

ClientDataSet – I deo

Nikola Radivojević

Trenutno živimo u svetu gde je već skoro svaka pomisao na informacioni sistem vezana za mrežu.

Trenutno živimo u svetu gde je već skoro svaka pomisao na informacioni sistem vezana za mrežu, ako ne globalnu, onda barem lokalnu, tako da se polako, ali sigurno i sistemi za upravljenje bazama podataka orijentišu na tu stranu i gube se tragovi desktop (lokalnim) bazama podataka. Takav razvoj na sceni informacionih sistema naveo je i kompaniju

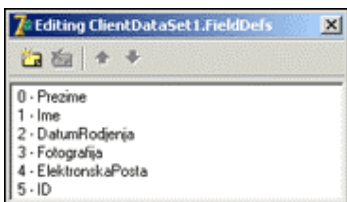
Borland da se orijentiše u pravcu velikih mreža i Interneta, a zatvori projekat koji je godinama razvijan (BDE – Borland Database Engine). Sa izlaskom distribucije Delphi 6, odnosno Kylix 1 proizvoda uvedena je nova tehnologija, DBExpress, koju i sam Borland preporučuje za razvoj novih sistema, a BDE je ostavljen za podršku ranijih projekata. Iako ne naročito omiljena platforma za rad sa bazama podataka, BDE je imao mnoge dobre strane i ogromne mogućnosti i olakšice u razvoju aplikacija koje su se oslanjale na lokalne baze podataka (Dbase, Paradox, Access, FoxPro...). Kao alternativu za rad sa lokalnim bazama Borland je ponudio “MyBase” koja se ogleda u komponenti ClientDataSet.

Cilj ovoga članka će biti da se ukratko opišu bitne karakteristike navedene komponente i način primene i mogućnosti u korišćenju lokalnih podataka i komponentu ClientDataSet. Biće pokazan način kreiranja strukture podataka, pregled, pretraga i indeksiranja podataka, dok će u sledećim člancima biti objašnjeni načini ograničavanja prikaza podataka, pretrage po određenim kriterijumima, postavljanja svojstva agregacija, mogućnosti kloniranja kursora podataka i opisane prednosti korišćenja ove komponente u analiti podataka u distribuiranim informacionim sistemima.

Kreiranje lokalnih tabela

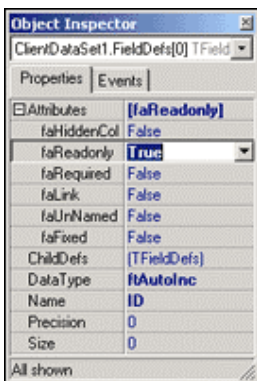
Podaci u okviru ClientDataSet komponente se čuvaju u memoriji, a njima se pristupa preko interfesja ove komponente koji ima kompatibilnosti sa standardizovanim operacijama nad bazama podataka. Bitna razlika između ovakvog pristupa i ostalih tehnologija je u brzini. Pošto se ovde podaci drže u memoriji računara, rad sa njima se odlikuje velikom brzinom. Na osnovu date brzine u radu bilo je moguće komponentu proširiti sa mnogim dodatnim metodama i osobinama, a da se ne primeti pad performansi u radu.

U martovskom članku Omega magazina, upotrebio sam ClientDataSet koja je preko komponente DataSetProvider bila povezana sa SQLDataSet komponentom iz koje je crpela podatke. U ovome slučaju DataSetProvider daje meta podatke ClientDataSet komponenti i omogućava na taj način formiranje struktura u memoriji gde će se čuvati podaci. Pošto u ovoj primeni nema DataSetProvider-a potrebnu strukturu podataka ćemo morati sami da napravimo. ClientDataSet podatke o strukturi drži preko kolekcije FieldDefs. Recimo da želimo napraviti tabelu koja će nam omogućavati rad sa podacima o osobama, i da su nam potrebni podaci prezime, ime, datum rođenja, pol osobe, fotografija, adresa stanovanja i elektronska pošta. Da bismo definisali ovu strukturu podataka za vreme projektovanja aplikacije trebamo: otvoriti novi projekat za izradu aplikacije, dodati komponentu ClientDataSet na formu i dodati u kolekciju FieldDefs komponente ClientDataSet potrebne definicije, sa željim atributima tako da naša kolekcija na kraju izgleda kao na slici 1.



