კლიენტს ეცნობება, რამდენად წარმატებული აღმოჩნდა მისი გონებრივი მოღვაწეობის შედეგი.

DELPHI-ში ვქმნით სერვერული გამოყენების მოდულს File → New → Other→ Web Server Application ბრძანების გაცემით. ვაღებთ მოქმედებების რედაქტორს და ზემოთ განხილული შემთხვევის ანალოგიურად შეგვაქვს მასში ორი მოქმედება. პირველი მათგანი რეაგირებს მომხმარებლის არჩევანზე. მის PathInfo თვისებას ვანიჭებთ “/1” მნიშვნელობას. მეორე – დუმილით გათვალისწინებული მოქმედების დანიშნულება კი მომხმარებელისათვის შეკითხვის დასმა არის. ამ მოქმედების თვისებას true მნიშვნელობას ვანიჭებთ.

შემდეგი ნაბიჯია მოქმედებებისათვის OnAction ხდომილობების დამმუშავებლის დაწერა. ჯერ შევქმნათ კოდი დუმილით გათვალისწინებული მოქმედების დამმუშავებლისათვის:

```
procedure TWebModule1.WebModule1WebActionItem2Action (
                    Sender: Tobject; Request: TWebRequest;
                    Response: TWebResponse; var Handled: Boolean);
begin
 Response.Content := “<H1> შევისწავლოთ გამრავლების ტაბულა! </h1>” +
     “<P> გვიპასუხეთ, რამდენია ორჯერ ორი?</p><HR>” +
```

```
  "<A HREF = 'TwiceTwo.exe/1?4'>4</a> თუ " +
  "<A HREF = 'TwiceTwo.exe/1?5'>5</a> ? ";
  Handled := true;
end;
```

ვხედავთ, რომ HTML-კოდში ჩართულია ორი დაყრდნობა "4" და "5" ტექსტებით. ნებისმიერი პასუხის არჩევისას მოთხოვნა გადაეგზავნება პირველ მოქმედებას. არჩეული დაყრდნობის ტექსტის გაანალიზების შედეგად პირველი მოქმედება სხვადასხვა პასუხებს აფორმირებს.

პირველი მოქმედების OnAction ხდომილობის დამმუშავებელს შეიძლება ასეთი სახე ჰქონდეს:

```
procedure TWebModule1.WebModule1WebActionItem1Action (
                    Sender: Tobject; Request: TWebRequest;
                    Response: TWebResponse; var Handled: Boolean);
begin
 if (Request.Query = '4')
  then Response.Content := 'პასუხია 4. თქვენ შესანიშნავი მათემატიკოსი
ბრძანდებით!'
  else Response.Content := 'პასუხია 5. თქვენ მათემატიკაში ცოტა
მოიკოჭლებთ!'
  Handled := true;
end;
```

პროექტს შევინახავთ, ვაკომპილირებთ და exe მოდულის სახით განვათავსებთ ჩვენი WEB-სერვერის შესრულებად კატალოგში. ახლა უკვე შესაძლებელია მისი ბროუზერში გამოძახება.

აღვნიშნოთ, რომ შეიძლებოდა ეს ამოცანა სხვაგვარადაც (უფრო მარტივად) გადაგვეწყვიტა – შეკითხვა პირდაპირ HTML-ფაილში ჩაგვედო, ნაცვლად მისი ფორმირებისა დუმილით გათვალისწინებული მოქმედების მიერ:

```
<HTML>
<HEAD>
<TITLE> გამრავლების ტაბულა </title>
</head>

<BODY>
<H1> შევისწავლოთ გამრავლების ტაბულა! </h1>
 <P> გვიპასუხეთ, რამდენია ორჯერ ორი?</p>
<HR>
<A HREF = "http://mycomputer/cgi-bin/TwoTwo.exe/1?4"> 4 </a> თუ
<A HREF = "http://mycomputer/cgi-bin/TwoTwo.exe/1?5"> 5 </a> ?
<HR>
</body>
</html>
```

აქ ხაზი უნდა გავუსვათ შემდეგ გარემოებას. რადგანაც მოცემულ შემთხვევაში HTML–ფაილი და შესრულებადი (exe) ფაილი შესაძლოა, სხვადასხვა საქაღალდეებში განთავსდნენ, წინა შემთხვევისაგან

განსხვავებით, დაყრდნობებში მიეთითება შესრულებადი ფაილის სრული მისამართი.

ხელახლა ფორმირებულ exe-ფაილში, ცხადია, გასათვალისწინებელი იქნება მხოლოდ ერთადერთი მოქმედება, რომლის PathInfo="\1".

HTML–კოდში ფორმების და ცხრილების გამოყენება

ზემოთ მოყვანილ მაგალითში კლიენტის სხვადასხვა პასუხების ფორმირებისათვის გამოყენებული იყო დაყრდნობები. მაგრამ, როცა პასუხების რიცხვი დიდია, ამასთან, განსხვავებული ფორმატის მქონე მონაცემების შემცველიც, ამჯობინებენ <FORM> </form> ელემენტის გამოყენებას. მის საწყის ტეგში განისაზღვრება რამდენიმე ატრიბუტის მნიშვნელობა. მათგან, უპირველეს ყოვლისა, აღვნიშნოთ action ატრიბუტი, რომლის მნიშვნელობა მიუთითებს მომხმარებლის მიერ ფორმაში შეტანილი მონაცემების მიმღების URL-ზე. მოვიყვანოთ მაგალითი:

action = "http://mycomputer/cgi-bin/test.exe"

შემდეგ, method ატრიბუტი („get" ან „post" მნიშვნელობით) გამოიყენება HTTP პროტოკოლის გამოყენებული მეთოდის მითითებისათვის.

get მეთოდის გამოყენების შემთხვევაში მოთხოვნაში ჩადებული მონაცემები იმყოფება Request ობიექტის QueryFields თვისებაში, ხოლო post მეთოდისათვის – ამავე ობიექტის ContentFields თვისებაში.

გავიხსენოთ, რომ კომპონენტები ფორმაში შეიტანება შემდეგი ელემენტების მეშვეობით:

<INPUT type="text">

<INPUT type="submit">

<TEXTAREA> და სხვ.

გარდა ტიპისა, ამ ელემენტებს გააჩნიათ რიგი სხვა ატრიბუტებისა.

submit ღილაკით სერვერულ გამოყენებას ფორმიდან გადაეგზავნება მონაცემები - name=value წყვილით სახელდებული სტრიქონების თანმიმდევრობა.

იმის და მიხედვით, რომელი მეთოდი არის არჩეული: post თუ get, სერვერზე გადაგზავნილი ატრიბუტების მნიშვნელობები სხვადასხვა გზით წაიკითხება:

Request.ContentFields.Values('result')

Request.QueryFields.Values('result')

ორივე გამოსახულება დაგვიბრუნებს result ატრიბუტის მნიშვნელობას.

ფორმაში კომპონენტების უკეთ განლაგების მიზნით, ხშირად იყენებენ ცხრილებს. ცხრილი შეიცავს სტრიქონებს, სტრიქონები შედგება უჯრედებისაგან. სწორედ უჯრედებში განალაგებენ კომპონენტებს.

განვიხილოთ ისეთი გამოყენების დაპროექტების მაგალითი, როდესაც მომხმარებელს სთავაზობენ, პასუხი გასცეს შეკითხვაზე – რამდენია ორი შემთხვევით არჩეული ერთთანრიგიანი რიცხვის ნამრავლი. მომხმარებლის წინაშე ეკრანზე უნდა აისახოს ამდაგვარი სახის ინფორმაცია:

შეამოწმეთ, რამდენად კარგად იცით გამრავლების ტაბულა!

