Baze podataka u Visual C#

UVOD

Uvod

‘U ovoj lekciji bavimo se bazama podataka u okviru Visual C#. Videćemo šta je to disconnected model baze podataka, DataForm, i šta je ADO.NET i DataGrid.

Ključna pitanja:

Šta je i kako se koristi DataForm?

Šta je to ADO.NET?

Objasniti disconnected model baze podataka u Visual C#?

Napomena: Pogledati kurs o bazama podataka na Digiš-u.

Baze podataka u Visual C#

01

BUILT-IN DATABASE ENGINE

Baze podataka se kreiraju pomoću specijalizovanih programa, ***database managers****.*

Visual C# je izvanredan za rukovanje podacima. U Visual C# ima ugradjena ***database engine****,* alat za kreiranje baze podataka, a takodje Visual C# uključuje drajvere za različite servere baza podataka. Ovaj ugradjeni alat ***built-in database engine*** (***MSDE Database engine***) se nalazi u okviru svih verzija Visual Studio-a, i to je u stvari jedna modifikovana verzija ***SQL Server***-a (***a cut-down edition of SQL Server, SQL Server Light***). Podaci tj. fajlovi memorisani pomoću Visual C# mogu biti korišćeni od strane drugih aplikacija npr. Microsoft Access, itd. (**.mdb, .mdf** fajlovi).

Cilj nam je ovde da proučimo kako pomoću Visual C# da kreiramo aplikacije sa bazama podataka. Takodje, i da upravljamo vezama sa bazama podataka, u cilju pretrage baze podataka i pisanja izveštaja. Ali, cilj nam je ovde da damo samo uvod u rad sa bazama podataka u Visual C#, dakle kreiranje nekoliko jednostavnih aplikacija sa bazama podataka, a za razvoj složenijih aplikacija sa bazama podataka nema prostora u okviru ovog predmeta, i potrebno je konsultovati druge knjige, i pažljivo i detaljno proučiti Visual Studio dokumentaciju.

Šta je to baza podataka? To je kolekcija podataka tj skup informacija, i čak neka Excel-tabela ili Word-tabela ili Word-lista predstavlja pojednostavljenu bazu podataka, ali takve jednostavne liste i tabele su nefleksibilne i teške za rukovanje ako su velikog obima, pa velike baze podataka se kreiraju pomoću specijalizovanih programa, ***database managers***. Visual C# omogućuje lako 1)kreiranje, i 2)rukovanje malih i velikih baza podataka. Naime, pomoću Visual C# mogu se kreirati C# forme za 1)pretragu, 2)ažuriranje podataka, ili 3)izveštavanje o podacima. 4)Takodje, može se koristiti programski kod u C#, da se obave proračuni, ili procesiraju veliki skupovi podataka. Šta je to baza podataka? U stvari, baza podataka, *database*, je kolekcija tj skup tabela koje su povezane na neki način, npr. tabela klijenata i tabela narudžbi, medjutim neke baze podataka sadrže samo jednu tabelu. Da se potsetimo terminologije koja se koristi medju programerima baza podataka. ***SQL, Structured Query Language***,to je jezik koji se koristi za pretraživanje i ažuriranje podataka, i potebno je njegovo znanje kod pravljenja *database applications*. Red tj zapis, *row* tj *record*, je jedan red u tabeli. Neka tabela , ***table,*** je lista podataka tj lista informacija organizovana po poljima tj kolonama.

MICROSOFT ACCESS I SQL SERVER

Microsoft Access *(tj skraćeno* ***Access****) je deo* ***Microsoft Office Profesional****, koji omogućuje pravljenje baza podataka, i on se ne nalazi u okviru Visual Studio.*

Najbolji način da naučite kako se prave baze podataka, je da sami napravite jednu bazu podataka. Visual Studio je izvanredan alat za pravljenje aplikacija sa bazama podataka, ali je to programski alat, dakle alat koji traži programiranje. U ovom predavanju ćemo pokazati kako da uradimo jednostavnu aplikaciju sa bazom podataka. Dve su opcije, za pravljenje takve aplikacije, ili da napravite potpuno novu bazu podataka, ili da koristite za to neku već postojeću bazu podataka. U sledećem poglavlju, napravićemo jednu novu ali jednostavnu bazu podataka, u cilju ilustracije, i daljeg korišćenja u okviru Visual C#.

Microsoft Access (tj skraćeno ***Access***) je deo ***Microsoft Office Profesional***, koji omogućuje pravljenje baza podataka, i on se ne nalazi u okviru Visual Studio. Umesto *Access*-a, može se koristiti neki drugi ***database manager***, npr. SQL Server, itd. ***Microsoft Access*** je je namenjena pravljenju jednostavnijih aplikacija koje zahtevaju bazu podataka. ***Access*** je namenjen običnim korisnicima a ne programerima, ali je vrlo popularan, koriste ga milioni korisnika, i ima neke prednosti u odnosu na profesionalnije alate, npr. u odnosu na ***SQL Server.*** *SQL* server je u okviru *C# Express*-a, i on je namenjen za upotrebu sa drugim proizvodima, i ne poseduje sopstvene alate za rad sa formama. U okviru C# Express-a nalaze se alati za SQL Server, npr. *Database Explorer*.

Visual C# i Access: Visual C je nešto komplikovaniji za rukovanje sa bazama podataka od **Microsoft Access**, ali ima neke prednosti u odnosu na ***Microsoft Access***, npr ***Microsoft Access*** se koristi samo za baze podataka, dok ***Visual C#*** ima niz ostalih razvojnih mogućnosti. Medjutim, ***Visual C#*** i ***Microsoft Access*** odlično kooperišu. Npr pomoću ***Access*** može se napraviti baza podataka, a zatim da se pomoću ***Visual C#*** da se uradi upakovana aplikacija. **Microsoft Access** je koristan eksterni, ***standalone***, pomoćni alat za Visual C#. Sve verzije ***Visual Studio***-a imaju ugradjenu ***MSDE Database engine*** (Microsoft Database engine), a to je redukovana verzija ***SQL Server***-a.

TABELA

Evo jednostavne tabele.

Tabela:

CustomerID Name

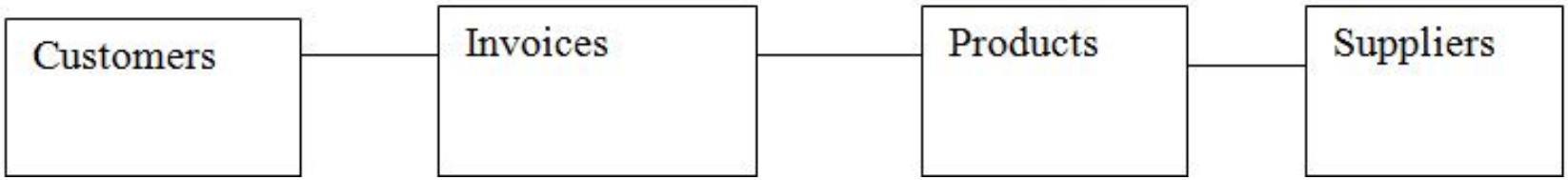
1 Popovic

2 Jovanovic

3 Despotovic abela:

BAZA PODATAKA **COFFEESQL.MDF** KAO KOLEKCIJA TABELA

*Na slici je baza podataka **CoffeeSQL.mdf** kao kolekcija tabela napravljena pomoću SQL Server-a*



Sl1: baza podataka

BAZE PODATAKA U VISUAL C#

*Pomoću Visual C# može se ostvariti veza sa nekom postojećom bazom podataka, tj. fajlom napravljenim sa alatom za pravljenje baze podataka (npr. ***.mdb**, ili ***.mdf**).*

U Visual C# ima niz „kontrola“ za rad sa bazama podataka. Pomoću Visual C# može se ostvariti veza sa nekom postojećom bazom podataka, tj. fajlom napravljenim sa alatom za pravljenje baze podataka (npr. ***.mdb**, ili ***.mdf**). Zatim, Visual C# omogućuje da se podaci iz baza podataka prikažu na „formi“, a takodje ove „kontrole“ omogućuju da se podaci u bazi podataka edituju tj. ažuriraju. Takodje, pomoću ***Visual C#*** mogu se praviti nove baze podataka, uz pomoć ugradjenog ***SQL Server***-a (***SQL Server Light)***.

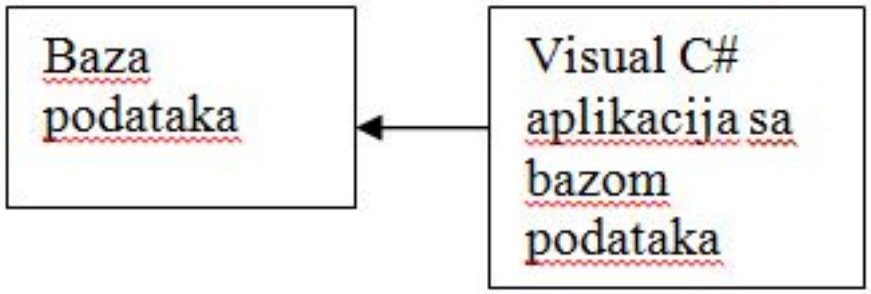
Visual C#

- može da rukuje sa maim i velikim bzama podataka
- omogućuje pravljenje Windows formi za pretragu, i za ažuriranje baze podataka
- i kreiranje izveštaja (*reports*) baza podataka (npr. prikazati listu članova sportskog kluba)
- takodje omogućuje da se obavljaju proračuni ili procesiranje podataka u okviru baza podataka

Visual C# omogućuje povezivanje na fajlove baza podataka napravljene pomoću Access-a (*.mdb fajlovi), i na fajlove SQL Server baza\ podataka (*.mdf fajlovi). ***Access*** je baziran na fajl-serveru (dovlače se svi podaci preko mreže, i onda obradjuju se ovi podaci lokalno), a ***SQL Server*** je baziran na klijent-serveru (baza podataka se čuva na serveru, njihova obrada se izvodi na serveru, a preko mreže se dovlače samo neophodni podaci).

VEZA IZMEDJU VISUAL C# I BAZE PODATAKA

Na slici se vidi veza izmedju Visual C# i baze podataka.



Sl2: C# aplikacija sa bazom podataka

.MDB FORMAT

*Ako nemate **Microsoft Access,** možete: Da nabavite neku bazu podataka u **.mdb** formatu (to je format koji koristi Microsoft Access), npr. Northwind.mdb, ili Nwind.mdb.*

Northwind Traders je fiktivna kompanija koja prodaje neke proizvode. U bazi podataka Northwind ima niz tabela sa informacijama o proizvodima ove fiktivne kompanije, o kupcima proizvoda, o narudžbinama kupaca, dobavljačima, isporučiocima, zaposlenima u kompaniji, itd. Npr. tabele Orders i Products mogu biti izabrane za ilustraciju rada sa bazom podataka pomoću Visual C#.

Može da se kreira baza podataka pomoću ***Microsoft Access database manager***-a. Ako nemate ***Microsoft Access,*** možete:

Da nabavite neku bazu podataka u **.mdb** formatu (to je format koji koristi *Microsoft Access*), npr. Northwind.mdb, ili Nwind.mdb, koja dolazi kao ugledni primer u mnogim *Microsoft* aplikacijama

Ili da koristite neki drugi *database manager*, npr. SQL Server, ili Visual FoxPro,dBase, ili Lotus Approach

Bilo koja baza podataka koja ima **ODBC** (***Open Database Connectivity***) drajver može se koristiti pomoću Visual C#

RELACIONE BAZE PODATAKA:

RBP su najrasprostranjeniji princip skladištenja podataka danas.

Relacione baze podataka: (RBP) koriste povezane tabele, npr. neka firma može da ima razdvojene tabele za Klijente, Narudžbe, i proizvode, itd. Ovakve **relacione baze podataka** su moćnije i fleksibilnije, ali komplikovanije da se rukuje sa njima, u odnosu na baze podataka sa jednom tabelom, kao npr. naš primer: Sportskiklub.MDB. Dakle, relacione baze podataka su zasnovane na tabelama, a tabele se sastoje od kolona i redova, a izmedju tabela postoje ustanovljene relacije koje omogućuju ostvarivanje veza izmedju podataka, uklanjanje redudantnosti, itd. RBP su najrasprostranjeniji princip skladištenja podataka danas.

KLIJENT-SERVERSKI SISTEMI BAZA PODATAKA:

*Pomoću **ASP.NET** mogu se praviti **Visual C# Web** aplikacije sa bazama podataka.*

Klijent-serverski sistemi baza podataka: koji imaju odnos ***Client-Server***, npr. *SQL Server*, MySQL, ili **Oracle**, zahtevaju složeno programiranje baza podataka, ali su neophodni za velike mreže, sa 10 ili više korisnika. Za razliku od **SQL Server**-a, ***Access*** nema odnos ***Client-Server,*** već fajl-server odnos**. ADO.NET** podržava niz raznih tehnologija, i ***Visual C#*** obezbedjuje skup drajvera pomoću kojih se može pristupiti raznim tehnologijama baza podataka, npr. *SQL Server*, MySQL, ili **Oracle**, itd.

ASP.NET (povezivanje na **Web): ASP.NET** omogućuje da se neka aplikacija koja je razvijena u C# nadogradi tako da omogućuje njenu upotrebu preko Interneta. Dakle, pomoću ***ASP.NET*** mogu se praviti ***Visual C# Web*** aplikacije sa bazama podataka. Prema tome, ASP.NET je ***framework*** unutar .NET-a, namenjen razvoju Web aplikacija, uključujući Web aplikacije sa bazama podataka (ali tada se koristi i ***ADO.NET)***.

Cilj ovog predavanja je samo da uvede studenta u oblast programiranja baza podataka pomoću Visual C#, ali pošto je ovo posebna i velika oblast, polje izvan opsega ovog predmeta, nećemo dublje ulaziti u ovu oblast. Za dalji rad treba proučiti knjige u ovom poljui i dokumentaciju Visual Studio.

Kreiranje baze podataka pomoću *Access*-a

02

KREIRANJA BAZE PODATAKA .MDB POMOĆU *ACCESS*-A

*Čak iako niste familijarni sa **Access**-om, u donjem primeru ćete moći da se snadjete, i da ponovite sličan primer sa drugim alatom,npr. **SQL Server**-om.*

Sada ćemo ilustrovati kako da kreiramo bazu podataka pomoću ***Microsoft Access***. Medutim, na potpuno sličan način može se napraviti baza podataka pomoću ***SQL Server***-a. Ovde ćemo ilustrovati primer baze podataka za upravljanje sportskim klubom. Medjutim, taj primer je lako adaptirati da važi za neki video klub, ili adresar, itd. Čak iako niste familijarni sa ***Access***-om, u donjem primeru ćete moći da se snadjete, i da ponovite sličan primer sa drugim alatom,npr. ***SQL Server***-om. Ako želimo da koristimo neku već posatojeću bazu podataka, onda nam nije potrebna ova baza podataka koju kreiramo u ovom poglavlju.