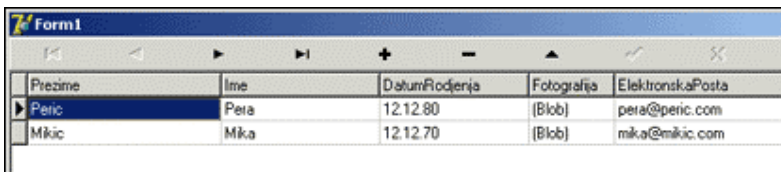
Slika 1: Kolekcija FieldDefs

Da bismo završili definisanje meta podataka moramo dodeliti i potrebne atribute datim poljima, tako da polje ID bude tipa ftAutoInc, polja Prezime i Ime budu tipa ftString sa dužinom 30, DatumRodjenja tipa ftDateTime, Fotografija tipa ftBlob, a elektronska pošta tipa ftString sa dužinom 250. Da bi nam polje autoinc stvarno bio jedinstveni broj koji korisnik neće moći da menja možemo postaviti atribut faReadOnly u ObjectInspector-u, kao što je prikazano na slici 2.



Slika 2: Postavljanje atributa

Kada smo završili definisanje željene strukture podataka, potrebno je napraviti DataSet. Ovo se može za vreme projektovanja odraditi na preko pomoćnog menija na komponenti ClientDataSet, izborom opcije Create DataSet. Sada je ClientDataSet u mogućnost da pruži našoj aplikaciji podršku za rad sa ovim podacima. Za probu stavite na formu DataSource i DBGrid i DBNavigator komponente i povežite ih (DataSource.DataSet = ClientDataSet, DBGrid.DataSource = DataSource i DBNavigator.DataSource = DataSource). Pokrenite aplikaciju i videćete da preko komponente DBGrid možete upravljati podacima, kao što se vidi na slici 3.



Slika 3: Primer aplikacije

Sada ćemo dodati mogućnost aplikaciji da učita podatke sa lokalnog diska prilikom startovanja i da po potrebi snimi podatke na hard disk. Da biste snimili strukturu podataka za vreme projektovanja aplikacije iz pomoćnog menija komponente ClientDataSet izaberite jednu od opcija koja počinje sa Save To. Postoje 3 različita formata u koje se mogu snimati podaci i učitati iz datoteke. To su binarni format, xml format i xml-utf8. Izaberite format xml-utf8 i snimite



datoteku sa nazivom aplikacije i ekstenzijom xml u direktorijumu gde vam se nalazi i program. Zatim dodajte jedno dugme na formu i dodajte sledeći kod:

```
procedure TForm1.bSaveClick(Sender: TObject);
begin
  ClientDataSet1.SaveToFile(ClientDataSet1.FileName);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  with ClientDataSet1 do
    SaveToFile(FileName);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  with ClientDataSet1 do begin
    FileName := ChangeFileExt(Application.ExeName, '.xml');
    LoadFromFile(FileName);
  end;
end;
```

Sada već imamo aplikaciju koja je potpuno funkcionalna, ali postoji i sigurniji način. Zamislite da se nekako ošteti datoteka sa podacima, naša aplikacija neće više raditi. Ili da ako želite da distribuirate dalje aplikaciju morali bi da delimo aplikaciju sa praznim podacima. Mnogo je bolje rešenje dinamički kreirati strukturu podataka korišćenjem FieldDefs osobine, tako da ako se problem sa učitavanjem datoteke javi, možemo kreirati novu praznu datoteku. Dodajte sledeću proceduru u vaš kod,

```
procedure TForm1.CreateFieldDefs;
begin
  with ClientDataSet1.FieldDefs do
  begin
    Clear;
    with AddFieldDef do
    begin
      Name := 'ID';
      DataType := ftAutoInc;
      Attributes := [faReadOnly];
    end; //with AddFieldDef do
```



```
Add('Prezime', ftString, 30);  
Add('Ime', ftString, 30);  
Add('DatumRodjenja', ftDateTime);  
Add('Fotografija', ftBlob);  
Add('ElektronskaPosta', ftString, 250)  
end;  
ClientDataSet1.CreateDataSet;  
end;
```

i izmenite sledeću proceduru da izgleda ovako:

```
procedure TForm1.FormCreate(Sender: TObject);  
begin  
  with ClientDataSet1 do begin  
    FileName := ChangeFileExt(Application.ExeName, '.xml');  
    try  
      LoadFromFile(FileName);  
    except  
      if FileExists(FileName) then begin  
        ShowMessage('Doslo je do greske u datoteci!!!' + #13 + #10 + 'Podaci su  
          sacuvani u .back datoteci');  
        RenameFile(FileName, ChangeFileExt(FileName, '.back'));  
      end;  
      CreateFieldDefs;  
      SaveToFile(FileName);  
    end;  
  end;  
end;
```