სავარაუდო შედეგის ფანჯარაში შეტანის შემდეგ დააწკაპუნეთ «პასუხი» ღილაკზე:

7 X 3 = -------- -- პასუხი –

ახალი რიცხვები-

დავწეროთ კოდი ამ სურათის ამსახველი HTML-დოკუმენტისათვის:

```
<HTML>
<HEAD>
<TITLE> გამრავლების ტაბულა </title>
</head>

<BODY>
<H1> შეამოწმეთ, რამდენად კარგად იცით გამრავლების ტაბულა!
</h1>

<P>სავარაუდო შედეგის ფანჯარაში შეტანის შემდეგ დააწკაპუნეთ
«პასუხი» ღილაკზე:
 </p>

<FORM method="POST" action="http://mycomputer/cgi-bin/mult.exe/1" >
<TABLE>
 <TR><TD><input type="text" name="num1" size="1"></td>
        <TD> x </td>
        <TD><input type="text" name="num2" size="1"></td>
        <TD> = </td>
        <TD><input type="text" name="result" size="3"></td>
        <TD><input type="submit" name="post" value="პასუხი"></td>
  </tr>
</table>
</form>

 <FORM method="POST" action="http://mycomputer/cgi-bin/mult.exe/">
   <P><input type="submit" name="New" value="ახალი რიცხვები"></p>
 </form>

</body>
</html>
```

<u>კლიენტსა და სერვერს შორის მონაცემების გაცვლა</u>

ახლა კი ავაგოთ სერვერული გამოყენება, რომელმაც უნდა გადაწყვიტოს ზემოთ დასმული ამოცანა.

დასაწყისში ყურადღება გავამახვილოთ ერთ გარემოებაზე, რომელსაც ადგილი აქვს ამდაგვარი სახის გამოყენებების შექმნის დროს:

დავუშვათ, დიალოგი იწყება კლიენტისათვის ორი რიცხვის გაგზავნით. როგორც უკვე ვიცით, ამის შემდეგ სერვერსა და კლიენტს შორის კავშირი წყდება. სერვერული გამოყენების ხელმეორედ გამოძახება ხდება მხოლოდ მაშინ, როცა პასუხის გაცემის შემდეგ მომხმარებელი აწკაპუნებს „პასუხი“ ღილაკზე. მაგრამ საქმე ისაა, რომ სერვერულმა გამოყენებამ უკვე აღარ იცის, რა რიცხვები გადაუგზავნა მან კლიენტს წინა დაკავშირებისას. ამ პრობლემის გადასაწყვეტად არსებობს ორი გზა:

- გადაგზავნილი რიცხვები დამახსოვრებული იქნეს სერვერის გარე მეხსიერებაში;
- პასუხის გაცემის მომენტში კლიენტმა სერვერს უკან გადმოუგზავნოს აღნიშნული რიცხვები.

შევნიშნოთ, რომ მეორე შემთხვევაში აღნიშნული პრობლემის მოხსნასთან ერთად ფართოვდება სერვერის შესაძლებლობები: კლიენტს უკვე თვითონაც შეეძლება გადასამრავლებელი რიცხვები შეარჩევა.

ვქმნით სერვერის ახალ მოდულს. მოქმედებების რაოდენობა იქნება ორი:

პირველი მათგანი გააანალიზებს კლიენტის პასუხს (მისი PathInfo="/1"), ხოლო მეორე კი (მისთვის PathInfo="", Default=true) – შექმნის HTML დოკუმენტს.

მეორე ხდომილობის OnAction დამმუშავებელში ჩავწეროთ:

```
procedure TWebModule1.WebModule1WebActionItem2Action(
                    Sender: TObject; Request: TWebRequest;
                    Response: TWebResponse; var Handled: boolean);
var Num1, Num2: word;
begin
   Randomize;
  Num1 := Random(8)+1;
   Num2 := Random(8)+1;
  Response.Content:='<html><head><title> გამრავლების ტაბულა' +
   '</title></head><body>'
   '<h1>შეამოწმეთ თქვენი ცოდნა გამრავლების ტაბულაში </h1>';
Response.Content := Response.Content + 'ჩაწერეთ ფანჯარაში, რას' +
'უდრის ნამრავლი და დააჭირეთ „პასუხ“ ღილაკზე.'
Response.Content := Response.Content +
    '<form method="POST"  '  +
    'action="http://mycomputer/cgi-bin/mult.exe/1" <table>' +
    '<table><tr><td><input type="text" name="num1" size="1" ' +
    'value=IntToStr(Num1) +' ></td><td> x ' +
    '<td><input type="text" name="num2" size="1" value=' +
    IntToStr(num2) + '></td><td> = ' +
    '<td><input type="text" name="result" size="3"></td>' +
```

```
        '<td><input type="submit" name="post" value="პასუხი">' +
        '</td></tr></table></form><BR>' +
         '<form method="POST" ' +
         'action="http://mycomputer/cgi-bin/mult.exe"> ' +
         '<p><input type="submit" value="ახალი რიცხვები"  ' +
         'name="New"></p></form></body></html>';
  Handled:=true;
end;
```

Randomize ფუნქციის ამოქმედება აუცილებელია გამოყენების ყოველი გაშვებისას. Random ფუნქციის მეშვეობით Num1 და Num2 ცვლადებს ენიჭებათ შემთხვევითად არჩეული რიცხვების მნიშვნელობები 1 - 9 დიაპა-ზონიდან.

შემდეგ ხდება ზემოთ აღწერილი HTML-დოკუმენტის ფორმირება.

რაც შეეხება პირველი მოქმედებისათვის OnAction ხდომილობის დამმუშავებელს, მას შეიძლება ჰქონდეს ასეთი სახე:

```
procedure TWebmodule1.Webmodule1WebActionItem2Action(
                        Sender: TObject; Request: TWebRequest;
                        Response: TWebResponse; var Handled: boolean);
var Num1, Num2: word;
begin
   if Request.ContentFields.Values['result'] =
     IntToStr(StrToInt(Request.ContentFields.Values['num1']) *
     StrToInt(Request.ContentFields.Values['num2'] ) )
  then  Response.Content:=
       '<b>გილოცავთ!'  </b><p>  თქვენ ბრწყინვალე მათემატიკოსი
ბრძანდებით!'
   Else Response.Content :=
                    'მათემატიკა ცოტა დაგვიწყებიათ' <p>' +
                    '<a HREF=http://mycomputer/cgi-bin/mult.exe">'  +
                    '<p>კიდევ  სცადეთ </a></p>';
     Handled: true;
end;
```

შევინახოთ გამოყენება Mult სახელით (სწორედ ეს სახელი ფიგურირებს ზემოთ მოყვანილი კოდის დაყრდნობის ტეგებში), მოვახდინოთ მისი კომპილირება და შესრულებადი მოდული მოვათავსოთ სერვერის შესრულებად კატალოგში. შევამოწმოთ გამოყენება მუშაობაში – დასმულ შეკითხვაზე გავცეთ სწორი და არასწორი პასუხები.

HTML შაბლონების გამოყენება

ზემოთ განხილული გამოყენების მაგალითზე დავრწმუნდით, თუ რამდენად მოუხერხებელია OnAction პროცედურაში შედარებით რთული სახის HTML დოკუმენტის ფორმირება. გამოსავალი მდგომარეობს HTML-შაბლონების გამოყენებაში. ესენი გახლავთ სპეციალური ტეგები,

რომლებიც სერვერული გამოყენების მიერ შეიცვლება HTML-გამოსახულებების სასურველი ფრაგმენტებით.

მოვიყვანოთ ასეთი ტეგის მაგალითი:

<#TagName Param1=Value1 Param2=Value2 … >

აქ TagName შაბლონის სახელია, რომელიც შეიძლება შეირჩიოს მომხმარებელმა. ძირითადად ასეც ხდება, თუმცა მომხმარებელს შეუძლია ისარგებლოს სტანდარტული სახელის და ფუნქციის მქონე შაბლონებითაც, როგორიც არის, მაგალითად, Link (შაბლონი გამოიყენება კავშირების განსაზღვრისათვის), Image (აფიქსირებს გრაფიკულ ელემენტებს) და სხვ.