Evo primera kreiranja baze podataka Sportskiklub.MDB pomoću *Access*-a:

1. Startovati Access i izabrati da se kreira nova baza podataka, prazna baza podataka. Ići na ***File Selection dialog>,*** *pa* navigovati na direktorijum C:\mydatabases, I otkucati ime:

SportskiKlub.MDB

2. Kada se otvori baza podataka, izabrati opciju da se napravi tabela u ***Design view***. Otvara se tada ***Table Designer***.

DIZAJNIRANJE TABELE U ACCESS-U:

Ovde ćemo ilustrovati dizajniranje tabela pomoću access-a.

Ovde ćemo ilustrovati kako se definiše struktura tabele, i ovaj važan korak odredjuje koja vrsta podataka se može memorisati u tabeli:

1. Da bi dodali u tabeli polje, ***Field***, treba otkucati ime polja (***field name***) u prvom redu tabele, npr. **Prezime**.

2. Proveriti tip podatka, ***data type***, da li je u redu, npr TEXT,

3. Onda, u donjem delu Table Designer-a editovati ***field properties***, veličinu polja, ***field size***, i Zahtevanost, ***Required*** (Yes/no), npr.

Field size 30

Required YES

4. Dodati ostala polja, npr Ime, Adresa, Telefon, Email, Lični broj, kao što je prikazano u donjoj tabeli:

Tabela

NAME		TYPE		SIZE	REQUIRED
Prezime	Text		30	yes	
Ime		Text		30	no
Adresa		Text		50	no
Telefon	Text		30	no	
Email		Text		30	no
LicniBroj	AutoNumber	n/a	yes		

DIZAJNIRANJE TABELE U ACCESS-U:

U tabeli se mogu dodavati novi zapisi, naime jednostavno se klikne u praznom redu i ukucavaju se novi podaci.

5. Zatim, selektovati polje Prezime i zadati ***Indexed property***: Yes>Duplicates OK. Takodje, pomoću miša izabrati polje LicniBroj, i zadati da bude tzv. primarni ključ (***Primary Key***), naime samo kliknuti ikonu ***Key*** kada ste izabrali prethodno polje LicniBroj.

6. Kliknuti ***Close*** da bi se memorisala tabela, i zadati ime ovoj tabeli, npr **ClanoviKluba**.

7. Ime ove tabele će se pojaviti u **Access**-prozoru, i treba je onda duplo kliknuti da bi se ona otvorila. I ona bi izgledala kao što je dole ilustrovano, pod uslovom da se unesu podaci za nekoliko redova, npr. 4 ili 5 redova:

Prezime	Ime	Adresa	Telefon	Email	LicniBroj
Petrovic.....					
Simic					
Lukic					
Divac.....					

8. U tabeli se mogu dodavati novi zapisi, naime jednostavno se klikne u praznom redu i ukucavaju se novi podaci. U prethodnim koracima smo koristili ***Access***, ali ubuduće nećemo više koristiti Access već samo tabelu napravljenu pomoću *Access*, a umesto *Access* ćemo koristiti ***Visual Studio*** tj ***Visual C#.***

TABELE U BAZAMA PODATAKA:

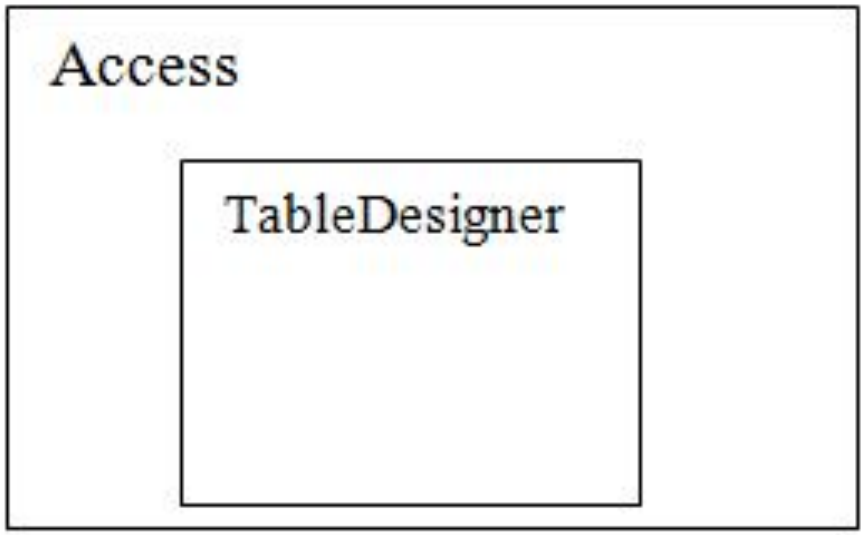
*Jedna od svojstava (properties) polja bila je osobina **Required,** koja se može zadati kao Yes ili No, i govori da li je obavezno dati taj podatak ili ne.*

Koristeći ***Microsoft Access*** upoznali smo se sa nekim bitnim osobinama tabela u bazama podataka. Npr. ***field type***, tip polja, nam govori da polja u tabeli imaju tip podataka, slično kao što se za varijable u C# koristi tip podataka. Tipovi polja koji se najćešće koriste su: **Text** – niz znakova do 255 znakova, **Integer** – ceo broj, **Long** – dugi ceo broj, **Memo** – niz znakova do 65355 znakova, **Boolean** – True ili False tj 1 ili 0, **Single** – decimalni broj, **Double** – veliki decimalni broj sa više decimala, **Currency** – podaci za valute, **Date/time** – podaci za datum I vremetabela u bazama podataka .Takodje, jedna od svojstava (*properties*) polja bila je osobina ***Required,*** koja se može zadati kao *Yes* ili *No*, i govori da li je obavezno dati taj podatak ili ne. Polje LicniBroj je izabrano kao ***Primary Key***, i kao ***Unique***, dakle ovo polje omogućuje pouzdanu i jednoznačnu identifikaciju nekog reda u tabeli.

Ovde navedene osobine se odnose na tabele u MDB formatu, npr. ***SportskiKlub.MDB***. A to je format koji koristi ***Microsoft Access***. Pomoću ***Visual Studio*** mogu se koristiti ovakve tabele, ali isto tako i tabele dobijene pomoću FoxPro, ili SQL Server, itd.

TABLE DESIGNER

Slika ilustrujeTable Designer.



Sl1: table designer

DataForm, i *DataForm Wizard*

03

DATAFORM, I DATAFORM WIZARD

*Pomoću **Data Form Wizard,** može se izvršiti povezivanje sa nekom postojećom bazom podataka, i takodje može se kreirati forma **DataForm.***

Data Form Wizard je „dijalog-boks“ u Visual Studio-u, pomoću koga može da se kreira ***DataForm***-a, dakle jedna forma koja je specijalno predvidjena za rad sa bazom podataka. Pomoću **Data Form Wizard,** može se izvršiti povezivanje sa nekom postojećom bazom podataka, i takodje može se kreirati forma **DataForm** (specijalno kreirana forma koja je zgodna za rad sa bazama podataka, i koja se brzo formira pomoću ***Data Form Wizard***), i posle ta aplikacija omogućuje prikazivanje podataka i rad sa podacima pomoću te forme.

Medjutim ako nemate *Data Form Wizard* u vašoj verziji Visual C#, ne morate da koristite *Data Form Wizard*, onda se umesto toga može koristiti npr. ***Data Grid***. **Dakle,** pored ***Data Form***-e, u Visual Studio postoji „kontrola“ **DataGrid.** Takodje, napomenimo da u Visual C# Express postoji „kontrola“ ***DataGridView***, koja je slična sa kontrolom ***DataGrid***.

DATAFORM-PRIMER

Evo primera kreiranja dataform-a.

Pogledajmo ilustraciju upotrebe ***Data Form Wizard*****,** za povezivanje sa bazom podataka, i za kreiranje odgovarajuće forme, forme ***DataForm1***, tj. jednostavne aplikacije sa bazom podataka:

1. Startovati novi projekt u Visual C#. Kada se projekat otvori, zatim izabrati **File>Add New Item**, i onda ***Data Form Wizard*** (izbor izmedju Windows Form, Class, Data Set, itd.). Prihvatiti predefinisano (*default)* ime ***DataForm1.cs*** koje se pojavljuje na dijalog-boksu, ii kliknuti Open.

2. Kliknuti Next, i onda vas ***Data Form Wizard*** pita da izaberete postojeći ili kreirate novi ***dataset.*** Izabrati ‘***’Create a new data set named’’***, i ukucati: **dsSportskiKlub,** i kliknuti **Next.**

3. Onda ste zapitani da izaberete ***Connection***, i vi kliknete ***New Connection***. Time se otvara dijalog ***Data LinkProperties***.

4. U ovom dijalogu ***Data LinkProperties*** kliknuti dugme ***Provider*** (***Data provider***), a onda kliknuti ***Microsoft Jet 4.0 OLE DB Provider (***izabrati iz čitave liste ***data provider-a)***, i onda OK. I time smo rekli da ćemo koristiti podatke dobijene pomoću Data provider-a od *Microsoft Access*-a, a to je ***Microsoft Jet 4.0 OLE DB Provider*** .

5. Kliknuti u dijalogu ***Data LinkProperties*** na dugme ***Connection***, a onda ***Select or enter a database name***, i navigovati na bazu podataka **SportskiKlub.MDB** koja je prethodno napravljena pomoću ***Access*****-a,** tj. fajl **SportskiKlub.MDB** koji je prethodno kreiran pomoću *Microsoft Access*-a.

Onda kliknuti ***Test Connection***, I treba da se pojavi ***Test Connection Succeeded***, I onda kliknuti OK da bi zatvorili dijalog ***Data LinkProperties***.

6. Time smo se vratili u ***Data Form Wizard***, i kliknuti onda Next. Pa će se tada pojaviti **Choose tables or views,** pomoću kojeg možemo da odredimo šta će biti prikazano na našoj ***DataForm***-i. Kliknuti tabelu **ClanoviKluba**. I onda otići na desno pomoću kliktanja strelice u sredini dijaloga, i onda kliknuti *Next*.

DATAFORM-PRIMER

Evo primera kreiranja dataform-a.

7. U ovom koraku se biraju polja tj. kolone koje će biti prikazane na formi DataForm. U tom cilju čekirati tj potvrditi sva polja (polja: **Prezime, Ime, Adresa**, ID **itd.**), i onda kliknuti ***Next***.

8. Sada je potrebno izabrati ***Display style***, tj stil prikazivanja podataka, Npr izabrati ***Single record***, tj jedan zapis, ili alternativno izabrati ***Grid*** (***All records in a grid***) ako želimo da vidimo odjednom listu zapisa a ne samo jedan zapis. A ostale opcije štiklirati tj čekirati da se prihvataju, naime, štilikrati da na **DataForm** dodaju dugmad:

Cancel all, Add, Delete, Cancel, Navigation controls.

Onda kliknuti ***Finish.***

9. Otvoriti programski kod za glavnu formu, koja je predefinisana po imenu **Form1.cs**, i u glavnoj metodi (metoda ***Main()***) staviti ***DataForm1*** umesto ***Form1***.

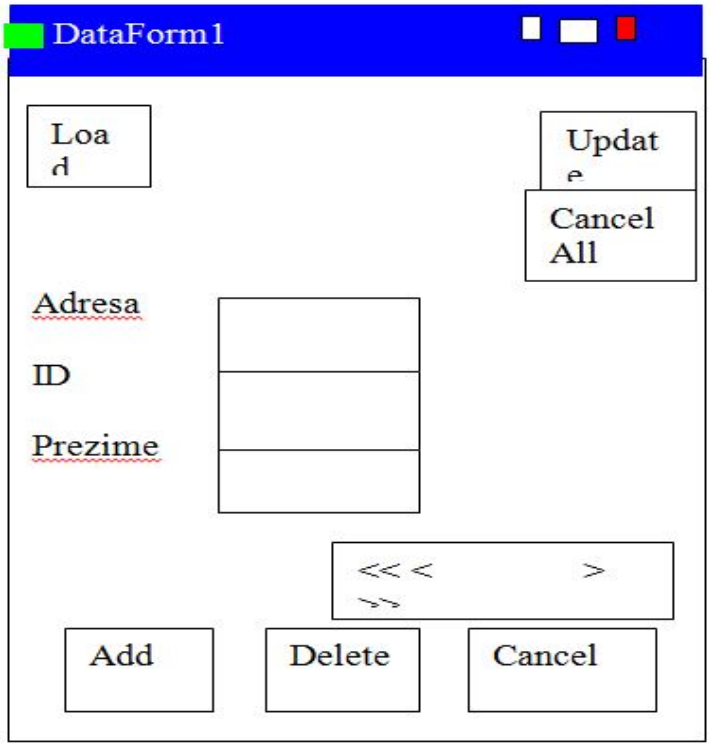
10. Ako se krene sa izvršavanjem projekta, pojavljuje se DataForm1, sa dugmadima koja su gore navedena (**Cancel All, *Add, Delete, Cancel, Navigation controls***). Takodje, automatski tu su i dugmad ***Load i Update.*** Na formi kliknuti ***Load*** da bi se prikazali podaci. Imaju dugmad za pomeranje napred i nazad po tabeli, ali se u odredjenom momentu prikazuje samo jedan zapis.

11. Da bi se dodao novi red (zapis) u bazu podataka, kliknuti dugme ***Add***, čime se dodaje prazan zapis koji se može onda popuniti, i onda kliknuti **Update** da bi memorisali unete podatke.

12. Dakle napravili smo jednostavnu Visual C# aplikaciju, koja komunicira sa bazom podataka **SportskiKlub.MDB,** (koristi podatke iz tabele **ClanoviKluba),** i prikazuje podatke iz baze podataka na formi ***DataForm1***, i omogućeno je da se mogu dodavati ili menjati ili brisati podaci u bazi podataka (komande ***Load***, ***Add, Delete, Update***).