Sada imamo potpuno funkcionalu aplikaciju. Sami možete implementirati dodavalje fotografije u istoimeno polje isto kao što biste uradili da je bilo koja druga baza u pitanju i bilo koja druga komponenta koja radi sa tabelama baze podataka.

Navigacija, pretraživanje i promena podataka

Kada posmatramo bilo koju bazu podataka, često se javlja potreba da se izvrši neka kalkulacija programski, gde će biti potrebno proći preko određenog skupa podataka, bilo da ćemo vršiti prolazak kroz sve podatke ili tražiti određeni podatak i promeniti određene podatke prema potrebi. U prethodnom primeru je upotrebljena komponenta DBGrid za prikaz, navigaciju i promenu podataka, što pri dovoljnoj količini podataka koji su ne sortirani i nije neka naročita olakšica. ClientDataSet vam pruža sledeće mogućnosti u navigaciji podataka: First, Prior, Next, Last, MoveBy, i postavljanje vrednosti u osobinu RecNo, tako da možete dodati 3 komponente Button i jednu Edit komponentu i dodati sledeći kod:



```
procedure TForm1.bNext10Click(Sender: TObject);
begin
  ClientDataSet1.MoveBy(10);
end;
```

```
procedure TForm1.bPrev10Click(Sender: TObject);
begin
  ClientDataSet1.MoveBy(-10);
end;
```

```
procedure TForm1.bRecNoClick(Sender: TObject);
begin
  try
    ClientDataSet1.RecNo := StrToInt(eRecNo.Text);
  except
    ShowMessage('Illegal RecNO');
  end;
end;
```

Ostale metode se pozivaju analogno prikazanom, ali su već ugrađene u DBNavigator, tako da ih nećemo posebno implementirati. Zamislite sada situaciju da želite da se prikazana elektronska pošta formatira tako da podaci ne mogu sadržati velika slova. Nezgodno je kod tabela sa velikom brojem zapisa vršiti promene podataka prelaženjem preko svih zapisa kada su komponente za prikaz podataka vezane za njih, jer prikazivanje podataka često može trajati mnogo duže od samo lociranja podataka i njihove izmene. Dodajte sada na formi jednu komponentu Button i sledeći kod:

```
procedure TForm1.bFormatEPClick(Sender: TObject);
begin
  with ClientDataSet1 do
    try
      DisableControls;
      if not ClientDataSet1.Active then ClientDataSet1.Open;
      First;
      while not ClientDataSet1.EOF do
        begin
          Edit;
          FieldByName('ElektronskaPosta').AsString :=
            LowerCase(FieldByName('ElektronskaPosta').AsString);
        end;
      Next;
    end;
end;
```



```
Post;  
Next;  
end;  
finally  
  EnableControls;  
end;  
end;
```

Komandama `DisableControls` i `EnableControls` omogućavamo da se operacije koje dugo traju izvršavaju brzo i bez prikazivanja u kontrolama, tako da operacija u našem slučaju traje veoma kratko i nad velikim brojem podataka. U ovome primeru smo koristili pretragu od prvoga ka zadnjem zapisu, a isto tako možemo koristiti i obrnuti redosled, gde bi kod izgledao:

```
...  
ClientDataSet1.Last;  
with
```

Recimo da ste završili aplikaciju i da želite da je podelite svojim prijateljima, ali biste isto želeli da im ostavite i svoje podatke. To možete uraditi u staru upisujući samo svoje podatke u xml datoteku, ali pošto smo napravili da se tabela sama genriše u slučaju loše datoteke, bilo bi zgodno da i automatski ubacimo naše podatke. Ovo možemo uraditi na sledeći način

```
Insert; //ili Append  
  FieldByName('Prezime').asString := 'Mikic';  
...  
Post;
```

a postoji i alternativa tako da možemo ceo zapis ubaciti odjednom korišćenjem naredbe `InsertRecord` ili `AppendRecord`. Dodajte sledeći kod u program:

```
procedure TForm1.InsertMyData;  
begin  
  ClientDataSet1.InsertRecord(['Mikic', 'Mika', EncodeDate(1970,12,12), null,  
    'Mika@Mikic.com', 'Ostali']);  
end;
```