შაბლონების ჩასართავად გათვალისწინებულია Internet ჩანართზე განთავსებული PageProducer კომპონენტი. სწორედ ამ კომპონენტზე დაყრდნობით გადავწყვეტთ ზემოთ დასახულ ამოცანას.

ვქმნით ახალ სერვერულ მოდულს იმავე მოქმედებებით. პირველი მოქმედებისათვის უცვლელად გადმოგვაქვს კოდი OnAction-სათვის. რაც შეეხება მეორე მოქმედებას, მისთვის კოდის დაწერამდე WebModule1 სერვერის ფანჯარაში გადავიტანოთ PageProducer კომპონენტი. ჩავათვალიეროთ ამ კომპონენტის თვისებები ობიექტების ინსპექტორის ფანჯარაში. უპირველეს ყოვლისა, გვაინტერესებს მისი HTMLDoc და HTMLFile თვისებები. პირველს გააჩნია TStrings ტიპი და მასში ჩვეულებრივი საშუალებების გამოყენებით ხდება HTML-კოდის შეტანა. HTMLFile თვისებაში კი შეიძლება ნაჩვენები იქნეს *.html (*.htm) ფაილის სახელი (თუ უპირატესობას ვანიჭებთ *.html დოკუმენტის ცალკე ფაილად გაფორმების ხერხს).

ორივე მიდგომას გააჩნია დადებითი და უარყოფითი მხარეები.

*.html მონაცემების ცალკე ფაილად შენახვა იძლევა რამდენიმე პროგრამის მიერ მისი ერთდროული გამოყენების შესაძლებლობას. ამასთან, ადვილდება დოკუმენტის მოდერნიზაციაც. ნაკლი კი ისაა, რომ სერვერული გამოყენების მისამართის შეცვლისას უნდა შეიცვალოს დოკუმენტზე დაყრდნობებიც, რის გამოც საჭირო ხდება გამოყენების ხელახლა კომპილირება.

იმ შემთხვევაში, თუ ეს *.html მონაცემები მხოლოდ მოცემულ გამოყენებას სჭირდება და მათში არც მომავალში იგეგმება რაიმე ცვლილებების შეტანა (რაც იშვიათი მოვლენაა), მაშინ უპირატესობა ენიჭება პირველ მიდგომას.

PageProducer კომპონენტთან დაკავშირებულია ერთადერთი OnHTMLTag ხდომილობა. მას ადგილი მაშინ აქვს, როდესაც HTML-დოკუმენტის ტექსტში შაბლონი გამოიყენება. OnHTMLTag ხდომილობის დამმუშავებლის სათაურს ასეთი სახე აქვს:

```
procedure TWebModule1.PageProducer1HTMLTag (
    Sender: TObject;  Tag: TTag;  const TagString: String;
    TagParams: TStrings;   var ReplaceText: String);
```

Tag პარამეტრი შაბლონის ტიპს განსაზღვრავს, თუ შაბლონის სახელი (TagString პარამეტრი) ერთ-ერთი სტანდარტულად მიჩნეული სახის გახლავთ. წინააღმდეგ შემთხვევაში ამ ტიპს ანიჭებენ tgCustom

მნიშვნელობას. (შევნიშნოთ, რომ, უმეტესწილად, შაბლონისათვის არასტანდარტულ სახელს ირჩევენ).

tgParams-ით განისაზღვრება შაბლონის პარამეტრები (თუკი ეს საჭიროა).

ReplaceText პარამეტრში დამმუშავებელმა უნდა შეიტანოს შაბლონის შემცვლელი ტექსტი. თუ ტექსტი შეტანილი არ იქნა, შაბლონი ცარიელი სტრიქონით შეიცვლება.

ქვემოთ მოცემულია სხვადასხვა პარამეტრების მნიშვნელობათა ურთიერთშეთანადების ცხრილი:

Tag	TagString	TagParams	ReplaceText
tgLink	LINK	დაყრდნობა	<A>ელემენტში განთავსებული HTML ბრძანებები
tgImage	IMAGE	გრაფიკული გამოსახულების აღწერა	<IMG> ტეგი
tgTble	TABLE	ცხრილის ელემენტის აღწერა	<TABLE>ელემენტში განთავსებული HTML ბრძანებები
tgImageMap	IMAGEMAP	ცხელ უბანთან დაკავშირებული გრაფიკული გამოსახულება	<MAP>ელემენტში განთავსებული HTML ბრძანებები
tgObject	OBJECT	ActiveX ელემენტზე დაყრდნობა	<OBJECT>ელემენტში განთავსებული HTML ბრძანებები
tgEmbed	EMBED	Netscape ელემენტზე დაყრდნობა	<EMBED>ელემენტში განთავსებული HTML ბრძანებები
tgCustom	რაიმე სხვა ნებისმიერი სახელი	შეიძლება იყოს ცარიელი ან განისაზღვროს გამოყენების მიერ	გამოყენებით განსაზღვრული ბრძანებების ნებისმიერი მიმდევრობა

PageProducer კომპონენტის მიბმა სერვერული გამოყენების ამა თუ იმ მოქმედებასთან ორი გზით შეიძლება განხორციელდეს.

<u>პირველი</u> მათგანი გულისხმობს Producer თვისებაში მოდულში შეტანილი შესაბამისი PageProducer კომპონენტის სახელის ჩვენებას. ასეთ შემთხვევაში PageProducer კომპონენტით მოწოდებული HTML-დოკუმენტის ტექსტი Response.Content თვისების მეშვეობით გადაეცემა OnAction ხდომილობის დამმუშავებელს (ცვლილებების შესატანად). როდესაც დოკუმენტში ცვლილებების შეტანა საჭირო არ გახლავთ, მაშინ OnAction-ში არაფერს ვწერთ.

თითოეულ PageProducer კომპონენტი მოცემულ მომენტში Producer თვისებით მხოლოდ ერთი მოქმედებას შეიძლება მიებმეს. თუ აღმოჩნდა, რომ იგი რამდენიმე მოქმედებასთან არის მიბმული, გარდა პირველი

მოქმედებისა, ყველა სხვა მოქმედების Producer თვისებას იძულების წესით მიენიჭება nil მნიშვნელობა.

<u>განვიხილოთ მეორე გზა</u>. ამ შემთხვევაში PageProducer კომპონენტის OnAction ხდომილობის დამმუშავებელთან დაკავშირება ხდება ამ დამმუშავებელიდან PageProducer.Content მეთოდის გამოძახებით. ასეთი გამოძახებისას PageProducer კომპონენტი აანალიზებს დოკუმენტს და აგენერირებს OnHTMLTag ხდომილობას, რომლის დამმუშავებელშიც ახდენს შაბლონების შეცვლას.

ყოველივე ზემოთ თქმულის გათვალისწინებით, ხელახლა დავწეროთ სერვერული გამოყენების კოდი. როგორც უკვე აღვნიშნეთ, მოდულში ვითვალისწინებთ იმავე მოქმედებებს და ამავე დროს მასში ვამატებთ PageProducer კომპონენტს.

ძირითადი HTML-დოკუმენტის ტექსტი, რომელიც ადრე გენერირდებოდა მეორე მოქმედების დამმუშავებელში, შეიძლება შეტანილი იქნეს ცალკე ფაილში, ანდა PageProducer კომპონენტის HTMLDoc თვისებაში:

```
<HTML>
<HEAD>
<TITLE> გამრავლების ტაბულა </title>
</head>

<BODY>
<H1> შეამოწმეთ, რამდენად კარგად იცით გამრავლების ტაბულა!
</h1>

<P>სავარაუდო შედეგის ფანჯარაში შეტანის შემდეგ დააწკაპუნეთ
«პასუხი» ღილაკზე:
 </p>

<FORM method="POST" action="http://mycomputer/cgi-bin/mult1.exe/1" >
<TABLE>
 <TR><TD><input type="text" name="num1" size="1"
value="<#Tnum1>"></td>
        <TD> x </td>
        <TD><input type="text" name="num2" size="1"
value="<#Tnum2>"></td>
        <TD><input type="text" name="result" size="3" ></td>
        <TD><input type="submit" name="post" value="პასუხი"></td>
  </tr>
</table>
</form>

 <FORM method="POST" action="http://mycomputer/cgi-bin/mult1.exe">
   <P><input type="submit" name="New" value="ახალი რიცხვები"></p>
 </form>
```

```
</body>
</html>
```

ვხედავთ, რომ დაყრდნობებში მოდულის სახელი mult.exe შეცვლილია mult1.exe-ით (ეს გამოყენების ახალი ვარიანტის სახელია); num1 და num2 ელემენტების აღწერაში დაემატა value ატრიბუტები <#Tnum1> და <#Tnum2> მნიშვნელობებით (მოიცემა შაბლონების მიერ). თუ ტექსტი ცალკე ფაილშია განთავსებული, PageProducer კომპონენტის HTMLFile თვისებაში უნდა მიეთითოს მისი სახელი.