SKICA **DATAFORM**-E SA NAVIGACIONIM DUGMADIMA I DUGMETOM CANCEL ALL

*Slika prikazuje skicu **DataForm**-e (napravljena pomoću Data Form Wizard), sa navigacionim dugmadima i dugmetom Cancel All, itd. (nisu prikazana sva polja iz tabele baze podataka).*



Sl1: dataform

UNAPREĐENJE DATAFORM-E

DataForm-a napravljena pomoću DataForm Wizard-a u prethodnom poglavlju,ima nedostatke.

DataForm-a je najbrži način da napravite aplikaciju sa bazom podataka. Ali , DataForm se može puno unaprediti, ili napraviti potpuno drugačija forma. DataForm-a napravljena pomoću *DataForm Wizard*-a u prethodnom poglavlju,ima nedostatke:

- Polja su alfabetskom redosledu a ne logičnom redosledu
- Redovi nisu sortirani
- Nema načina tj. za sada nije predvidjeno da se pretražuje baza podataka

PROGRAMIRANJE DATAFORM-E

Evo kako izgledaju poboljšanja aplikacije sa dataform-om.

Sada ćemo pokazati kako može da se poboljša i isprogramira DataForm-a koju smo prethodno generisali pomoću *DataForm Wizard*-a. Programske instrukije nisu date, jer ovde nećemo ulaziti u detalje programiranja baza podataka. Evo kako izgleda to poboljšanje:

Dodati dugme ***Search*** (tj. Pretraga), I dati mu ime ***btnSearch.*** I zadati svojstvo ***Enabled*** da bude ***False*** (da se ne bi nekontrolisano aktiviralo), I biće stavljeno ***True*** umesto ***False*** u programu. Dodati ***TextBox*** poi menu ***txtSearch***, I staviti ponovo svojstvo ***Enabled*** da je **False**.

Pomoću miša promeniti red tekst-boksova i nalepnica (*Labels*) da bi imali logičan redosled polja a ne alfabetski.

Duplo kliknuti dugme **Load** da se otvori ***Click event handler*** za ovo dugme**:**

{……

}

Tu se mogu ukucati instrukcije za sortiranje, koje ovde nećemo prikazati.

Onda duplo kliknuti dugme ***Search,*** i otvara se *Click event handler* za ovo dugme:

{……

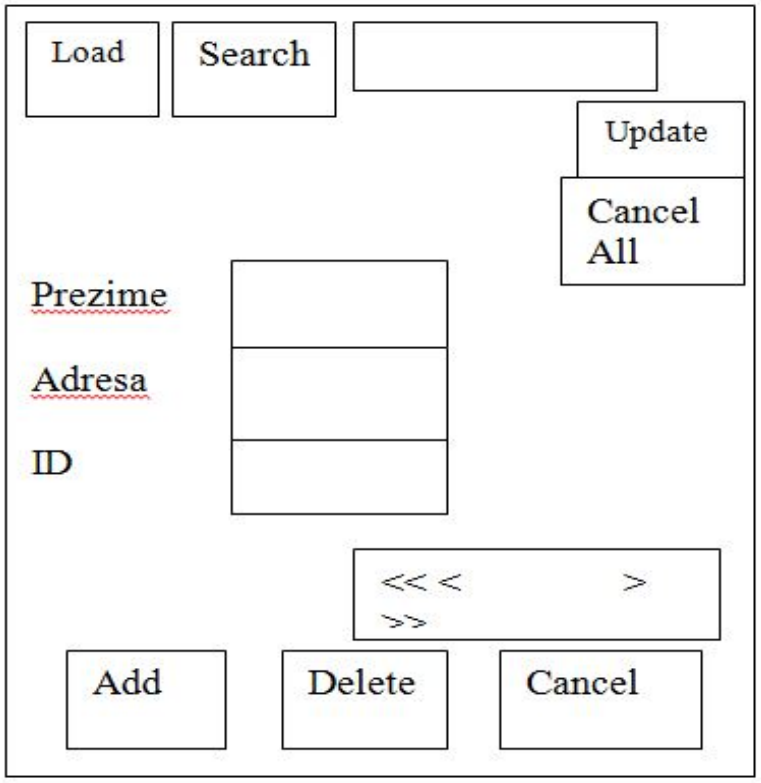
}

Onda se dodaju instrukcije za pretragu baze podataka.

Izvršiti aplikaciju, kliknuti ***Load***, I uraditi pretragu pomoću ***Search.***

SKICA *DATAFORM*-E SA DUGMETOM *SEARCH* I *TEXTBOX*-OM *TXTSEARCH*

Na slici je skica DataForm-e sa dugmetom Search i TextBox-om txtSearch pored dugmeta Search(nisu prikazana sva polja iz tabele baze podataka).



sl2: dataform

Razdvojeni model baze podataka

04

DISCONNECTED DATABASE MODEL:

*U aplikaciji napravljenoj pomoću **DataForm Wizard**, jedina dugmad koja izazivaju spajanje veze sa izvornom bazom podataka su dugmad **Load** i **Update**.*

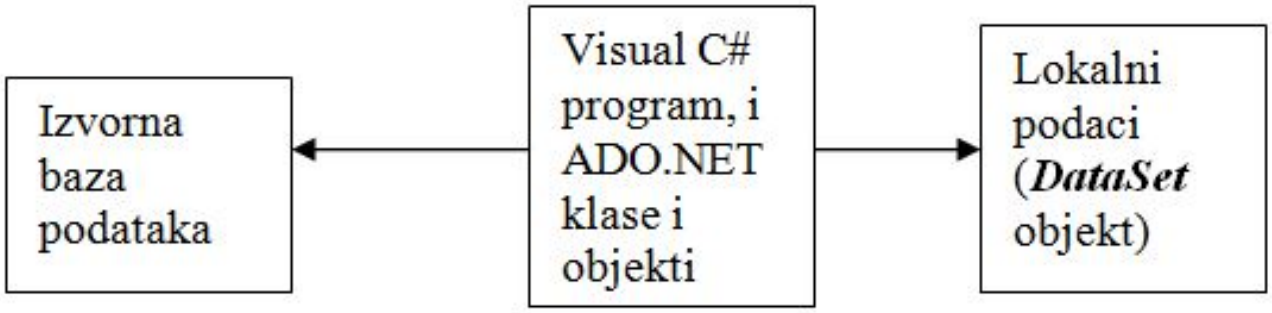
ADO.NET je Visual C# biblioteka klasa za rad sa bazama podataka, dakle ***ADO.NET*** je deo ***.NET***-a namenjen komunikaciji sa bazama podataka i obradu (procesiranje) baza podataka. ***ADO.NET*** sadrži čitav niz funkcionalnosti (funkcija) koje omogućuju laku interakciju sa bazama podataka i procesiranje baza podataka. ***ADO.NET*** koristi ***disconnected data model***, što znači da se veza sa bazom podataka koristi samo kod učitavanja (*retrieving*) podataka iz baze ili kod ažuriranja (*updating*) originalnih podataka u bazi. Dakle operacije kao što su navigacija kroz podatke, i čak dodavanje i promena pojedinih zapisa, može da se obavi bez odlaska u izvornu bazu podataka. Ove izmene u izvornoj bazi se dese tek kada se konkretno naloži updating tj. ažuriranje u izvornoj bazi podataka.

Npr. u aplikaciji napravljenoj pomoću ***DataForm Wizard***, jedina dugmad koja izazivaju spajanje veze sa izvornom bazom podataka su dugmad ***Load*** i ***Update***. Vi možete da editujete podatke, dodajete zapise i brišete zapise pomoću ***Add*** i **Delete**, ali ništa se neće desiti u izvornoj bazi dok ne kliknete **Update.** Takodje postoji opcija ***Cancell All***, što poništava sve izmene od poslednjeg ***Update***.

Razdvojeni model podataka je dakle model gde ADO.NET ima sopstveni **database manager,** sopstveni alat za obradu baze podataka. Prema tome, umesto da se drži povezan sa izvornom bazom podataka sve svreme, ***ADO.NET*** vrši uvoz podataka iz izvorne baze u svoj sopstveni **database manager,** predstavljen sa **DataSet** objektom. Objekat **DataSet** memoriše sve promene i dodatke podataka, i kada želite da ažurirate izvornu bazu, program vrši izmene u izvornoj bazi. Ovaj pristup iako izgleda komplikovan ima niz prednosti, ali ima i nedostatke.

RAZDVOJENI MODEL PODATAKA (*DISCONNECTED DATA MODEL*)

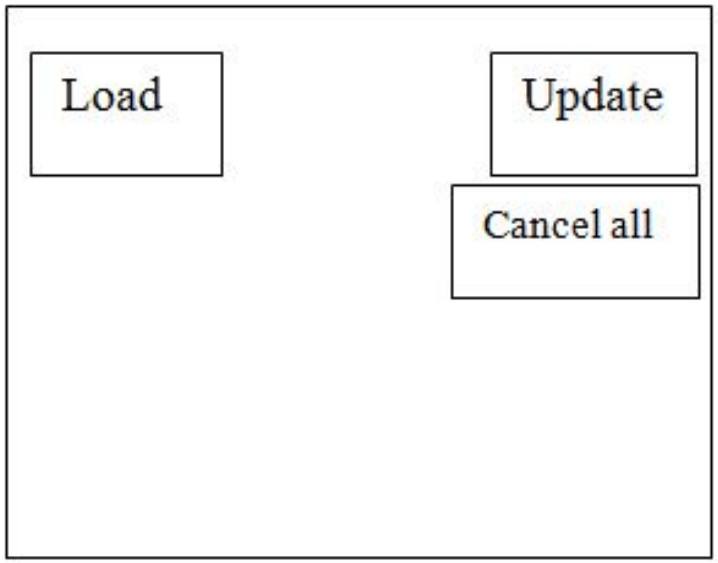
Slika prikazuje razdvojeni model podataka (disconnected data model).



Sl1: disconnected model

SKICA FORME SA DUGMADIMA LOAD, UPDATE, CANCEL ALL

Na slici se vidi skica forme sa dugmadima Load, Update, Cancel all.



Sl2: dataform

ADO.NET

05

ADO.NET

***ADO.NET** je **Visual C#** biblioteka klasa za rad sa bazama podataka.*

ADO.NET je biblioteka programa u okviru Visual C# (***Visual C# database library***) koja omogućuje programiranje Visual C# aplikacija koje koriste baze podataka. Dakle povezivanje sa nekom bazom podataka i rad sa nekom bazom podataka (pretrage, sortiranje, procesiranje podataka, itd.) može se isprogramirati (programiranje VisualC# aplikacija sa bazama podataka) pomoću ***ADO.NET*** , ali je za to potrebno detaljno poznavanje biblioteke ***ADO.NET***, što nije predvidjeno u okviru ovog predmeta.

ADO.NET je ***Visual C#*** biblioteka klasa za rad sa bazama podataka. ADO.NET koristi *disconnected data model*, što znači da se veza sa bazom podataka koristi samo kod učitavanja (*retrieving*) podataka iz baze ili kod ažuriranja (*updating*) originalnih podataka u bazi. Dakle operacije kao što su navigacija kroz podatke, i čak dodavanje i promena pojedinih zapisa, može da se obavi bez odlaska u izvornu bazu podataka. Ove izmene u izvornoj bazi se dese tek kada se konkretno naloži ***updating*** tj. ažuriranje u izvornoj bazi podataka. Npr. u aplikaciji napravljenoj pomoću ***DataForm Wizard***, jedina dugmad koja izazivaju spajanje veze sa izvornom bazom podataka su dugmad ***Load*** i ***Update***. Vi možete da editujete podatke, dodajete zapise i brišete zapise pomoću **Add** i ***Delete***, ali ništa se neće desiti u izvornoj bazi dok ne kliknete ***Update.*** Takodje postoji opcija ***Cancell All***, što poništava sve izmene od poslednjeg ***Update***.

Razdvojeni model podataka je dakle model gde ***ADO.NET*** ima sopstveni ***database manager***, sopstveni alat za obradu baze podataka. Prema tome, umesto da se drži povezan sa izvornom bazom podataka sve vreme, ***ADO.NET*** vrši uvoz podataka iz izvorne baze u svoj sopstveni ***database manager***, predstavljen sa ***DataSet*** objektom (klasom). Objekat ***DataSet*** memoriše sve promene i dodatke podataka, i kada želite da ažurirate izvornu bazu, program vrši izmene u izvornoj bazi. Ovaj pristup iako izgleda komplikovan ima neke velike prednosti (autonomnost,...), ali ima i neke nedostatke (asinhronost, ..

ADO.NET KLASE

Evo kratkog opisa ADO.NET objekata (klasa).

Ako se pogleda aplikacija napravljena pomoću ***DataForm Wizard***, primećuje se da ona koristi tri nevidljiva objekta, ***Connection, DataAdapter, i Dataset.*** Ovo su objekti dobijeni preko **ADO.NET** klasa. Tačna kombinacija i tipovi uključenih ***ADO,NET*** objekata zavisi od izabranog tipa veze, ***connection***, izmedju aplikacije i baze podataka. U primeru sa DataForm Wizard koristili smo ***OLEDB connection***, pa su uključene ***ADO.NET*** objekti bili ***OleDbConnectio***n i ***OleDbDataAdapter***, a objekat (klasa) ***Dataset*** je uvek isti, bez obzira na tip veze sa bazom podataka.

Evo kratkog opisa ADO.NET objekata (klasa):

Dataset je u stvari lokalni ***database manager*** koji je u okviru ADO.NET, i služi za operisanje sa tabelama, a same tabele su reprezentovane preko ***DataTable*** objekata.

DataTable, predstavlja skup zapisa. Često, ove tabele su dobijene pretragom tabela u izvornoj bazi podataka.

DataView. Služi za prikaz podataka iz neke tabele, obično to je sortiranje i filtriranje zapisa u nekoj tabeli.

iDbConnection, upravlja vezom sa izvornom bazom podataka, npr. pomoću metoda Open i Close,

iDbCommand, predstavlja SQL instrukcije. ***SQL*** instrukcije: ***select, update, delete, insert***.

iDbDataAdapter, za eksport i import izmedju ***Dataset*** i izvorne baze podataka.

UPOTREBA ADO.NET KLASA U APLIKACIJAMA

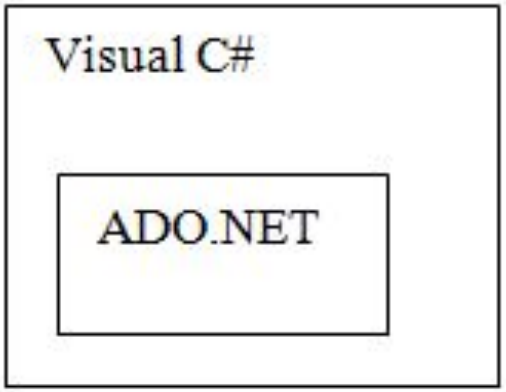
Evo nekih ADO.NET klasa: Dataset, DataTable, DataView, iDbConnection, iDbCommand, i iDbDataAdapter.