Obratite pažnju da vrednost za polje ID nije stavljena u `InsertRecord` metodi. Ovo sam namerno izostavio iz razloga što smo dodelili atribut `faReadOnly` polju ID, tako da operacija ne bi uspela u slučaju da smo prosledili neku vrednost. Metodi `InsertRecord` možemo prosledi onoliko vrednosti koliko želimo, sa time što će se te vrednosti upisivati redom kako su definisane u `FieldDefs` osobini, tako da morate voditi računa prilikom definisanja polja, kao i o tome da, ako postoje određeni atributi koji su definisani u `FieldDefs` osobini, budu zadovoljni.

Rad sa indeksima

Kao i kod svih komponenti koje su izvedene iz `TDataSet` klase postoje indeksi. Indeksi su bitni za rad sa podacima, jer direktno utiču na redosled prikazivanja podataka, olakšavaju pretrage,

određivanje ograničenja, povezivanja DataSet naslednika, kao i otvaranje drugih mogućnosti koje programeri mogu koristiti u zavisnosti od potreba.

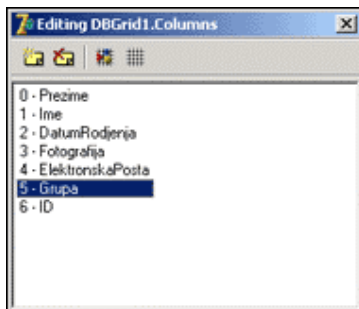
Nastavićemo sa doradom našeg malog primera. Dodajte sada u FieldDefs kolekciji još jedan podatak. Nazovite ga “Grupa”, postavite mu tip ftString i ostavite predefinisano dužinu 20. Dodajte i proceduri CreateFieldDefs sledeću liniju:

```
Add('Grupa', ftString, 20);
```

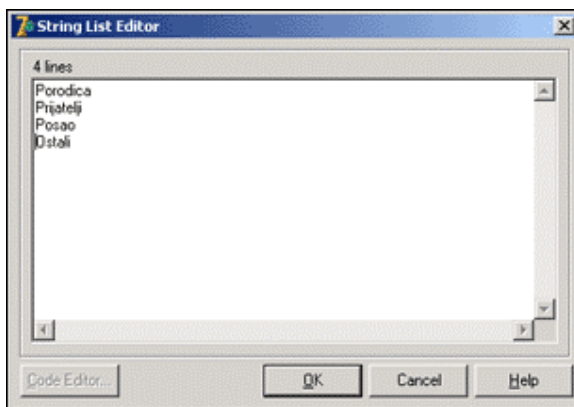
odmah ispred definisanja polja “ID”. Izmenite liniju u proceduri InsertMyData, tako da izgleda:

```
ClientDataSet1.InsertRecord(['Mikic', 'Mika', EncodeDate(1970,12,12), null,
'Mika@Mikic.com', 'Ostali']);
```

Iz pomoćnog menija komponente DBGrid izaberite opciju *Columns Editor*. Otvoriće vam se kolekcija za definisanje kolona koje će se prikazivati u ovoj komponenti. Dodajte sve kolone klikom na dugme *Add All Fields*, tako da vam kolekcija izgleda kao na slici 4. Zatim za kolonu *Grupa* promenite osobinu *PickList*. Dodajte stavke, tako da izgledaju kao na slici 5.



Slika 4: Kolekcija Columns



Slika 5: PickList Editor

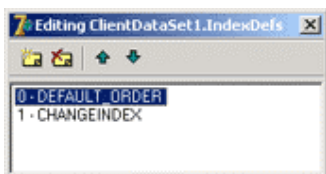
Proverite da li je osobina *ButtonStyle* za kolonu *Grupa* ima vrednost *chs.Auto*. Kompajlirajte sada program i pokrenite ga. Videćete da je dodata kolona *Grupa* na komponenti *DBGrid*.