რედაქტორის კოდის ფანჯრის Preview ჩანართიდან შეიძლება წარმოდგენა მივიღოთ, თუ როგორი იქნება WEB-ფურცელზე ინფორმაციის განლაგება (თუმცა აქ შაბლონები თავდაპირველი სახით აისახება). პირველი მოქმედებისაგან განსხვავებით, მეორისათვის იცვლება OnAction ხდომილობის დამმუშავებლის კოდი – იგი მნიშვნელოვნად მარტივდება: Num1 და Num2 ცვლადების გამოცხადება გატანილია OnAction-ის ფარგლებს გარეთ. ეს ცვლადები გლობალური სახისაა, რათა მათ მნიშვნელობებზე გასვლა შეძლოს PageProducer1 კომპონენტის OnHTMLTag ხდომილობის დამმუშავებელმაც. ამ უკანასკნელის კოდს შეიძლება ასეთი სახე ჰქონდეს:

```
procedure TWebModule1.PageProduserHTMLTag(Sender: Pobject;
                                Tag: Ttag; const TagString: String;
                                 TagParams: Tstrings; var ReplaceText: String);
begin
  if (TagString = 'TNum1')
     then ReplaceText :=InToStr(Num1)
     else if (TagString ='TNum')
    then ReplaceText := IntToStr(Num2);
end;
```

მოცემულ ხდომილობის დამმუშავებელში ადრე HTML-ტექსტში მოყვანილი შაბლონები იცვლება Num1 და Num2 ცვლადების მნიშვნელობებით.

პროექტს ვინახავთ mult1 სახელით და ვაკომპილირებთ. წინა ვარიანტისაგან განსხვავებით, იდენტური დანიშნულების მქონე ამ გამოყენების კოდი გაცილებით მოკლე და გამჭვირვალეა. გარდა ამისა, თუ ჩვენ კოდს ცალკეული ფაილის სახით წარმოვადგენთ, მისი მოდერნიზება გამოყენების ხელახალი კომპილაციის გარეშეც მოხერხდება.

მონაცემთა ბაზებთან მუშაობა ინტერნეტში

<u>მონაცემთა ცხრილის ჩათვალიერება</u>

ვქმნით ახალ WEB-გამოყენებას. მასში ვითვალისწინებთ მხოლოდ ერთ მოქმედების ობიექტს, რომელსაც სახელად ვარქმევთ Default-ს და, ბუნებრივია, მის Default თვისებას ვანიჭებთ true მნიშვნელობას. გადაგვაქვს WEB-სერვერის ფანჯარაში Table და DataSetTableProducer კომპონენტები.

ჩვენთვის ნაცნობი გზით Table1 ცხრილური კომპონენტი დავაკავშიროთ SQL მონაცემთა ბაზის რომელიმე, მაგალითად, Pers ცხრილთან, ხოლო DataSetTableProducer კომპონენტი კი დავაკავშიროთ Table1-თან. ამ მიზნით, DataSetTableProducer კომპონენტის DataSet თვისებისათვის ავირჩიოთ Table1 მნიშვნელობა; ამასთან, Header თვისებაში შევიტანოთ ეკრანზე ცხრილის გამოყვანამდე საჭირო HTML-კოდის შემდეგი ფრაგმენტი:

```
<HTML>
<H1> საკადრო შემადგენლობა </h1>
<BODY>
```

შესაბამისად, Footer თვისებაში ჩაიწერება:

```
</body>
</html>
```

შევიტანოთ სათაური Caption თვისებაში და აქვე მნიშვნელობა განვუსაზღვროთ MaxRow თვისებასაც.

DataSetTableProducer კომპონენტის სხვა თვისებებისათვის მნიშვნელობების მინიჭება უფრო მოსახერხებელია სპეციალური რედაქტორიდან, რომელსაც გამოვიძახებთ კომპონენტის კონტექსტური მენიუდან ან მასზე ორჯერ დაწკაპუნებით.

რედაქტორის ფანჯარაში ვირჩევთ ეკრანზე ცხრილში გამოსაყვან ველებს Add New ან Add All Fields ღილაკის მეშვეობით. შევნიშნოთ, რომ Add All Fields ღილაკი მხოლოდ მაშინ არის ქმედუნარიანი, თუ მანამდე ვისარგებლეთ რედაქტორის მომსახურებით.

ცხრილში ყოველი ახალი ელემენტის შეტანის შემდეგ ობიექტების ინსპექტორში განვსაზღვრავთ მნიშვნელობებს მისი FieldName და Caption თვისებებისათვის.

ფანჯრის მარცხენა მხარეში შესაძლებელია თვისებები განესაზღვროს უშუალოდ ცხრილს.

ამის შემდეგ OnAction ხდომილობის დამმუშავებელში (იგი აქ ერთადერთია) შეგვაქვს ოპერატორები:

```
Table1.Active := true;
Response.Content := DataSetTableProducer1.Content;
Table.Active := false;
```

პროექტს შევინახავთ DB1 სახელით და შევასრულებთ ყველა სხვა საჭირო ქმედებას. შედეგად, ბროუზერში აისახება მონაცემთა ბაზის

Personal ცხრილიდან გადმოგზავნილი ინფორმაცია (მას აქაც ცხრილის სახე ექნება).

ჩვენ შეგვიძლია პროგრამაში ჩავრთოთ, კლიენტის მოთხოვნისდა მიხედვით, სხვადასხვაგვარად ინდექსირებული ცხრილის ეკრანზე გამოყვანის შესაძლებლობაც. აღნიშნული მიზანი მიიღწევა Table1 კომპონენტის IndexName თვისებისათვის შესაბამისი მნიშვნელობის მინიჭებით. შევიტანოთ DataSetTableProducer1 კომპონენტის header თვისებაში კოდი:

```
<HTML>
 <head>

 </head>

 <body>
<h1> საკადრო შემადგენლობა</h1>

<Table Width=”100%” Align= “Left” Border=1>
   <TR><TD>ბუღალტერია</TD> <TD> სანოძე  </TD>
        <TD>      სანოძე   </TD> <TD> სანოძე   </TD>
        <TD>      1985          </TD> <TD>  მდედრ   </TD>
         <TD>22</TD></TR>
  <TR><TD>ბუღალტერია</TD><TD>რამიშვილი</TD>
        <TD> რამიშვილი  </TD> <TD>რამიშვილი</TD>
        <TD>    1985          </TD><TD>    მდედრ   </TD>
        <TD>22</TD></TR>
  <TR><TD>ბუღალტერია</TD><TD>ქაჩლიშვილი</TD>
        <TD> ქაჩლიშვილი  </TD> <TD>  ქაჩლიშვილი </TD>
        <TD>    1985          </TD><TD>    მამრ   </TD>
        <TD>22</TD></TR>
</Table></body>
</html>
```

ამის შემდეგ უნდა მოხდეს ამ დაყრდნობებზე რეაგირების მომხდენი მოქმედების ფორმირება და მისი PathInfo თვისებისათვის “/index” მნიშვნელობის მინიჭება.