U narednoj glavi videćemo upotrebu ADO.NET klasa u razvoju aplikacije gde se koristi „kontrola“ ***DataGrid.*** Medjutim, mogu se praviti razne aplikacije uz upotrebu ADO.NET-a, npr. može se na formu postaviti ***ListBox*** koja se zatim popunjava uz pomoć ***ADO.NET***. Ili npr. na formu postaviti ListBox i DataGrid, kao što je dole ilustrovano. Onda se mogu dodati razne instrukcije za pretragu ako vršimo samo pretragu i uzimanje podataka iz izvorne baze podataka, a ako vršimo i ažuriranje izvorne baze pdataka onda koristimo i instrukcije za ažuriranje izvorne baze podataka. I u zavisnosti od toga biće upotrebljene pojedine klase iz ADO.NET-a, a to su klase:

Dataset, DataTable, DataView, iDbConnection, iDbCommand, i iDbDataAdapter, gde malo slovo „**i“** ispred imena klase znači klasni interfejs.

ADO.NET BIBLIOTEKA

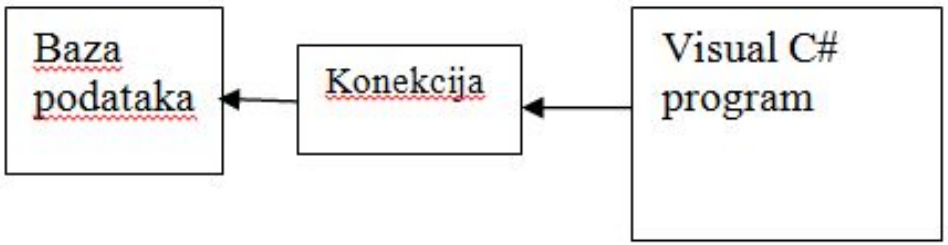
*Naslici vidimo **ADO.NET** biblioteku kao deo Visual C#-a..*



Sl1: ADO.NET

VEZA IZMEĐU VISUAL C# APLIKACIJE I BAZE PODATAKA

Slika ilustruje vezu izmedju Visual C# aplikacije i baze podataka.



sL2: KONEKCIJA BAZE PODATAKA

DataGridView, i DataGrid

06

DATAGRID

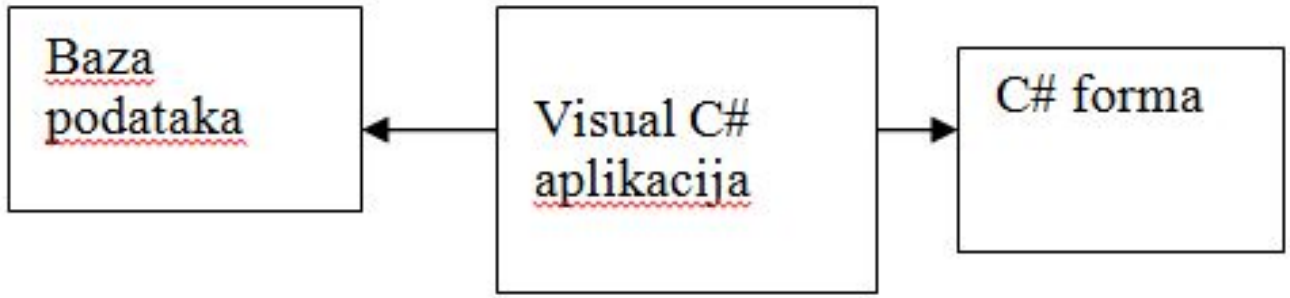
***DataGrid**, to je veoma korisna i veoma korišćena „kontrola“ u Visual C#.*

DataForm Wizard, DataGrid, *DataGridView*, raspolživi su u VisualC #, i oni nam omogućuju da pravimo različite „forme“ na kojima se prikazuju podaci „pokupljeni“ iz neke baze podataka. Ova baza podataka može biti kreirana pomoću Access, ili SQL Servera, itd. U Visual C# Express povezivanje sa bazom podataka se obavlja pomoću ***Data Source Configuration Wizard***. ***DataGridView*** je raspoloživo u ***Visual C# Express***, a ***DataGrid*** u ***Visual Studio***-u.

DataGrid, to je veoma korisna i veoma korišćena „kontrola“ u Visual C#, to je objekat tj. alat koji je raspoloživ u Toolbox-u, i koji prikazuje podatke u obliku mreže podataka, slično kao *spreadsheet*, a i omogućuje editovanje tih podataka. Dakle, DataGrid je „kontrola“ namenjena tabelarnom prikazivanju podataka, a takodje koristi se za dodavanje ili brisanuje redova ili ažuriranje postojećih redova u nekoj tabeli, i pri tome je u sprezi tj. konekciji sa izvornom bazom podataka.

UPOTREBA VISUAL C# KOD BAZE PODATAKA

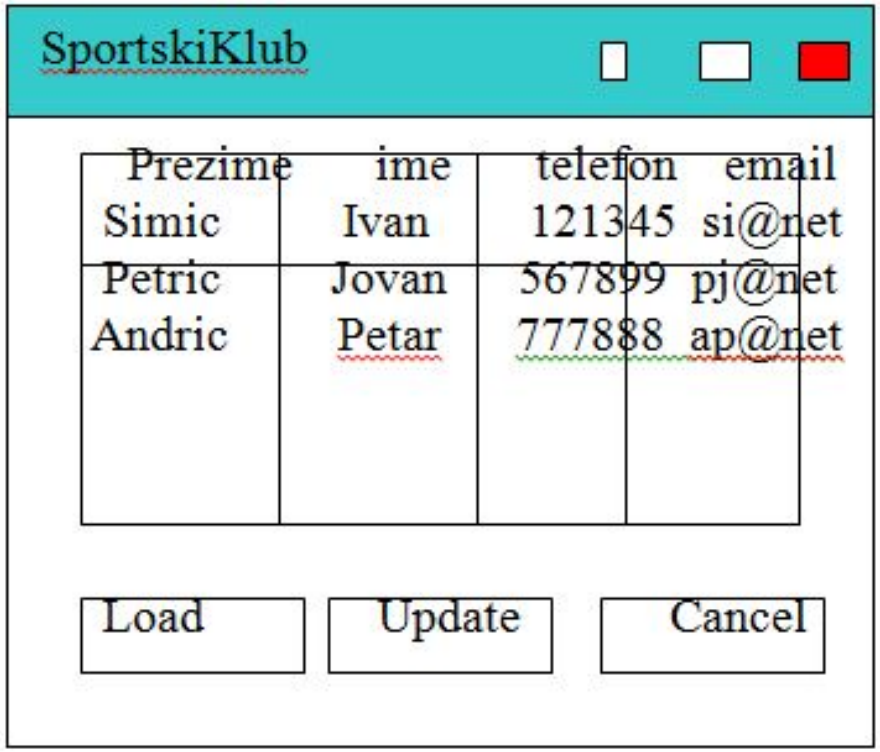
Na slici je ilustrovana upotreba Visual C# kod baze podataka.



sl1: datafom

FORMA SA DATAGRID I 3 DUGMETA

Na slici je ilustrovana forma sa DataGrid i 3 dugmeta.



Sl.2: Datagrid

DATAGRID-ILUSTRACIJA

*Evo dole je data ilustracija korišćenja **DataGrid** kontrole.*

Evo dole je data ilustracija korišćenja ***DataGrid,*** u kombinaciji sa izvornom bazom podataka. Koristićemo bazu podataka koju smo formirali u prethodnom predavanju ***SportskiKlub.mdb***. Prvo ćemo se povezati sa tom bazom podataka. A onda primeniti „kontrolu“ **DataGrid**. Takodje, koristićemo ***Data Adapter Configuration Wizard***, a to je GUI tj. dijalog-boks koji omogućuje korisniku da se poveže sa izvornom bazom podataka, i takodje da odabere podatke koje želi da prikazuje na „formi“.

1. Otići na ***Server Explorer window***: ***View menu*** > ***Server Explorer >Data connections***. I tamo se može naći ***connection*** sa bazom podataka koju smo formirali u prethodnom predavanju **Sportskiklub.mdb**. Ako medjutim tada nije bila uspostavljena veza sa tom bazom podataka (naime, ako nije uradjen primer sa ***Data Form Wizard*** i ***DataForm***), onda uraditi sledeće: u ***Data Link Properties dialog*** (***Data Form Wizard dialog***) kliknuti ***Povider,*** i onda **OLE DB Provider**, i onda uneti ime izvrorne baze podataka (ili ga selektovati tj. pronaći gde je memorisan na disku: npr. C:\mydata***Sportskiklub.mdb***), pa zatim kliknuti ***Test connection*** a ovo će biti praćeno porukom ***Connection Succeeded***. Dakle tako smo se povezali sa bazom podataka ***Sportskiklub.mdb.***

2. Na formu postaviti objekte ***Data Grid***, i 3 dugmeta. Zadati ime tj. ***name property*** za ova dugmad da budu

btnLoad, btnUpdate, i btnCancel, i ***text-property*** da bude ***Load, Update, Cancel***

3. U ***Toolbox***-u, u sekciji **Data**, izabrati ***OleDbDataAdapter***, i njega treba prevući na formu. Pomoću ovog elementa ***OleDbDataAdapter***, mogu se učitavati ppodaci iz baze podataka tj. unositi u naš program. Takodje on omogućuje da se ažurirani podaci pošalju nazad u izvornu bazu podataka. Dakle OleDbDataAdapter omogućuje komunikaciju sa bazom podataka u dva smera.

DATAGRID-ILUSTRACIJA

*Evo dole je data ilustracija korišćenja **DataGrid** kontrole.*

4. Pošto ste prevukli **OleDb*DataAdapter*** na formu, ovo će izazvati pojavu jednog GUI koji se zove ***Data Adapter Configuration Wizard***. (On pomaže kod specificiranja konekcije i komandi koje ***DataAdapter*** koristi, i pri tome treba pružiti informaciju o konekciji sa bazom , i doneti neke odluke kako komande da se storniraju i izvršavaju). I zatim treba kliknuti **Next.**

5. Onda na strani koja se pojavila u ***Data Adapter Configuration Wizard*** izabratri konekciju za bazu podataka: ***ACCESS Sportskiklub.mdb*.**

ltd

Vežba 7

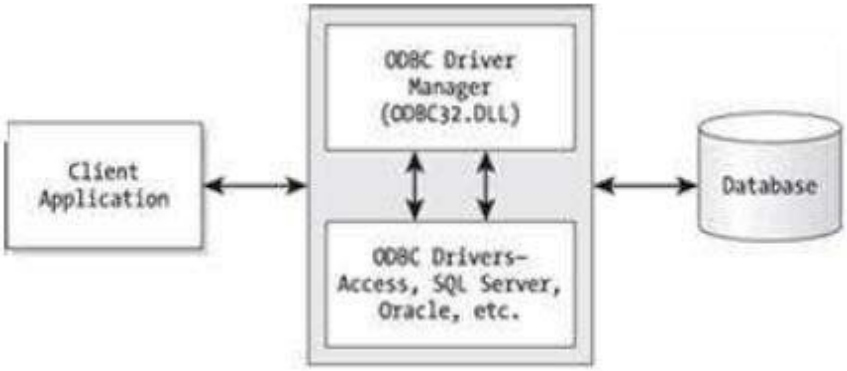
07

USPOSTAVLJANJE KONEKCIJE SA BAZOM PODATAKA

ODBC (Open Database Connectivity) je prva tehnologija za pristup bazi podataka razvijena od strane Microsofta.

ODBC

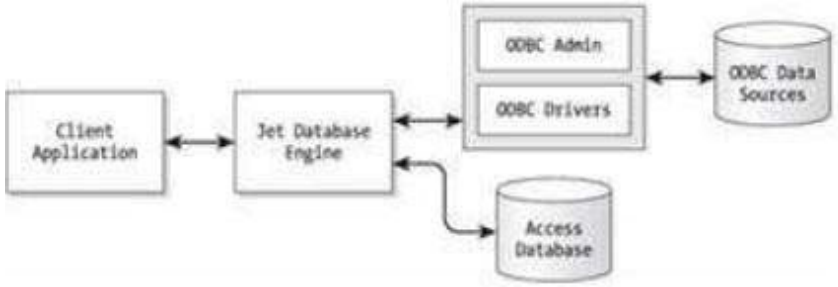
ODBC (Open Database Connectivity) je prva tehnologija za pristup bazi podataka razvijena od strane Microsofta. Obezbenuje C/C++ API (aplikacioni programski interfejs) niskog nivoa za pristup podacima iz baze. Teška je komunikacija sa nerelacionim izvorima podataka.



Slika-1: ODBC

DAO

DAO (Data Access Objects) je skup COM (Component Object Model) interfejsa za pristup bazi podataka. DAO je jednostavniji za korišćenje nego ODBC API. Za pristup bazi koja nije Access DAO interno koristi ODBC kao što je to prikazano na slici. Na taj način se uvodi jedan dodatni sloj za pristup ODBC izvoru podataka.



Slika-2: DAO

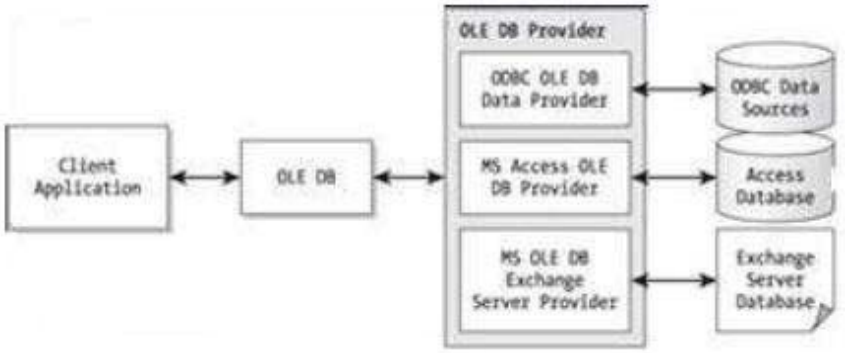
USPOSTAVLJANJE KONEKCIJE SA BAZOM PODATAKA DRUGI DEO

ODBC (Open Database Connectivity) je prva tehnologija za pristup bazi podataka razvijena od strane Microsofta.