Sada pretpostavimo da biste iz određenih razloga želeli da vam se podaci u aplikaciji sortiraju prema koloni *Grupa*. Postavite na komponenti *ClientDataSet* vrednost osobine *IndexFieldNames* da bude “Grupa”. Ako opet pokrenete aplikaciju videćete da su vam podaci sortirani prema datoj

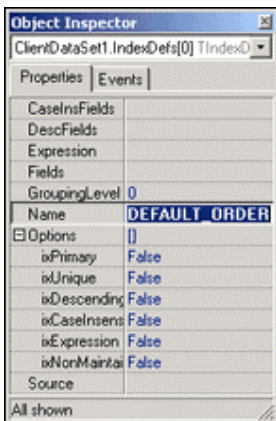
koloni. Možda biste sada želeli da možete sortirati podatke i prema drugim kolonama. Ovo je jednostavno uraditi. Dodajte na komponentu *DBGrid* događaj *OnTitleClick* i dodajte sledeći kod:

```
procedure TForm1.DBGrid1TitleClick(Column: TColumn);
begin
    (Column.Grid.DataSource.DataSet as TClientDataSet).IndexFieldNames :=
    Column.FieldName;
end;
```

Pokrenite aplikaciju videće promene. No ako biste poželeti da sortirate u opadajućem redsledu, ta neće moći da uradite na ovaj način. Da bi ovo bilo moguće mormo malo oozbiljnije da se pozabavimo indeksima. Iz pomoćnog menija na *ClientDataSet* komponenti odaberite *Create DataSet*. Pogledajte sada osobinu *IndexName*. Videće da postoje dva indeksa *CHANGEINDEX* i *DEFAULT_ORDER*. Otvorite sada kolekciju *IndexDefs* i pogledajte osobine koje može imati indeks. Dobićete prikaz kao na slikama 6 i 7. Sada možete dodati koje god indekse želite i možete im postaviti osobine kako želite. Sve što treba kasnije da uradite jeste da osobini *IndexName* na komponenti *ClientDataSet* postavite vrednost na ime vašeg indeksa.



Slika 6: Kolekcija *IndexDefs*



Slika 7: Osobine Indeksa

Bitno je istaći su osobine *IndexFieldNames* i vrednosti *IndexName* međusobno isključive, tj da postavljanje jedne isključuje postojenje druge vrednosti. Bitno je još spomenuti da kada snimate podatke u neku od datoteka se ne snimaju podešavanja za indekse, tako da ih uvek morate posebno postaviti. Kao što se struktura podataka može kreirati za vreme izvršavanja aplikacije, takoo mogu i indeksi. Ma koliko god indeksa kreirali nećete narušiti performanse aplikacije, jer se *ClientDataSet* fizički pravi indekse tek kada se oni trebaju upotrebiti. Dodajte sada sledeći kod

```
procedure TForm1.CreateIndexDefs;
begin
```



```
with ClientDataSet1 do
begin
  with IndexDefs.AddIndexDef do
  begin
    Name := 'IdxGrupaDesc';
    Fields := 'Grupa';
    Options := [ixDescending];
  end;
  IndexName := 'IdxGrupaDesc';
end;
end;
```

i dodajte poziv ove procedure kao zadnju naredbu u proceduri *FormCreate*. Videćete da je primenjen vaš indeks i da su podaci sortirani po koloni *Grupa* i to u opadajućem redu. Sada kada kliknete na bilo koje zaglavlja nad kontrolom DBGrid videćete da se poništava definisan indeks i da se podaci sortiraju u rastućem redosledu prema izabranoj koloni. Sada možete sami izmeniti proceduru *DBGrid1TitleClick* tako da se dinamički menja vaš definisan indeks, i da se ako je već vrši indeksiranje po datoj koloni da promeni redsled indeksiranja (iz rastućeg u opadajući i obrnuto).

Pregled

Prikazane su osnove rada sa komponentom kao što je ClientDataSet. Opisana mogućnost kreiranja lokalne baze, objašnjen način navigacije kroz podatke, pretrage i indeksiranja podataka. Objasnjeno je kako se može za vreme projektovanja postaviti i ograničiti aplikacija, kao i kako se može za vreme izvršavanja mogu odraditi navedene operacije. U sledećim člancima biće objašnjeno kako da ograničite prikaz podataka, pronađete podatke po odgovarajućim kriterijumima, kako da iskoristite mogućnosti agregacije koje pruža ova komponenta, rad sa kursorima podataka i mogućnost njihovog kloniranja i kako da sve ove mogućnosti na pravilan način iskoristite u jednom klijent server okruženju.