დაბოლოს, OnAction ხდომილობის დამმუშავებელში ჩავწერთ კოდს:

```
case StrToInt (recuest.Query) of
 1: Table.IndexName := ’fio’;
 2: Table.IndexName := ’depfio’;
 3: Table.IndexName := ’year’;
 4: Table.IndexName := ’ ’;
end;
Table1.Active := true;
Response.Content := DataSetTableProducer1.Content;
Table1.Active := false;
```

პროექტი შევინახოთ DB2 სახელით (ეს სახელია მითითებული Header-ში) და ჩავატაროთ სხვა საჭირო მანიპულაციებიც. ვნახავთ, რომ ბროუზერიდან შესაძლებელი გახდება მასში ცხრილის სხვადასხვა ველის მნიშვნელობების მიხედვით სორტირებული სახით გამოყვანა.

შესაძლებელია, აგრეთვე, კლიენტმა განსაზღვროს კრიტერიუმების სახე მონაცემების ფილტრაციისათვის და სხვ.

ცალკეულ ჩანაწერებთან მუშაობა

წინა პარაგრაფში გამოყენებულ DataSetTableProducer კომპონენტს არ შეეძლო ცხრილში აესახა memo და გრაფიკული ინფორმაციის შემცველი ველები.

ამ მიზნის განსახორციელებლად სხვა - DataSetPageProducer კომპონენტი გამოიყენება. იგი ბევრი რამით წააგავს ადრე განხილულ PageProducer კომპონენტს, რომელიც უზრუნველყოფდა HTML-შაბლონებთან მუშაობას. მაგრამ DataSetPageProducer კომპონენტი მონაცემთა კრებულთან კავშირდება DataSet თვისებით და იგი ავტომატურად ახდენს იმ შაბლონების შეცვლას, რომელთა სახელები ემთხვევა ველების სახელებს. შაბლონების ჩანაცვლება ხორციელდება მიმდინარე ჩანაწერში ველების არსებული მნიშვნელობებით.

ამრიგად, DataSetPageProducer კომპონენტის მეშვეობით შეიძლება არჩეულ იქნეს ცალკეული ჩანაწერების ველები, როგორც ტექსტური, ასევე memo და გრაფიკული გამოსახულებების შემცველნი.

კვლავ ვქმნით ახალ მოდულს და მასში გადაგვაქვს Table და DataSetTableProducer კომპონენტები. მონაცემთა ბაზის ცხრილთან მიბმას წინა შემთხვევის ანალოგიურად ვახდენთ. ოღონდ თავს ვიკავებთ მონაცემთა კრებულის ინდექსირებისაგან. ამ ქმედებას პროგრამულად განვახორციელებთ.

DataSetTableProducer კომპონენტს რედაქტორის მეშვეობით, წინა მოდულში გამოყენებულთა გარდა, დავუმატოთ კიდევ ერთი სვეტიც. მივმართოთ ობიექტების ინსპექტორს და Caption-ში ჩავწეროთ “დამატებითი-ინფ”, ხოლო FieldName-ში – “დამ-ინფ”. შევნიშნოთ, რომ ბოლო ქმედებაზე შეიძლება უარიც ვთქვათ.

მომხმარებლის მიერ ამორჩეული თანამშრომლის შესახებ შესახებ დამატებითი ინფორმაციის გასათვალისწინებლად მივმართავთ უჯრის შემცველობის გენერირების OnFormatCell ხდომილობის დამმუშავებელს.

ხდომილობის დამმუშავებლის სათაურს ექნება სახე:

```
procedure TWebModule1.DataSetTableProducer1FormatCell ()
   Sender: Tobject;   CellRow, CellColumn: integer;
var Bgcolor: THTMlBgcolor; var Align: THTMLAlign;
var Valign: THTMLAlign; var CustomAtts, CellData: string);
```

CellRow და CellColumn პარამეტრები უჯრედის კოორდინატებს განსაზღვრავენ. სტრიქონებისა და სვეტების ინდექსების ათვლა ხდება 0–დან. ამასთან, ყურადღება უნდა მიექცეს იმ ფაქტს, რომ ნულოვანი ინდექსი სათაურის სტრიქონს ეთმობა.

BGColor პარამეტრი უჯრედის ფონის ფერს განსაზღვრავს, Align და Valign – ჰორიზონტალსა და ვერტიკალში პოზიციებს.

CellData-ში შეიტანება ტექსტი ან HTML-ბრძანებები, CustomAttrs-ში კი დამატებითი ატრიბუტები აისახება.

ჩვენს შემთხვევაში OnFormatCell-ში ჩავწერთ:

```
procedure TWebModule.DataSetTableProducer1FormatCell (
  Sender : TObject ; CellRow, CellColumn: integer;
  var BgColor :THTMLBgColor ; var Align:THTMLAlign;
  var VAlign : THTMLAlign; var CustomAttrs, CellData: String );
begin
  if(CellRow>0) and (CellColumn=7)
       then CellData := '<a href= "/cgi-bin/db3.exe/record ? N=' +
   Table1Num.AsString +  ' ">ჩათვალიერება</a>' ;
end ;
```

ამ კოდს მერვე სვეტის (მისი ინდექსია 7) თითოეულ უჯრაში, გარდა სათაურის უჯრისა, შეაქვს დაყრდნობა ჩვენი გამოყენების (რომელსაც შემდგომ სახელად მიენიჭება «DB3») მოქმედებაზე. ამ მოქმედებას გადაეცემა პარამეტრი N სახელითა და ისეთი მნიშვნელობით, რომელიც ტოლია მონაცემთა ბაზის ცხრილში Num სახელის მქონე გასაღებური ველის მნიშვნელობის. სწორედ ამ პარამეტრის მეშვეობით ახდენს გამოყენება მოთხოვნაში მითითებული ჩანაწერის იდენტიფიცირებას. უჯრედში შეიტანება ტექსტი «ჩათვალიერება».

Default მოქმედების OnAction ხდომილობის დამმუშავებელში შეგვაქვს კოდი:

```
Table1.Active := false;
Table1.IndexName := 'depfio';
Table1.Active := true;
Response.Content := DataSetTableProducer1.Content ;
```

შედეგად ხდება ცხრილის ინდექსირება და უჯრედის შევსება.

დავასრულოთ გამოყენება და გავუშვათ შესრულებაზე. შევამჩნევთ, რომ «ჩათვალიერება» დაყრდნობებს ჯერ არსად გავყავართ.

WEB-მოდულში ჩავამატოთ მოქმედება, რომელსაც დაეკისრება სწორედ ეს ფუნქცია – ობიექტის შესახებ დამატებითი ინფორმაციის გამოყვანა. ვუწოდოთ ამ მოქმედებას "record" და PathName თვისებას მივანიჭოთ "/record". აღნიშნული მოქმედება უნდა ურთიერთობდეს DataSetPageProducer კომპონენტთან. გადავიტანოთ ეს კომპონენტი WEB-მოდულში და მის DataSet თვისებას მნიშვნელობად განვუსაზღვროთ Table1, HTMLDoc თვისებაში კი შევიტანოთ შემდეგი ტექსტი:

```
<HTML>
 <head>   </head>
 <body>
   <h1>თანამშრომლის ჩანაწერის ჩათვალიერება<h1>
   <TABLE width="100%" Align="Left">
     <TR><TD Align="Left"><b>გვარი</TD><TD><b>სახელი</TD>
          <TD><b>მამის სახელი</TD><TD><b>დაბადების თარიღი
         </TD><TD><b>სქესი</TD><TD><b>ასაკი</TD></TR>
  <TR><TD><#Fam></TD><TD><#Nam></TD><TD><#Par></TD>
        <TD><#Year_b></TD><TD><#Sex></TD><TD><#Age></TD></TR>
</TABLE>
<br><br><br><hr>
<TABLE width="100%">
        <TR><TD Align="Left"><b>daxasiaTeba</TD></TR>
        <TR><TD><#charact></TD><TD><#Photo></TD></TR></TABLE>
</body>
</html>
```

ეს ტექსტი აფორმირებს კადრების შემადგენლობის ამსახველ WEB-ფურცელს.