OLE-DB

OLE-DB(Object Linking and Embedding Database) tehnologija omogućava pristup kako relacionim tako i nerelacionim izvorima podataka. Da bi se koristila ova ehnologija potrebno je instalirati OLE-DB snabdevač podataka (data provider) koga

obezbenuje proizvonač baze podataka.



Slika-3: OLE-DB

ADO I ADO.NET

ADO je skup ActiveX komponenti koje obezbenuju objektno orijentisani pristup ka OLEDB. ADO se može shvatiti kao COM omotač oko OLE DB-a. ADO se može koristiti sa bilo kojim programskim jezikom koji podržava COM.

ADO.NET je nastao razvojem ADO-a i koristi sve prednosti .NET okruženja

.NET snabdevači podataka (Data Providers)

.NET snabdevač podataka je skup klasa koje se koriste za konektovanje na izvor podataka. Unutar .NET Framework-a nalaze se SQL Server .NET snabdevač podataka i OLE DB.NET snabdevač podataka. SQL Server .NET snabdevač podataka koristi Tabular Data Stream (TDS) protokol za slanje zahteva i primanje odgovora od SQL Server-a. Ima visoke performanse jer je protokol TDS veoma brz jer pristupa SQL Server-u direktno bez OLE DB ili ODBC sloja. Klase SQL Server .NET snabdevača podataka nalaze se u prostoru imena System.Data.SqlClient. Ovaj snabdevač podataka se preporučuje za sve aplikacije koje rade sa SQL Server 7 verzijom ili više.

OLE DB snabdevači podataka

On komunicira sa izvorom podataka sa OLE DB provajderom specifičnim za taj izvor podataka. Preporučuje se korišćenje ovog snabdevača podataka za aplikacije koje komuniciraju sa Accessom. Klase ovog snabdevača podataka nalaze se u prostoru imena System.Data.OleDb.

KONEKCIJE

Pre bilo kakvog rada sa bazom podataka potrebno je kreirati a zatim otvoriti konekciju.

Pre bilo kakvog rada sa bazom podataka potrebno je kreirati a zatim otvoriti konekciju. U ADO.NET-u se kreira objekat klase Connection. Klasa System.Data.SqlClient.SqlConnection omogućava kreiranje konekcije na SQL Server bazu podataka. Klasa System.Data.OleDb.OleDbConnection omogućava kreiranje konekcije na svaki izvor podataka sa pridruženim OLE DB provajderom.

Connection objekat

Svaki Connection objekat implementira System.Data.IDbConnection interfejs. Najčešće korišćene metode ovog interfejsa su Close( ) i Open( ) kao i property ConnectionString.

SQL Server – ConnectionString

Konekcija na SQL Server bazu podataka kreira se instanciranjem klase SqlConnection. Kreiranje SqlConnection objekta može se izvršiti pozivom podrazumevanog konstruktura. Zatim se preko svojstva ConnectionString definišu parametri konekcije koji se pišu u obliku stringa. Drugi način kreiranja SqlConnection objekta je da se konstruktoru ove klase prosledi string koji opisuje konekciju na SQL server bazu podataka. Konekcija na SQL server zavisi od tipa autentifikacije.

U nastavku je prikazano kreiranje konekcionog stringa za SQL server autentifikaciju. Konekcioni string ima parametre DataSource koji predstavlja ime SQL Servera sa kojim se radi (npr. localhost za lokalni SQL Server), Initial Catalog odrenuje bazu podataka na SQL Serveru sa kojom se radi, user id predstavlja korisničko ime na SQL serveru a password predstavlja šifru tog korisnika. Pri kreiranju konekcionog stringa atributi se razdvajaju znakom ";".

```
SqlConnection con = new SqlConnection();
con.ConnectionString = "...";
SqlConnection con = new SqlConnection("...");
//SQL Server autentifikacija
SqlConnection con = new SqlConnection();
string konekcioniString = "Data
Source=localhost;" +
"Initial Catalog=Northwind;user
id=userid;password=password ");
con .ConnectionString = konekcioniString;
```

KONEKCIJE DRUGI DEO

Pre bilo kakvog rada sa bazom podataka potrebno je kreirati a zatim otvoriti konekciju.

U slučaju windows autentifikacije korisnik se ne mora prijavljivati na SQL Server već je dovoljno da bude korisnik domena na kome SQL server radi. Zbog toga korisnik koji ima windows nalog na domenu može pristupiti SQL Serveru koristeći izraz Integrated Security=SSPI umesto specificiranja korisničkog imena i šifre.

```
SqlConnection con = new SqlConnection();
string konekcioniString = "Data Source=localhost;" +
"Initial Catalog=Northwind;Integrated
Security=SSPI"); con.ConnectionString =
konekcioniString;
```

OLE-DB Konekcija

Za izvor podataka koji nije SQL Server u konekcionom stringu se mora navesti parameter Provider. Za pristup Access bazi podataka može se koristiti provajder Microsoft.Jet.OLEDB.4.0.

```
OleDbConnection con = new OleDbConnection();
string konekcioniString = "Data Source= c:\Nortwdind.mdb;"
+
"Provider=Microsoft.Jet.OLEDB.4.0"); SQL Server 6.5
Provider=SQLOLEDB;Data Source=London;Initial
Catalog=pubs;User ID=sa;Password=2389; Oracle Server:
Provider=MSDAORA;Data Source=ORACLE8I7; User
ID=OLEDB;Password=OLEDB;
```

Obrada izuzetaka

Preporučuje se da se operacije nad bazom podataka koje mogu da dovedu do greške u program pišu unutar try bloka. Unutar try bloka vrši se i otvaranje konekcije pozivom metode Open objekta klase SqlConnection. Samo kreiranje konekcije trebalo bi da bude izvan try bloka jer bi u protivnom ograničili oblast važenja ovog objekta. Ukoliko done do izuzetaka oni su objekti klase SqlConnection. Poželjno je i definisanje finally bloka u kojem će se vršiti zatvaranje konekcije bez obzira da li je pri izvršavanju sql komandi došlo do grešaka ili nije.

```
try
{
con.Open(); // (Izvrsavanje ADO.NET komandi)
}
catch (SqlException err)
{
Console.WriteLine(err.ToString());
}
finally
{
con.Close();
}
```

CONNECTION POOLING

Kada se neka konekcija zatvori ona se stavlja u konekcioni bazen.

Kada se neka konekcija zatvori ona se stavlja u konekcioni bazen. Kada se pozove ponovo Open metoda provajder prvo pretražuje bazen i ako konekcioni objekat ne postoji u bazenu kreira novi, ako postoji koristi ga. Definiše se maksimalni broj konekcija u bazenu. Definiše se i Connection Lifetime što odrenuje dužinu života konekcije u bazenu.

Događaji konekcije

Objekat SqlConnection ima doganaj StateChange koji se trigeruje kada konekcija menja stanje iz otvorenog u zatvoreno i obrnuto. Trenutno stanje konekcije se može iščitati korišćenjem propertija CurrentState, a prethodno stanje konekcije se iščitava korišćenjem propertija OriginalState. Ukoliko se pokuša da se ponovo otvori već otvorena konekcija generiše se izuzetak.

```
private void cnNorthwind_StateChange(object sender,System.Data.StateChangeEventArgs e)
{
MessageBox.Show("Tekuce stanje: " + e.CurrentState.ToString() + " Prethodno stanje: " +
e.OriginalState.ToString());
}
```

DATAGRIDVIEW

DataGridView je moćna, fleksibilna kontrola, jednostavna za korišćenje, koja se koristi za prikaz tabelarnih podataka

DataGridView je moćna, fleksibilna kontrola, jednostavna za korišćenje, koja se koristi za prikaz tabelarnih podataka. Može da se poveže sa jednom tabelom iz baze ili sa objektom koji je izveden iz tabele. Korišćenjem svojstva DataSource povezuje sa sa objektom klase DataTable ili objektom klase koja je izvedena iz klase

DataTable. Ako se izvrši povezivanje sa objektom klase DataSet mora se specificirati i njeno svojstvo DataMember kojim se naznačava sa kojom tabelom se vrši povezivanje. DataGridView kreira jednu kolonu za svaku kolonu iz izvora podataka. Header kolone kreira na osnovu naziva odgovarajućeg atributa iz tabele. Dozvoljava različite tipove selekcije: može se selektovati jedna ili više vrsta, jedna ili više kolona, jedna ili više ćelija, ili cela tabela (klikom na ćeliju u gornjem levom uglu).

Osobine DataGridView

Korišćenjem tastera Tab prelazi se iz jedne u drugu ćeliju tabele. Ima automatski tooltip kojim se prikazuje ceo sadržaj ćelije kada se pokazivač miša nalazi iznad nje. Podržava automatsko sortiranje, klikom na header kolone. Dvostrukim klikom izmenu header-a kolona vrši se automatsko podešavanje širine kolone.

Omogućava unos sadržaja u ćeliju nakon dvostrukog klika na ćeliju.

Povezivanje DataGridView kontrole sa tabelom iz koda

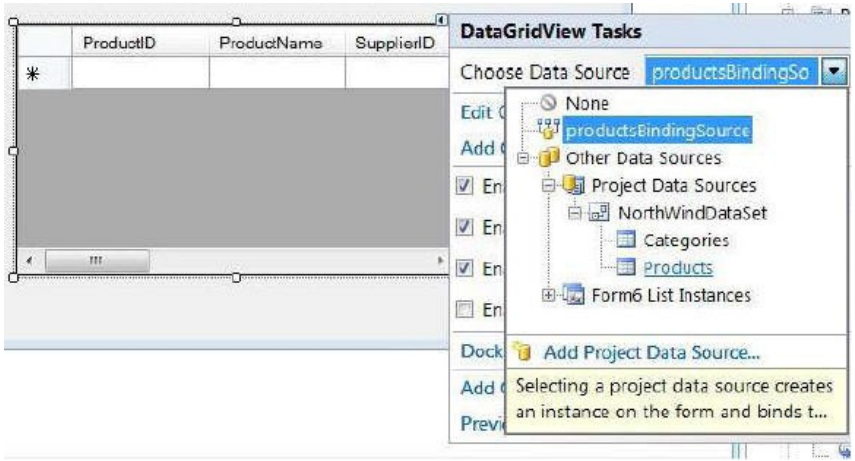
```
dataGridView1.DataSource = nds.Products;
dataGridView1.DataSource = nds;
```

Pomoću svojstva DataSource specificira se tabela iz tipiziranog DataSet objekta sa kojom se povezuje DataGridView kontrola. Drugi način je da se kao DataSource svojstvo specificira DataSet objekat, ali se u ovom slučaju mora definisati ataMember svojstvo, odnosno tabela iz DataSeta sa kojom se vrši povezivanje.

POVEZIVANJE DATAGRIDVIEW KONTROLE SA TABELOM U DIZAJN MODU

Povezivanje DataGridView kontrole sa tabelom u dizaj modu

Povezivanje DataGridView kontrole sa tabelom u dizajn modu



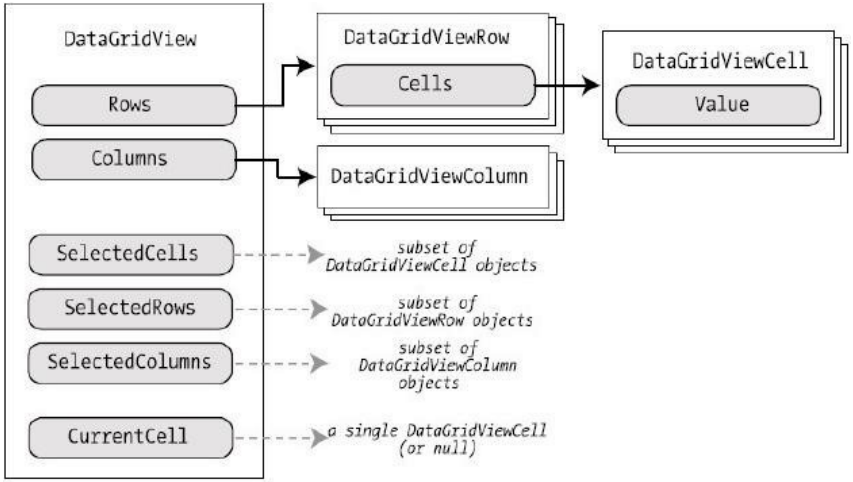
Slika-4: Povezivanje iz dizajnera

Klikom na strelicu u gornjem desnom uglu DataGridView kontrole otvara se prozorDataGridView Tasks. U ovom prozoru se klikom na ComboBox kontrolu Choose Data Source specificira tabela iz DataSeta koju treba prikazati unutar DataGridView kontrole. Ovim postupkom se automatski kreira odgovarajući BindingSource objekat kao i punjenje tabele iz DataSeta podacima iz baze podataka primenom TableAdapter klase.

Kod koji puni tabelu unutar DataSeta se dodaje u Load proceduri forme.

DataGridView objekti

Svojstvo Colums objekta klase DataGridView daje kolekciju DataGridViewColumn objekata. Svojstvo Rows daje kolekciju DataGridViewRow objekata. DataGridViewRow objekat ima svojstvo Cells kojim se prikazuje kolekcija DataGridViewCell objekata.



Slika-5: DataGridView

DATAGRIDVIEWCOLUMN OBJEKAT

DataGridViewColumn objektima date DataGridView kontrole se pristupa korišćenjem svojstva Columns.

DataGridViewColumn objektima date DataGridView kontrole se pristupa korišćenjem svojstva Columns. Svojstvo Colums vraća kolekciju DataGridViewColumn objekata. Sada se pojedininim kolonama tabele tj. DataGridViewColumn objektima pristupa ili korišćenjem rednog broja vrste u tabeli ili korišćenjem njenog naziva. Kada se u prozoru DataGridView Tasks klikne na opciju Edit Columns otvara se prozor EditColums u kome možemo iščitati naziv odrenene kolone koja ulazi u sastav DataGridView kontrole. U primeru na slici obranuje se Click doganaj na kontrolu Button. Unutar event handlera za klik doganaj na dugme, korišćenjem foreach petlje se iščitavaju nazivi svih kolona koje ulaze u sastav DataGridView kontrole.

Prikaz header-a kolona

```
DataGridViewColumn col = dataGridView1.Columns[0];
col.HeaderText = "ID proizvoda";
foreach (DataGridViewColumn dgvC in
dataGridView1.Columns)
{
listBox1.Items.Add(dgvC.HeaderText);
}
```

Zaglavlje odrenene kolone koja ulazi u sastav DataGridView objekta. U gornjem kodu, korišćenjem foreach petlje unutar ListBox kontrole se prikazuju header-i svih kolona naše DataGridView kontrole.