მოცემული ცხრილის HTML-ტექსტში შეტანილია შაბლონები შესაბამისი ველების სახელებით. აღნიშნული შაბლონები DataSetPageProducer კომპონენტის მიერ შეიცვლება მიმდინარე ჩანაწერის ველების მნიშვნელობებით.

record მოქმედების OnAction ხდომილობის დამმუშავებელში შეგვაქვს კოდი, რომელიც Locate მეთოდით განსაზღვრავს მიმდინარე ჩანაწერს:

```
Table1.Active := false;
Table1.Indexname:= ' ';
Table1.Active := true;
Table1.Locate('Nom', Request.QueryFields.Values['N'],[ ]);
Response.Content :=DataSetPageProducer1.Content;
```

მომდევნო ცხრილში განთავსდება თანამშრომლის დახასიათება და ფოტოსურათი. ჯერჯერობით მათ ნაცვლად HTML ტექსტში შეტანილია შაბლონები Charact და Photo სახელებით. ამ შაბლონების გაშიფრვა უნდა მოხდეს DataSetPageProducer კომპონენტის OnHTMLTag ხდომილობის დამმუშავებელში:

```
if (TagString =' ')
   then ReplaceText := Table1.FieldByName(' ').AsString
else if  (TagString =' Photo ')
   then ReplaceText := '< img src = photo?N = ' +
                          Table1.FieldByName('Num').AsString + ' > ';
```

Charact ველისაგან განსხვავებით, ეკრანზე ფოტოსურათის გამოტანას მეტი ძალისხმევა სჭირდება, რადგანაც დაუშვებელია HTML-დოკუმენტში გრაფიკული ველის პირდაპირი გზით გადაცემა. ამის გამო საჭირო ხდება HTML-დოკუმენტში <IMG> ტეგის ჩართვა, რომლიდანაც განხორციელდება

სერვერისადმი კიდევ ერთი მიმართვა. პასუხად კლიენტს გადმოეგზავნება ნაკადში განთავსებული ნახატი.

მაშასადამე, აუცილებელი ხდება სერვერის WEB-მოდულში ერთი მოქმედების დამატებაც. დავარქვათ მას Photo და PathInfo-ს მივანიჭოთ "/photo" მნიშვნელობა.

OnAction ხდომილობის დამმუშავებელში შევიტანოთ შემდეგი კოდი:

```
procedure TWebModule1.TWebModule1PhotoAction(Sender: TObject;
    Request: TWebRequest; Response; var Handled: Boolean);
var
  B: TBitMap;
S1: TMemoryStream;
begin
  Table1.Locate('Num',Request.QueryFields.Values['N'],[ ]);
  B := TBitMap.Create;
  B.Assign(Table1.FieldByName('Photo'));
  S1 := TMemoryStream.Create;
  B.SaveToStream(S1);
  S1.position := 0;
  Response.ContentType :='image/x - xbitmap';
  Response.ContentStream :=S1;
  B.Free;
end;
```

ეს კოდი ქმნის TBitMap ტიპის დროებითი სახის ობიექტს, რომელშიც დაფიქსირდება მიმდინარე ჩანაწერის Photo ველის შემცველობა.

შემდეგ იქმნება TMemoryStream ტიპის ნაკადის S1 ობიექტი. მასში შეიტანება B ობიექტის შემცველობა და ნაკადის პოზიციად განისაზღვრება მისი დასაწყისი, ხოლო დასაბრუნებელი შედეგისათვის კი - ტიპი (და ქვეტიპი):

```
Response.ContentType :=  'image/x-xbitmap';
```

აქ image ტიპია, ხოლო x-xbitmap – ქვეტიპი.

Response.ContentStream უნდა შეიცავდეს იმ ნაკადის მაჩვენებელს, რომლიდანაც აიღება ობიექტი.

დაბოლოს, ხდება დროებითი B ობიექტის განადგურება.

დავასრულოთ სერვერული გამოყენების შექმნა და შევამოწმოთ იგი მუშაობაში.

<u>მონაცემთა კრებულების რედაქტირება</u>

დავუშვათ, გვსურს ინტერნეტში გამოვაცხადოთ რაიმე ვაკანსიები და მსურველებს შევთავაზოთ, დარეგისტრირდნენ.

მოვიყვანოთ სარეგისტრაციო ფურცლის მაგალითი. მასში შეტანილი ინფორმაცია კლიენტის მიერ სერვერზე გადაიგზავნება და კორექტულობაზე შემოწმების გავლის შემდეგ განთავსდება მონაცემთა ბაზის Personal ცხრილში.

სარეგისტრაციო ფურცლისათვის დავწეროთ ასეთი HTML-კოდი:

```
<html>
  <head>
    <title>სამუშაოზე მიღება</title>
  </head>
<body>

<h1><b> თანამდედობაზე პრეტენდენტის რეგისტრაცია</h1>
<p>ჩაწერეთ თქვენს შესახებ ცნობები და დააჭირეთ
  ღილაკს "რეგისტრაცია"
</p>
<form method="POST" action = "DB4.exe/resp">
   <table>
      <tr>
        <td><strong> გვარი  </strong></td>
        <td><input type="text" name="fam" size="20"></td>
</tr>
 <tr>
        <td><strong>სახელი</strong></td>
        <td><input type="text" name="Nam" size="20"></td>
</tr>
<tr>
        <td><strong> მამის სახელი</strong></td>
        <td><input type="text" name="Par" size="20"></td>
</tr>
<tr>
        <td><strong> სქესი</strong></td>
        <td><input type="radio" value="mamr" checked name="sex">
                                <strong> მამრ  </strong>
        <td><input type="radio" value="mdedr" checked name="sex">
                                <strong> მდედრ  </strong></td></tr>
<tr>
        <td><strong> დაბადების თარიღი</strong> </td>
        <td><input type="text" name="Year" size="4"></td>
</tr>
<tr>
      <td></td>
      <td><div align= "center"><center><p><input type="submit"
            value="რეგისტრაცია" name="B1"></td>
  <tr>
</table>
</form>
</body>
</html>
```

ჩანს, რომ ფორმაში განთავსებულია ცხრილი, მასში კი – წარწერები და შეტანის კომპონენტები:

Fam, Nam, Par, Year (ერთსტრიქონიანი რედაქტირების ფანჯრებია);

B1 ღილაკით ხდება Post მეთოდის გამოძახება და DB4.exe გამოყენებისათვის ინფორმაციის გადაგზავნა („/resp“ გზით);

Sex კომპონენტი წარმოადგენს ორი რადიოღილაკისაგან შემდგარ ჯგუფს. მისგან DB4.exe გამოყენება ღებულობს ინფორმაციას პრეტენდენტის სქესის შესახებ.

მომზადებული Web-ფურცელი შეიძლება უშუალოდ განვათავსოთ სერვერზე და მუშაობა დავიწყოთ მისი გამოძახებით. მაგრამ უკვე ვიცით, რომ უმჯობესია, ფურცელი ჩავტვირთოთ სერვერული გამოყენებიდან.

დავაპროექტოთ ეს სერვერული გამოყენება. შევქმნათ Web-სერვერის ახალი მოდული და გადავიტანოთ მასზე Table და PageProducer კომპონენტები. წინა გამოყენებების მსგავსად, აქაც Table1 კომპონენტი მონაცემთა ბაზის Pers ცხრილთან დავაკავშიროთ.

შემდეგ, PageProducer1 კომპონენტის HTMLDoc თვისებაში შევიტანოთ Web-ფურცლის შესაქმნელად ზემოთ მომზადებული HTML-ტექსტი. შევქმნათ მოქმედება და მისი Default თვისება დავაყენოთ true-ში. რადგანაც ამ მაგალითში HTML-ტექსტი შაბლონებს არ შეიცავს, აღარ გვჭირდება OnAction-დან PageProducer1-ის გამოძახება. და, საერთოდაც, ეს დამმუშავებელი საჭირო აღარ გახლავთ. საკმარისია, მოცემული მოქმედების Producer თვისებაში მივუთითოთ PageProducer1 კომპონენტი.