DataGridViewRow objekat

```
private void button3_Click(object sender, EventArgs e)
{
// pristup drugoj vrsti
DataGridViewRow dgvR = dataGridView1.Rows[1];
// prikazi sadrzaj celija druge vrste
foreach (DataGridViewColumn dgvC in dataGridView1.Columns)
{
listBox1.Items.Add(dgvR.Cells[dgvC.Name].Value);
}
}
```

Korišćenjem svojstva Rows objekta klase DataGridView, pristupa se kolekciji DataGridViewRow objekata, koji predstavljaju vrste DataGridView kontrole. Vrstama unutar DataGridView kontrole se isključivo pristupa na osnovu njenog rednog broja, tj. indeksa koji je baziran na broju 0. U primeru na slajdu pristupa se drugoj vrsti, tj. pristupa se objektu klase DataGridViewRow koji predstavlja drugu vrstu grida. DataGridViewRow objekat ima svojstvo Cells koje vraća kolekciju DataGridViewCell objekata odnosno ćelije date vrste. Pojedinim ćelijama jedne vrste tj. ćelijama DataGridViewRow objekta pristupa se ili korišćenjem rednog broja kolone kojoj ćelija pripada ili korišćenjem naziva kolone kojoj ćelija pripada.

KLASE IZVEDENE IZ KLASE DATAGRIDVIEWCOLUMN

DataGridViewButtonColum klasa prikazuje dugmad tj. Button kontrole unutar jedne kolone DataGridView kontrole.

DataGridViewButtonColum klasa prikazuje dugmad tj. Button kontrole unutar jedne kolone DataGridView kontrole. Njoj odgovarjuća klasa za ćeliju je DataGridViewButtonCell.

Sledeća klasa je DataGridViewLinkColumn i njoj odgovarajuća klasa za ćeliju DataGridViewLinkCell čija je funkcionalnost slična funkcionalnosti Button kolone, ali se tekst prikazuje u formi linka.

DataGridViewCheckBoxColumn (DataGridViewCheckBoxCell se automatski kreira pri vizuelnom povezivanju DataGridView kontrole za tabelu koja sadrži Boolean tip podataka.

DataGridViewComboBoxColumn (DataGridViewComboBoxCell) se koristi se u situacijama kada je potrebno ograničiti unos u ćeliju tj. omogućiti korisniku da u ćeliju odabere jednu vrednost iz datog skupa vrednosti.

DataGridViewImageColumn (DataGridViewImageCell) se koristi se za prikaz slika. DataGridViewTextBoxColumn (DataGridViewTextBoxCell se koristi za prikaz teksta I predstavlja podrazumevani tip kolone u DataGridView kontroli.

KREIRANJE DATAGRIDVIEW KONTROLE KOJA NIJE POVEZANA SA IZVOROM PODATAKA

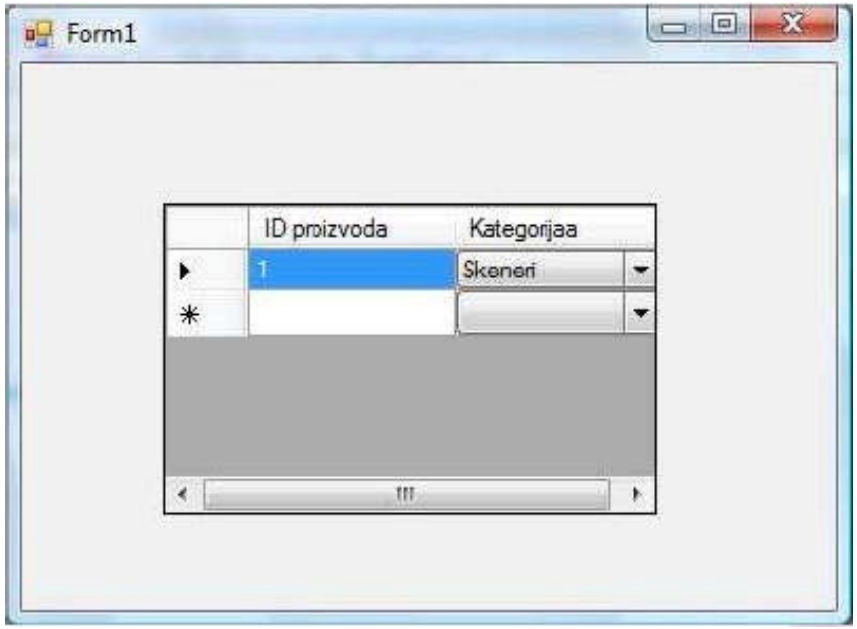
DataGridViewButtonColum klasa prikazuje dugmad tj. Button kontrole unutar jedne kolone DataGridView kontrole.

Uobičajena upotreba DataGridView kontrole je prikaz podataka iz neke tabele ili drugog izvorapodataka. Međutim, DataGridView kontrola se može iskoristiti za prikaz bilo kakvih podataka.

U donjem primeru najpre je na formu dodata DataGridView kontrola da bi se izvršilo instanciranje odgovarajuće klase. DataGridView kontrola nije povezana ni sa jednim izvorom podataka. Zatim se kreiraju se dve kolone DataGridView kontrole. Prva kolona je tipa DataGridViewTextBoxColumn a druga kolona je tipa DataGridViewComboBoxColumn. Posle inicijalizacije ovih kolona moraju se dodati u Columns koleciju DataGridView kontrole čime se kreira struktura DataGridView kontrole. Zatim se iz koda kreira i jedna vrsta DataGridView kontrole. Da bi se kreirala vrsta prethodno je neophodno kreirati ćelije koje ulaze u sastav ove vrste. U primeru to su ćelije tipa DataGridViewTextBoxCell i DataGridViewComboBoxCell. Tip ćelije koja ulazi u sastav reda mora odgovarati tipu kolone kojoj ta ćelija pripada.

```
DataGridViewTextBoxColumn  col1 = new DataGridViewTextBoxColumn();
col1.Name  = "ID";
col1.HeaderText  = "ID proizvoda";
dataGridView1.Columns.Add(col1);
DataGridViewComboBoxColumn  col2 = new DataGridViewComboBoxColumn();
col2.Items.Add("Stampaci");
col2.Items.Add("Skeneri");
```

Prikaz kreirane DataGridView kontrole



Slika-6: Kreirana DataGirdView kontrola

SELEKCIJA ĆELIJE

DataGridViewButtonColum klasa prikazuje dugmad tj. Button kontrole unutar jedne kolone DataGridView kontrole.

DataGridView kontrola ima svojstvo SelectionMode koje može imati sledeće vrednosti: CellSelect, FullColumnSelect, FullRowSelect, ColumnHeaderSelect i RowHeaderSelect.

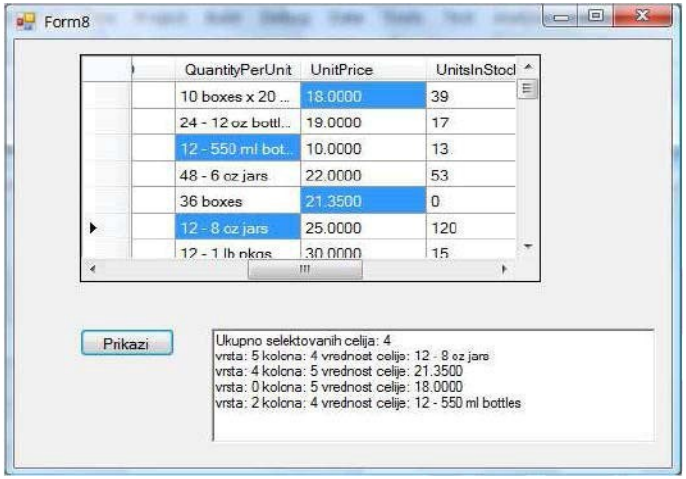
Sam naziv moda selekcije ukazije na to šta se može selektovati unutar DataGridView kontrole. Svojstvo MultiSelect postavljeno na vrednost True omogućava selektovanje više redova, kolona ili ćelija. SelectedCells svojstvo vraća kolekciju DataGridViewCell objekat nezavisno od moda selekcije DataGridView kontrole. Svojstvo SelectedRows daje selekciju selektovanih vrsta, dok svojstvo SelectedColumns daje kolekciju selektovanih kolona DataGridView kontrole.

Provera selektovanih ćelija

```
private void button1_Click(object sender, EventArgs e)
{
richTextBox1.AppendText("Ukupno selektovanih celija: " +
dataGridView1.SelectedCells.Count.ToString() + "\n");
foreach (DataGridViewCell celija in
dataGridView1.SelectedCells)
{
richTextBox1.AppendText("vrsta: " +
celija.RowIndex.ToString() + "kolona: " +
celija.ColumnIndex.ToString() + " vrednost celije: " +
celija.Value + "\n");
}
}
```

Pomoću svojstva SelectedCells DataGridView objekta pristupa se kolekciji selektovanih ćelija, tako da svojstvo Count ove kolekcije daje ukupan broj selektovanih ćelija. Primenom foreach petlje se prolazi kroz kolekciju selektovanih ćelija i za svaku od njih se prikazuje broj vrste I kolone kojima ćelija pripada kao i vrednost tj. podatak koji se nalazi u ćeliji.

Provera selektovanih ćelija GUI

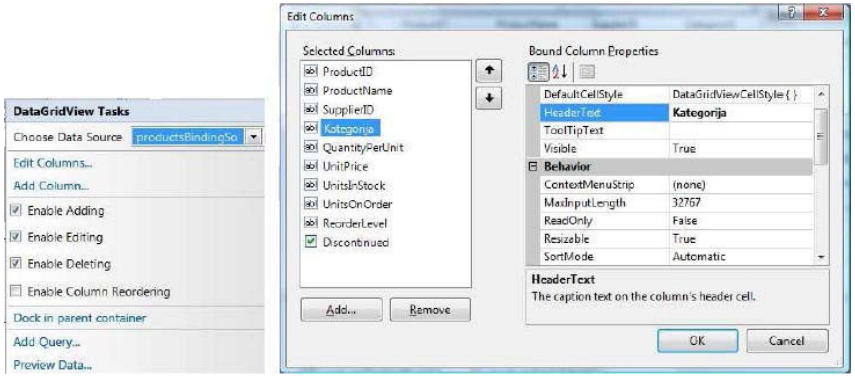


Slika-7: Selektovane ćelije

UBACIVANJE COMBOBOX KONTROLE U ĆELIJU DATAGRIDVIEW KONTROLE

Najpre se izvrši povezivanje DataGridView kontrole sa tabelom iz baze podataka.

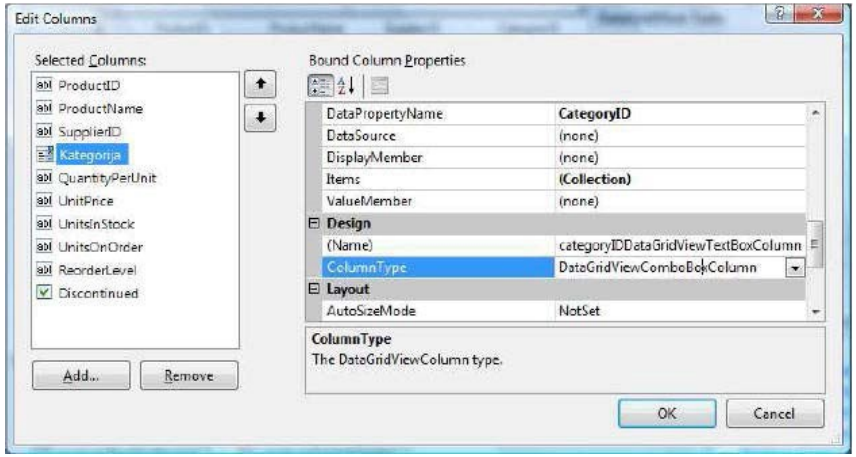
Najpre se izvrši povezivanje DataGridView kontrole sa tabelom iz baze podataka. Zatim se klikom na strelicu koja se nalazi u gornjem desnom uglu DataGridView kontrole otvara prozor DataGridView Tasks. Sada se u ovom prozoru odabere opcija Edit Columns... Selektuje se kolona kojoj želimo da promenimo tip. U primeru na slici DataGridView prikazuje Proizvode iz baze podataka Nortwind. Tabela proizvoda je povezana sa tabelom kategorija posredstvom atributa CategoryID koji predstavlja ID kategorije. Želimo da korisniku umesto ovog atributa prikažemo naziv kategorije kome proizvod pripada. Za početak će mo promeniti Header-ove kolone u kategorije.



Slika-8: Podešavanja

Promena tipa kolone

Sada ćemo koloni Kategorije promeniti tip. Unesto DataGridViewTextBoxColumn tipa kolone, postavićemo DataGridViewComboBoxColumn kolonu. Sada ovu kolonu treba povezati sa izvorom podataka, da bi podaci u tabeli proizvoda bili koegzistentni, odnosno da bi promena tipa kolone bila moguća.



Slika-9: Podešavanja tipa kolone

UBACIVANJE COMBOBOX KONTROLE U ĆELIJU DATAGRIDVIEW KONTROLE DRUGI DEO

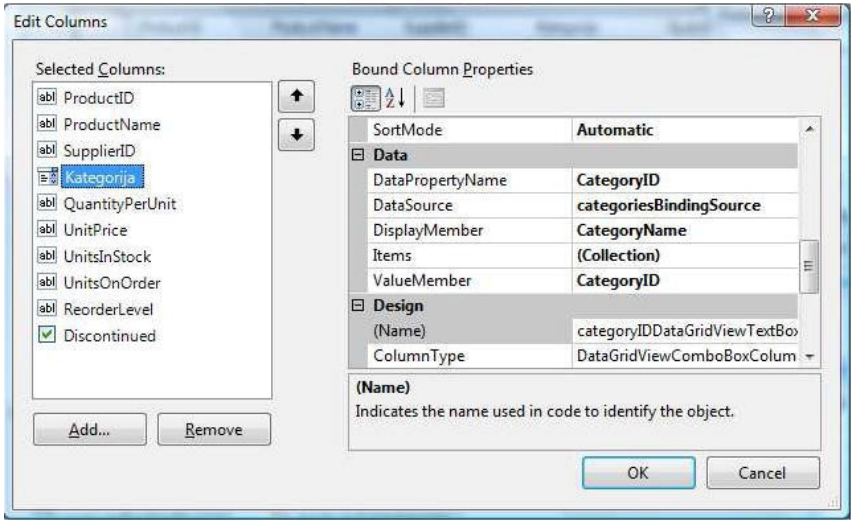
Najpre se izvrši povezivanje DataGridView kontrole sa tabelom iz baze podataka.

Povezivanje ComboBox kolone sa izvorom podataka

Najpre se definiše DataSource svojstvo ComboBox kolone tj. tabela Categories iz DataSeta.

Ovime se automatski kreira odgovarajući BindingSource objekat koji komunicira sa tabelomkategorija. Kao svojstvo DisplayMember postavi se kolona CategoryName tj. naziv kategorije, I to je vrednost koja će biti prikazana u ComboBox koloni. Kao svojstvo ValueMember postavi se kolona CategoryID, čime se omogućava veza sa starom TextBox kolonom.

Slika:



Slika-10: Podešavanja prikazne vrednosti

KLASA DATAADAPTER

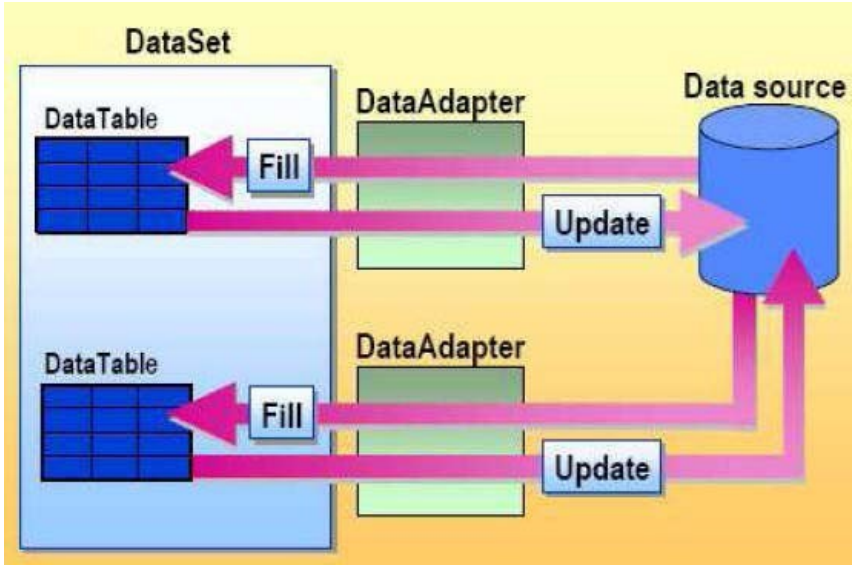
Klasa DataAdapter predstavljamost između podataka i objekta DataSet

Klasa DataAdapter predstavljamost između podataka i objekta DataSet. Objekat DataSet se koristi za čuvanje lokalne kopije podatakaiz baze. Pri čitanju podataka iz baze objekat DataAdapter prima podatke od objekta Connection i prosleđuje ih objektu DataSet. Promene na podacima iz objekta DataSet prosleđuje nazad do objekta Connection kako bi se podaci ažurirali u samom izvoru podataka.

RAD U DISKONEKTOVANOM OKRUŽENJU

Klasa DataAdapter predstavljamost između podataka i objekta DataSet

Slika



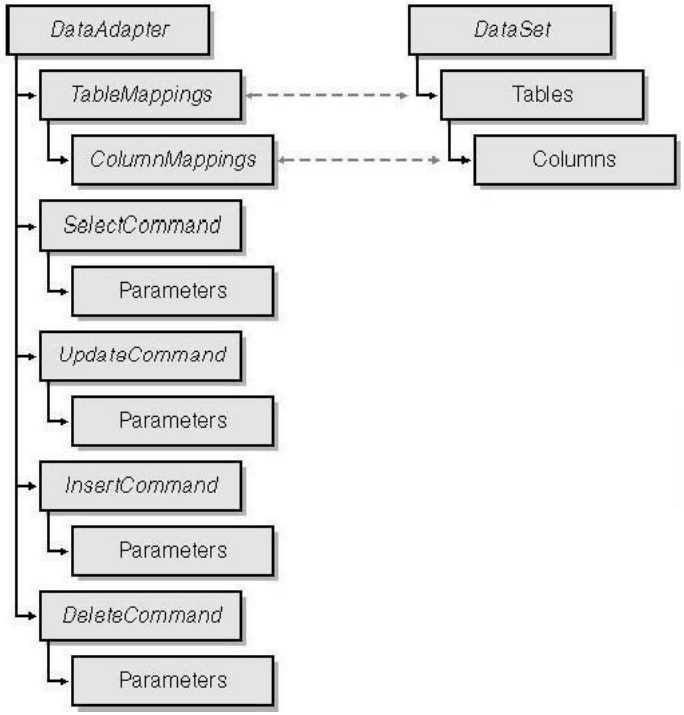
Slika-11: Primena ADO.Net

Na slici je ilustrovana primena primena pojedinih ADO.NET objekata pri radu u diskonektovanom okruženju. Objekat DataSet sadrži jednu ili više lokalnih kopija tabela iz baze podataka. Za svaku tabelu u DataSet-u se kreira po jedan objekat DataAdapter koji kao što je već receno služi kao most između izvora podataka i njegove lokalne kopije koja se nalaz unutar objekta DataSet. DataAdapter objekat predstavlja skup SQL komandi i konekcije na bazu koja se koristi da se napuni objekat DataSet podacima iz baze i da se promenama izvršenim na objektu DataSet ažurira baza podataka. Dva osnovna DataAdapter-a koja su integrisana u Visual Studio.NET su OleDbDataAdapter i SqlDataAdapter. Izvor adaptera zavisi od baze sa kojom se radi odnosno od snabdevača podataka koji se koristi.

STRUKTURA DATAADAPTERA

Klasa DataAdapter predstavljamost između podataka i objekta DataSet

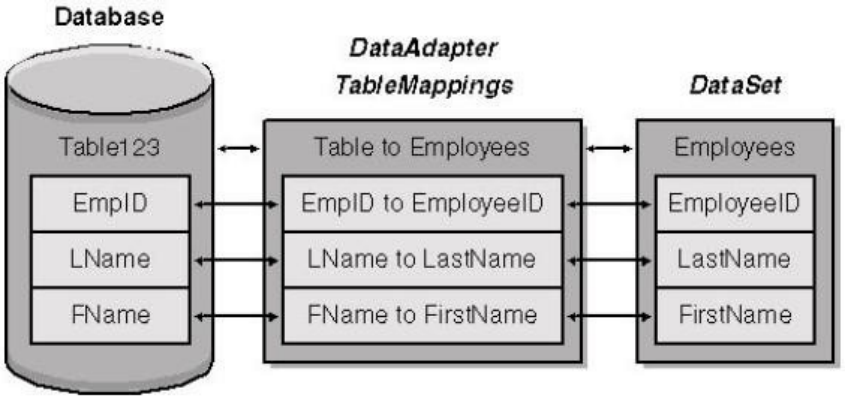
Slika



Slika-12: Struktura Data adaptera

Objekat DataAdapter sadrži SelectCommand objekat, InsertCommand objekat, UpdateCommand objekat i DeleteCommand objekat. DataAdapter takođe sadrži kolekciju TabelMapings koja služi za mapiranje naziva tabele i njihovih atributaiz baze podataka u odgovarajuć tabele u DataSet-u.

Podrazumevano su nazivi kolona u bazi podataka i odgobarajućem DataTable objektu koji se nalazi u DataSet-u identični. Na slici ispod prikazano je mapiranje tabele Table123 u tabelu Employees u DataSet-u.



Slika-13: Mapping

KARAKTERISTIKE OBJEKTA DATAADAPTER

Karakteristike objekta DataAdapter

Karakteristike objekta DataAdapter

Koristi Command objekteda komunicira sa bazom. SelectCommand predstavlja komandu kojom se čitaju podaci iz baze. Insert, Update i Delete se koristi da se promene nad podacima u DataSet-u iskoriste za ažuriranje podataka u bazi. Pri kreiranju DataAdapter objekta mora se specificirati SELECT komanda.

Svojstva i metode DataAdaptera

Svojstvo SelectCommand objekta klase SqlDataAdapter (ili OleDbDataAdapter) vraća SqlCommand objekat koji predstavlja SELECT komandu za čitanje zapisa iz baze podataka. Slično svojstva InsertCommand, UpdateCommand i DeleteCommand vraćaju odgovarajuće SqlCommand objekte koji predstavljaju

INSERT, UPDATE i DELETE komandu. Metoda Fill objekta SqlDataAdapter klase se koristi za punjenje tabele unutar DataSet objekta podacima iz baze. Metoda Update se koristi za ažuriranje baze podataka promenama u DataSet objektu.

Kreiranje objekta DataAdapter

Kreira se pozivom jednog od konstruktora:

```
SqlDataAdapter     O
{
}

SqlDataAdapter     (SqlCommand    mySqlCommand)
{
}

SqlDataAdapter     (string   selectCommandString,     SqlConnection     mySqlConnection)
{
}

SqlDataAdapter(string     selectCommandString,     string    connectionString)
{
}
//Kreiranje    Adaptera   1
//kreira   se   konekcija   za   svaki   adapter
SqlDataAdapter     da1  =  new  SqlDataAdapter(sqlSelectUpit,     konekcioniString);     SqlDataAdapter     da2  =  new  SqlDataAdapter(sqlSelectUpit,
mojaSqlKonekcija);     SqlDataAdapter     da3  =  new  SqlDataAdapter(komanda);
SqlDataAdapter     da4  =  new  SqlDataAdapter();
da4.SelectCommand     =  komanda;
```

U navedenom kodu se vidi kreiranje objekta SqlDataAdapter korišćenjem SELECT naredbe. Ilustrovana je upotreba različitih konstruktora ove klase. U prvom slučaju konstruktoru klase SqlDataAdapter se prosleđuje SELECT upit i odgovarajući konekcioni string. U drugom slučaju konstruktoru se prosleđuje SELECT upit i odgovarajući SqlConnection objekat. U trecem slučaju je najpre kreiran SqlCommand objekat koji predstavlja SELECT naredbu a zatim je on prosleđen kao parametar konstuktoru. U četvrtom slučaju poziva se podrazumevani konstruktor, a odgovarajućem SqlDataAdapter objektu se preko svojstva SelectCommand dodeljuje odgovarajući SqlCommand objekat koji predstavlja SELECT komandu.

PUNJENJE DATASET-A

Karakteristike objekta DataAdapter

U fragmentu koda ispod prikazano kreiranje DataAdapter objekta korišćenjem SELECT komande kao i njegova upotreba za punjenje odgovarajućeg DataSet-a. Da bi se pozvala Fill metoda DataAdapter-a najpre se mora otvoriti konekcija. Posle poziva Fill metode konekcija se zatvara. Prvi parametar Fill metode je naziv DataSet- a a drugi naziv tabele koju treba kreirati u DataSet-u.

```
SqlCommand mojaKomanda = new SqlCommand(); mojaKomanda.Connection = mojaKonekcija();
mojaKomanda.CommandText =
"SELECT TOP 5 ProductID, productName, UnitPrice " + "FROM Products " + "ORDER BY
ProductID";
SqlDataAdapter mojDataAdapter = new SqlDataAdapter();
mojDataAdapter.SelectCommand = mojaKomanda;
DataSet mojDataSet = new DataSet();
mojaKonekcija.Open();
int brojVrsta = mojDataAdapter.Fill(mojDataSet, “Products“);
mojaKonekcija.Close();
```

Punjenje DataSet-a korišćenjem uskladištene procedure

Najpre se kreira SqlCommand objekat kome se kao parametri prosleđuju naziv uskladištene procedure i odgovarajući konekcioni string. Specificira se CommandType svojstvo objekta na vrednost CommandType.StoredProcedure.

Kreira se SqlDataAdapter objekat primenom podrazumevanog konstruktora. Za SelectCommand svojstvo objekta klase SqlDataAdapter postavi se gore kreirani objekat SqlCommand.

Kod:

```
SqlCommand mojaSqlKomanda = new SqlCommand(); mojaSqlKomanda.Connection =
mojaSqlKonekcija; mojaSqlKomanda.CommandType =
CommandType.StoredProcedure; mojaSqlKomanda.CommandText = "CustOrderHist";
SqlParameter parametar = mojaSqlKomanda.Parameters.Add("@CustomerID",
SqlDbType.NvarChar, 5);
parametar.Value = "ALFKI";
SqlDataAdapter mojSqldataAdapter = new SqlDataAdapter();
mojSqlDataAdapter.SelectCommand = mojaSqlKomanda; DataSet mojDataSet = new
DataSet(); mojaSqlKonekcija.Open();
int numberOfRows = mojSqlDataAdapter.Fill(mojdataSet, "CustOrderHist");
mojaSqlKonekcija.Close();
DataTable myDataTable = mojDataSet.Tables("CustOrderHist");
```

Sada je SELECT komanda DataAdapter-a uskladištena procedura koja sadrži parametre. Punjenje DataSet-a je identično kao i u slučaju korišćenja SELECT komande.

DATASET

Karakteristike objekta DataAdapter

DataSet je memorijska predstava podataka, koja uključuje tabele, relacije izmenu tabela I ograničenja. Objekat DataTable se koristi da bi se predstavila tabela u DataSet-u. Konekcija sa bazom podataka nije neophodna da bi se manipulisalo sa podacima u DataSet-u. Podaci u DataSet-u se čuvaju na sličan način kao što se čuvaju u relacionoj bazi podataka. Podaci iz DataSet-a mogu se prikazati u XML formatu.

Svojstva objekta DataSet

Klasa DataSet ima Tables svojstvo koje daje koekciju DataTable objekata u DataSet- u. DataSet ima svojstvo Relations koje daje kolekciju objekata DataRelation koji se

koriste za opis veza izmenu tabela u DataSet-u. Objekat DataTable ima sledeće kolekcije:

- Columns (kolekcija objekta DataColumn)
- Rows (kolekcija objekta DataRow)
- Constraints (kolekcija objekta Constrant)
- ChildRelations (kolekcija objekta DataRelation)

Kreiranje objekta DataSet

Fragment koda ispod prikazuje kreiranje objekta DataSet. Ako se konstruktoru ne prosledi ime DataSet-a onda će mu biti dodeljeno podrazumevano ime NewDataSet. Zatim se instancira DataTable objekat i dodaje u kolekciju Tables DataSet objekta. Kada se prvi put kreira DataTable objekat on nema šemu. Zato se moraju kreirati objekti klase DataColumn i dodati u Columns kolekciju objekta klase DataTable.Objekat. DataColumn se može kreirati pozivom DataColumn konstruktora kao što je to ilustrovano unutar drugog pravougaonika. Drugi način je direktno dodavanje naziva kolone i tipa podataka za kolonu u Columns kolekciju objekta DataTable kao u trećem pravougaoniku. Ako vrednosti NULL nisu dozvoljene za kolonu onda se to specificira korišćenjem svojstva AllowDBNull i njegovim setovanjem na false. Ako želimo da vrednosti u koloni imaju jedinstvene vrednosti onda se to definiše korišćenjem svojstva Unique.

```
DataSet ds = new DataSet("Northwind"); DataTable dt = new
DataTable("MojaTabela"); ds.Tables.Add(dt);
DataColumn colCustomerID = new DataColumn("CustomerID",
typeof(Int32));
dt.Columns.Add(colCustomerID);
colCustomerID = dt.Columns.Add("CustomerID", typeof(Int32));
colCustomerID.AllowDBNull = false;
```

KREIRANJE PROSTOG PRIMARNOG KLJUČA

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ.