აქვე აღვნიშნოთ, რომ ზემოთ მოყვანილ HTML-ტექსტში ნაჩვენებია „/resp“ გზაც.

შევქმნათ მოქმედების მოდული და დავარქვათ მას Response. მასთან დაკავშირებულ OnAction ხდომილობის დამმუშავებელში შეგვაქვს კოდი, რომელიც ამოწმებს სერვერისათვის გადაგზავნილი ინფორმაციის სისწორეს, შეცდომების აღმოჩენის შემთხვევებში კლიენტს უგზავნის შესაბამის შეტყობინებებს, ხოლო თუ ყველაფერი წესრიგშია, მონაცემები შეაქვს ბაზაში.

ხდომილობის დამმუშავებელი შეიძლება ასეთი სახის იყოს:

```
procedure TWebModuleResponseAction (
            Sender: TObject; Request: TWebRequest;
             Response: TWebResponse; var Handled; Boolean);
var year: word;
begin
  if (Request.ContentFields.Values['Fam'] = ' ') or
     (Request.ContentFields.Values['Nam'] = ' ') or
     (Request.ContentFields.Values['Par'] = ' ') or
     (Request.ContentFields.Values['Year'] = ' ')
then bagin
     Response.Content :=
          '<p>სარეგისტრაციო ბლანკი არასრულად არის შევსებული.' +
          '<p> ხელმეორედ შეიტანეთ მონაცემები.'
               Handlend := true;
```

```
end
else begin
  try
  year := StrToInt(ContentFields.Values[' Year']);
except
Response.Content := '<p>დაბადების თარიღის ფორმატში შეცდომაა.' +
                        ' <p>ხელმეორედ შეიტანეთ მონაცემები.'
   Handled := true;
   exit;
end;
if (year < 1917) or (year > 1980)
then begin
Rersponse.Content := '<p>სამწუხაროდ, თქვენი ასაკი არ შეესაბამება
  ამ თანამდებობას' ;
    Handled := true;
     exit;
end;
Table1.Active := true ;
Table1.Insert ;
Table1Fam.AsString := Request.ContentFields.Values['Fam'] ;
Table1Fam.AsString := Request.ContentFields.Values['Nam'] ;
Table1Fam.AsString := Request.ContentFields.Values['Par'] ;
Table1Year_b.AsInteger := year ;
Table1Sex.Value := ContentFields.Values['Sex'] ;;
// ახალი ჩანაწერების გადაგზავნა მონაცემთა ბაზაში
Table1.Post;
Table1.Active := false;
Response.Content :=
'<p> გმადლობთ! <p>მონაცემები თქვენს შესახებ შეტანილ იქნა ბაზაში' +
'<p>ჩვენ განვიხილავთ თქვენს კანდიდატურას და შეგატყობინებთ შედეგს';
   Handled := true ;
end;
end;
```

თავდაპირველად, ხდება შემოწმება, შევსებულია თუ არა ყველა ველი. ხარვეზების აღმოჩენის შემთხვევაში მომხმარებელს ეცნობება ნაკლის შესახებ და მას ეძლევა საშუალება, ხელახლა შეავსოს ანკეტა. ამასთან, შესაძლებელია წინა ფურცელზე დაბრუნება და მხოლოდ ხარვეზების აღმოფხვრა მასზე.

შემდეგი ოპერატორი ამოწმებს Year ველის ფორმატის სისწორეს - იგი უნდა შეიცავდეს მხოლოდ მთელ რიცხვებს. მოწმდება აგრეთვე, ჯდება თუ არა ეს რიცხვი წინასწარ განსაზღვრულ დიაპაზონში ...

პროექტს შევინახავთ DB4 სახელით, ჩავატარებთ ყველა სხვა საჭირო მანიპულაციას და შევამოწმებთ მუშაობაში.

ზემოთ განხილულ თითოეულ მაგალითში ინტერნეტში განთავსებულ მონაცემთა ბაზებთან დასაკავშირებლად ვიყენებდით Table კომპონენტს. ცხადია, შესაძლებელია ვისარგებლოთ Query კომპონენტითაც. მხოლოდ ამ შემთხვევაში ფორმაში გადაგვაქვს არა DataSetTableProducer, არამედ QueryTableProducer კომპონენტი. მისი Query თვისება უნდა შეიცავდეს დაყრდნობას TQuery ტიპის მონაცემთა კრებულზე. სხვა მხრივ, ამ კომპონენტების თვისებები ერთმანეთისაგან არ განსხვავდება.

ყველაფერი ის, რაც აქამდე განვიხილეთ, მიეკუთვნებოდა ინტერნეტისათვის გამოყენებების დამუშავების ე.წ. Broker ტექნოლოგიას. მისი არსი შემდგომში მდგომარეობს:

Web Broker ტექნოლოგიაში ძირითადი სამუშაოების შესრულება ეკისრება სერვერს. იგი კლიენტს HTTP ოქმის მეშვეობით უგზავნის HTML ფურცელს. მომხმარებელი კლიენტური გამოყენების - ბროუზერის დახმარებით ჩაათვალიერებს ამ ფურცელს, გარკვეული მანიპულაციების ჩატარებით აფორმირებს მორიგ მოთხოვნას სერვერისადმი, ანდა დაყრდნობის მეშვეობით გადადის სხვა Web-ფურცელზე. პროცესს ციკლური ხასიათი აქვს. სერვერს შეუძლია ერთდროულად რამდენიმე კლიენტთან იმუშაოს.

WebSnap ტექნოლოგია

ინტერნეტში მუშაობისათვის, ზემოთ აღწერილ WEBSnap ტექნოლოგიის გარდა, იყენებენ რიგ სხვა ტექნოლოგიებსაც.

გავეცნოთ WEBSnap ტექნოლოგიას, რომლისთვისაც საფუძველს წარმოადგენს იგივე Web Broker ტექნოლოგია. შედეგად, WEBSnap-ს შეუძლია შეასრულოს ყველა ზემოთ აღწერილი ფუნქცია. მაგრამ WEBSnap ტექნოლოგია რიგ დამატებით საშუალებებსაც ფლობს, კერძოდ:

WEBSnap ტექნოლოგიაში დასაშვებია სერვერის მხარეზე სკრიპტების გამოყენებაც. მის ღირსებებში შედის, აგრეთვე, გამოყენების მრავალ-მოდულიანობა, კონკრეტული მომხმარებლის შესახებ ინფორმაციის დაფიქსირების შესაძლებლობა (რაც პაროლურ დაცვას აადვილებს), პროგრამირების ვიზუალიზაციის უფრო მაღალი ხარისხი და სხვ.

WebSnap სერვერი

Web Broker გამოყენების მსგავსად, WebSnap გამოყენებაც შედგება .exe ან .dll ტიპის შესრულებადი ფაილისა და HTML ფურცლების შაბლონების რიგი ფაილებისაგან. ასე რომ, ფურცლებში ცვლილებების შეტანა შესაძლებელია შესრულებადი ფაილის ხელახალი კომპილირების გარეშე.

როგორც აღვნიშნეთ, WebSnap გამოყენება შეიძლება მოიცავდეს რამდენიმე მოდულს. ძირითადი გახლავთ TWebAppPageModule-ის, ანუ Web-გამოყენების ტიპის მოდული. მის გარდა, გამოყენებაში შეიძლება ჩართული იქნეს TWebPageModule ტიპის Web-ფურცლის რამდენიმე მოდულიც. დასაშვებია აგრეთვე მონაცემების მომწოდებელი შემდეგი მოდულების შექმნაც:

TWebAppDateModule - გამოყენებისათვის;
TWebDateModule - Web-ფურცლისათვის (რამდენიმეც შეიძლება იყოს).

აღვნიშნოთ, რომ ამ მოდულებისათვის განკუთვნილია სხვადასხვა სახის კომპონენტები. ისინი განთავსებულია WebSnap ჩანართზე.

WebSnap გამოყენების შესაქმნელად გავცეთ ბრძანება:

File → New → Other

(შესაძლებელია ასევე ღილაკების ნებისმიერი პანელის კონტექსტურ მენიუში Internet ინდიკატორის მონიშვნა).

ავირჩიოთ WebSnap Application პიქტოგრამა.

გამოდის ფანჯარა, რომელშიც ვაყენებთ:

- ტიპისათვის CGI stand-alone executable-ს;
- ცოტა ქვემოთ ვირჩევთ Page Module ვარიანტს;
- პანელის ქვედა ნაწილში სახელს ვარქმევთ Web-ფურცელს (მაგალითად, Page1-ს).
- აქვე შესაძლებელია კეშირების ვარიანტის არჩევაც (ასეთ შემთხვევაში მოდული სეანსებს შორის შეინახება სერვერის მეხსიერებაში, რაც განაპირობებს დროში ეკონომიას, თუმცა, ცხადია, იწვევს მეხსიერების ხარჯს).

Page Options ღილაკზე დაწკაპუნებით გამოგვყავს შემდგომი დიალოგური ფანჯარა. ვახდენთ ასეთ არჩევანს:

- ფურცლის კომპონენტისათვის – Page Producer;
- სკრიპტისათვის – Jscript ;
- მოვნიშნავთ New File ინდიკატორს;
- შაბლონისათვის – Standard;
- სათაურის Title ველში ვწერთ – “საწყისი ფურცელი”;
- მოვნიშნავთ Published ინდიკატორს მოთხოვნისას ფურცლის ავტომატურად ასარჩევად.

OK და სერვერის პროექტი შექმნილია. ეკრანზე აისახება სერვერის მოდულის ფორმა მასზე დატანილი ყველა აუცილებელი კომპონენტით. რადგანაც ამ კომპონენტების თვისებები ავტომატურად გაიწყობა, დეტალებში გარკვევა არც ისე გადაუდებელი საქმეა. ერთადერთი, რაც სასურველია, გახლავთ Application Adapter კომპონენტის ApplicationTitle თვისებისათვის უფრო შინაარსობრივი სახელის მინიჭება.

ახლა კი პროექტში ჩავრთოთ მოდული მონაცემებისათვის. Internet ინსტრუმენტული პანელიდან გადმოვიტანოთ WebSnap DataModule.

გამოდის პატარა დიალოგური ფანჯარა Caching და Creation სიებით. კეშირებასთან დაკავშირებით არჩევანი უკვე გაკეთებული გვაქვს. რაც შეეხება Creation სიას, On Demand-ის არჩევისას მოდული შეიქმნება მისი გამოძახების მომენტში, ხოლო Always მას შექმნის გამოყენების შესრულების დაწყებისთანავე. ნებისმიერ ვარიანტში ეკრანზე აისახება ახლახან შექმნილი მონაცემთა მოდულის ფანჯარა. მონაცემთა კრებულად შევარჩიოთ Table კომპონენტი.

Table-ს ვაკავშირებთ მონაცემთა ბაზის Personal ცხრილთან, ველების რედაქტორიდან შემოგვყავს ყველა ველი, ვირჩევთ სასურველ ინდექსს და Active თვისებას ვანიჭებთ true მნიშვნელობას. შევნიშნოთ, რომ ყველა ეს ქმედება ჩვენთვის კარგად ნაცნობია. მაგრამ არის ერთი თავისებურებაც: იმ მიზნით, რომ WebSnap სერვერმა ავტომატურად შეძლოს მონაცემთა კრებულზე საჭირო ოპერაციების (მაგალითად, ნავიგაციის ოპერაციების) ჩატარება, ცხრილის Num პირველადი გასაღებური ველისათვის ProviderFlags თვისებაში უნდა ჩავრთოთ pfInKey ალამი.

რადგანაც Table კომპონენტი BDE-ს სფეროდან არის, აუცილებელი ხდება მონაცემების მოდულში Session კომპონენტის დამატებაც, რომლის AutoSessionName თვისებას მნიშვნელობად უნდა განვუსაზღვროთ true. დავინახავთ, რომ Session და Table კომპონენტების SessionName თვისებაში ჩაჯდება ავტომატურად გენერირებული სესიის სახელი.

WebSnap ჩანართიდან მოდულში ჩავამატოთ DataSetAdapter ადაპ-ტერი. მის DataSet თვისებაში დავაყენოთ Table1. შემდეგ საჭიროა ვაჩვენოთ Web-ფურცლის სკრიპტებში გამოყენებული ელემენტები (აქ ადაპტერის ველები). ორჯერ ვაწკაპუნებთ Adapter1 კომპონენტზე. გამოსულ ფანჯარაში New Item ღილაკზე დაწკაპუნებებით შეიძლება ადაპტერის ველების სხვადასხვა ტიპების შემოყვანა და მათი თვისებების განსაზღვრა, მაგრამ ჯერჯერობით ჩვენ უფრო მარტივი ამოცანა გვაქვს გადასაწყვეტი: საჭიროა მხოლოდ ადაპტერის ველების ობიექტების შექმნა, რომლებიც შესატყვისობაში იქნებიან მონაცემთა ცხრილის ველებთან. ამ მიზნით, კონტექსტური მენიუდან გავცემთ Add All Fields ბრძანებას. შედეგი აისახება Data თვისებაში.

ამჯერად დავაწკაპუნოთ Adapter კომპონენტის Actions თვისების გვერდით მყოფ მრავალწერტილიან ღილაკზე და გამოსულ ფანჯარაში გამოვიძახოთ კონტექსტური მენიუ. ავირჩიოთ ბრძანება Add All Actions. შედეგად საშუალება მოგვეცემა ავსახოთ და რედაქტირება გავუკეთოთ ყველა ველს, მათ შორის გრაფიკულებსაც.

პროექტი შევინახოთ WebSnapData სახელით. აღვნიშნოთ, რომ მთავარი მოდულის სახელი შეიძლება ნებისმიერი იყოს. რაც შეეხება მონაცემთა მოდულს, პრინციპში შესაძლებელია მისთვისაც ნებისმიერი სახელი ავირჩიოთ (მაგალითად, UData), მაგრამ უნდა გავითვალისწინოთ, რომ შემდგომში მოგვიწევს ზუსტად ამ სახელის მქონე მოდულზე დაყრდნობა.

ჩავრთოთ პროექტში ფურცელი, რომელშიც აისახება მონაცემთა ცხრილი. Internet პანელში ავირჩიოთ WebSnap PageModule. ფურცლის სახელად (Name თვისება) ავირჩიოთ “Page2”, სათაურად (Title თვისება) – “ფურცლის ჩათვალიერება”, ხოლო პროდიუსერად – AdapterPageProducer.

ეკრანზე აისახება ფურცლის მოდულის ფანჯარა, რომელიც შეიცავს ერთადერთ – AdapterPageProducer კომპონენტს. იგი უნდა დავაკავშიროთ ადრე შექმნილ მონაცემთა მოდულთან. გავცემთ ბრძანებას:

File → Use Unit → UData

ახლა უკვე შესაძლებელია AdapterPageProducer ადაპტერის გაწყობა. ორჯერ ვაწკაპუნებთ მასზე. გამოდის ფანჯარა, რომელშიც ხეზე თავდაპირველად ერთადერთი – AdapterPageProducer მწვერვალია. ვაწკაპუნებთ New item ღილაკზე. გამოდის დიალოგური ფანჯარა, რომელშიც ვირჩევთ ახალი კომპონენტის ტიპს – AdapterForm-ს. ხეზე მოვნიშნავთ ამ ახალ კომპონენტს და ანალოგიური წესით დავამატებთ მასზე AdapterGrid-ს, რომლისთვისაც განვსაზღვრავთ თვისებების მნიშვნელობებს.