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ. Kada se primarni ključ sastoji samo od jednog atributa tabele radi se o prostom primarnom ključu. Svojstvu PrimaryKey objekta DataTable mora se dodeliti niz objekata klase DataColumn. U slučaju prostog ključa ovaj niz se sastoji samo od jednog člana. Drugi način kreiranja primarno ključa je korišćenje Constraints kolekcije objekta DataTable. Prvi parametar Add metode je naziv ograničenje, drugi parametar je naziv kolone koja predstavlja primarni ključ, a treći parametar mora imati vrednost true da bi se naznačilo da se radi o primarnom ključu.

```
myDataTable.PrimaryKey = new DataColumn[]
{
myDataTable.Columns["CustomerID"]
};
Constraint pkCustomers =
dtCustomers.Constraints.Add("PK_Customers",dtCu
stomers.Columns[CustomerID], true);
```

Kreiranje složenog primarnog ključa

U sledećem kodu prikazano je kreiranje složenog primarnog ključa. Složeni primarni ključ sastoji se od dva ili više atributa tabele nad kojom je definisan. Kreiranje složenog ključa je slično kreiranju prostog primarnog ključa.

```
dtEmployess.PrimaryKey = new DataColumn[]
{
dtEmployees.Colums["FirstName"], dtEmployees.Colums["LastName"]
};
mojDataSet.Tables["Order Details"].Constraints.Add("PK_OrderDetails", new
DataColumn[]
{
mojDataSet.Tables["Order Details"].Columns["OrderID"],
mojDataSet.Tables["Order Details"].Columns["ProductID"]
}, true);
}
```

Ovo ograničenje označava da vrednost atributa ili grupe atributa nad kojim je definisano ovo ograničenje mora biti jedinstvena. Kreira se instanciranjem klase UniqueConstraint. U gornjem kodu su prikazana dva konstruktora ove klase. Prvi se može iskoristiti za kreiranje tzv. Složenog Unique ograničenja a drugi konstruktor samo za kreiranje prostog Unique ograničenja, tj. ograničenja nad jednim atributom tabele.

```
public UniqueConstraint(string name, DataColumn[] columns,
bool isPrimaryKey);
public UniqueConstraint(string name, DataColumn column);
ds.Tables[“Products”].Constraints.Add(new UniqueConstraint
("UC_ProductName",
ds.Tables["Products"].Columns["ProductName"]);
```

OGRANIČENJE FOREIGNKEY

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ.

Da bi se u tabeli koja se nalazi unutar DataSet-a kreiralo ograničenje strani ključ DataSet mora imati najmanje dva tabele. Roditeljska tabela mora imati primarni ključ.

U kodu je prikazan konstruktor klase ForeignKeyConstraint. Potrebno je definisati naziv ovog ograničenja, kolonu primarnog ključa u roditeljskoj tabeli i kolonu stranog ključa u tabeli potomku.

```
public ForeignKeyConstraint(string constraintName,
DataColumn parentColumn, DataColumn childColumn );
```

U sledećim kodovima su data dva načina kreiranja stranog ključa. Prvi je nstanciranje klase ForeignKeyConstraint i dodavanje objekta u Constraints kolekciju:

```
ForeignKeyConstraint mojFK = new
ForeignKeyConstraint("FK_Orders_OD",
mojDataSet.Tables["Orders"].Columns["OrderID"],
mojDataSet.Tables["Order
Details"].Columns["OrderID"]);
mojDataSet.Tables["Order
Details"].Constraints.Add(mojFK);
```

Drugi način je direktno dodavanje stranog ključa korišćenjem metode Add kolekcije Constraint:

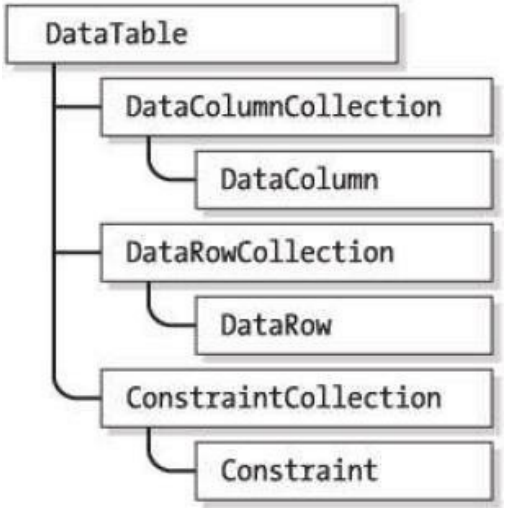
```
mojDataSet.Tables["Order
Details"].Constraints.Add( "FK_Pr_OD"
,mojDataSet.Tables["Products"].Columns["Product
ID"], mojDataSet.Tables["Order
Details"].Columns["ProductID"]);
```

Potrebno je definisati naziv ovog ograničenja, kolonu primarnog ključa u roditeljskoj tabeli I kolonu stranog ključa u tabeli potomku.

STRUKTURA OBJEKTA DATATABLE

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ.

Objekat DataTable sadrži kolekciju DataColumnCollection koja sadrži objekte klase DataColumn koji predstavljaju odgovarajuće kolone tabele. Takođe sadrži kolekciju dataRowCollection koja sadrži objekte klase DataRow (predstavljaju redove tabele) kao I kolekciju ConstraintCollection koja sadži objekte klase Constraint (predstavljaju ograničenja definisana nad tabelom).



Slika-14: DataTable struktura

Punjenje objekta DataTable korišćenjem DataAdaptera

Objekat DataTable koji se nalazi unutar objekta DataSet se može napuniti korišćenjem Fill metode DataAdapter objekta. Metoda Fill DataAdaptera izvršava njegovu SELECT komandu. U sledećem kodu Fill metode se u DataSet-u kreira tabela Customers i puni podacima iz baze podataka:

```
mySqlDataAdapter.SelectCommand = mySelectCommand;
DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Fill(myDataSet,
"Customers");
mySqlConnection.Close();
```

PUNJENJE DATASETA KORISTEĆI VIŠE ADAPTERA

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ.

Za punjenje DataSet-a može se koristiti više DataAdaptera. Svaki DataAdapter puni odrenenu tabelu u DataSetu. Pored DataSet-ova koji se kreiraju instanciranjem klase DataSet i koji se nazivaju netipizirani DataSet-ovi, postoje i tzv. tipizirani DataSet-ovi. Tipski DataSet-ovi se kreiraju na osnovu već postojećeg izvora podataka i pridružena im je XML šema. Tipski DataSet se može kreirati grafički u okruženju Visual Studio.NET. Najpre se kreira odgovarajući DataAdapter npr. SqlDataAdapter prevlačenjem SqlDataAdapter kontrole koja se nalazi u Data delu Toolbox-a. Na ovaj način se kreira objekat označen sa sqlDataAdapter1. Sada se klikne desnim tasterom miša na ovaj objekat i izabere se opcija Generate DataSet. Na ovaj način se DataSet objektu pridružuje xsd šema koja predstavlja xml dokument koji opisuju strukturu tabele iz koje DataAdapter čita podatke. Tipizirani DataSet je izveden iz klase DataSet pa zadržava sve osobine i metode netipiziranog DataSet-a. Kod tipiziranog DataSeta znatno je olakšan pristup njegovim članovima npr. tabelama se pristupa direktno a ne preko Tables kolekcije.

Kod:

```
Netipski DataSet
// vise data adaptera pune jedan DataSet
myDataAdapter.Fill(myDataSet, "Customers");
myDataAdapter1.Fill(myDataSet, "Employees");
Tipski DataSet
daCustomers.Fill(dsCustOrders.Customers);
daOrders.Fill(dsCustOrders.Orders);
```

Primer pristupa atributu kod tipiziranog i netipiziranog DataSeta

U fragmentu koda je prikazan način pristupa atributu CategoryName u prvoj vrsti tabele Categories za slučaj tipiziranog i netipiziraog DataSet-a.

```
// netipizirani
DataRow myRow =
dsNorthwind.Tables["Categories"].Rows[0];
string categoryName = (string) myRow
["CategoryName"];
// tipizirani
String categoryName =
dsNorthwind.Categories[0].CategoryName;
```

PRONALAŽENJE VRSTE U TABELI

Za svaki objekat DataTable koji se nalazi unutar DataSet-a potrebno je definisati primarni ključ.

Pronalaženje konkretnog DataRow objekta u objektu DataTable sastoji se iz nekoliko koraka:

- Korišćenjem objekta DataAdapter puni se tabela podacima.
- Kreira se primarni ključ za tabelu ukoliko se radi o netipiziranom DataSetu.
- Ako je DataSet tipiziran primarni ključ objekta DataTable postoji ako postoji ključ za odgovarajuću tabelu u bazi.
- Poziva se Find() metoda kolekcije DataRowCollection.

Pretraživanje po prostom ključu

Najpre se iz DataSet-a uzme odgovarajući objekat DatTable. Definiše se primarni ključ objekta DataTable ukoliko se radi o netipiziranom DataSet-u. Svostvo Rows objekta DataTable vraća DataRowCollection kolekciju koja sadrži kolekciju objekata DataRow. Pronalazi se DataRow objekat korišćenjem Find metode DataRowCollection kolekcije čija je vrednost ključa jednaka zadatoj vrednosti. Na kraju se iščitavaju atributi te vrste.

Pretraživanje po složenom ključu

Kreira se object niz čije su vrednosti jednake vrednostima atributa koji ulaze u sastav primarnog ključa. Potrebna vrsta nalazi se korišćenjem Find metode kolekcije DataRowCollection. Kada se pronane željena vrsta pristupa se iščitavanju njenih atributa.

Filtriranje i sortiranje DataRow objekata u objektu DataTable

Deklaracija metode Select objekta klase DataTable.

Ova metoda vraća niz objekata klase DataRow odnosno skup zapisa koji zadovoljavaju specificirani kriterijum. Ako je productDataRows naziv niza objekata DataRow koji je dobijen izvršavanjem Select naredbe objekta DataTable tada se atributi svake vste u rezultujućem skupu zapisa mogu dobiti korišćenjem dve foreach petlje:

```
foreach (DataRow myDataRow in productDataRows)
{
foreach (DataColumn myDataColumn in productsDataTable.Columns)
{
Console.WriteLine(myDataColumn + "= " + myDataRow[myDataColumn]);
```

MODIFIKOVANJE PODATAKA U BAZI KORIŠĆENJEM DATAADAPTERA

Potrebno je pored SELECT komande DataAdaptera (koja služi za čitanje podataka iz baze i punjenje DataSet-a), definisati i INSERT, UPDATE i DELETE komande.

Potrebno je pored SELECT komande DataAdaptera (koja služi za čitanje podataka iz baze i punjenje DataSet-a), definisati i INSERT, UPDATE i DELETE komande. Promene u DataSet-u se proslenuju do baze pozivanjem Update() metode DataAdapter-a.

Definisanje insert komande DataAdaptera

Potrebno je kreirati SqlCommand objekat koji će predstavljati INSERT naredbu. Osim poziva konstruktora objekat SqlCommand je moguće kreirati i pozivom metode CreateCommand odgovarajućeg SqlConnection objekta. Uobičajeno je korišćenje parametarskog INSERT upita gde će vrednosti parametara definisati korisnik. Posle dodavanja parametara InsertCommand svojstvu objekta klase DataAdapter pridružuje gore kreirani SqlCommand objekat.

Primer:

```
// INSERT KOMANDA
SqlCommand myInsertCommand = mySqlConnection.CreateCommand();
myInsertCommand.CommandText =
"INSERT INTO Customers(CustomerID, CompanyName, Address ) VALUES (@CustomerID,
@CompanyName, @Address)" ;
myInsertCommand.Parameters.Add("@CustomerID",
SqlDbType.NChar,5,"CustomerID");

myInsertCommand.Parameters.Add("@CompanyName",SqlDbType.NVarChar,40,"CompanyName");
myInsertCommand.Parameters.Add("@Address", SqlDbType.NVarChar,60,"Address");
```

Ubacivanje zapisa u bazu podataka

Poziva se NewRow() metoda objekta DataTable čime se kreira DataRow objekat koji ima istu strukturu kao i tabela. Za objekat DataRow definišu se vrednosti za atribute te vrste. Ovako ažurirani objekti DataRow se dodaju u Rows kolekciju tabele. Otvara se konekcija sa bazom podataka. Poziva se Update metoda DataAdaptera i proslenuje mu se kao parametar objekat DataTable koji je promenjen. Na ovaj način se promene nastale u DataSetu upisuju trajno u bazu podataka. Pri završenom ažuriranju potrebno je zatvoriti konekciju.

Primer:

```
// Korak 1: koriscenje NewRow() metoda objekta
DataTable
// za kreiranje nove vrste- DataRow
DataRow novaVrsta = mojaTabela.NewRow();
// Korak 2: postavi vrednosti za DataColumn objekte
// za novu vrstu - DataRow novaVrsta["CustomerID"] =
CustID; novaVrsta["CompanyName"] = Company;
novaVrsta["Address"] = addr;
```

UPDATE KOMANDA DATAADAPTERA

U kodu je prikazano kreiranje Update komande DataAdaptera. Ukoliko se DataAdapter kreira grafički u okruženju Visual Studio.NET onda se INSERT, UPDATE i DELETE

U kodu je prikazano kreiranje Update komande DataAdaptera. Ukoliko se DataAdapter kreira grafički u okruženju Visual Studio.NET onda se INSERT, UPDATE i DELETE komanda kreira automatski.

```
SqlCommand myUpdateCommand = mySqlConnection.CreateCommand();
myUpdateCommand.CommandText =
"UPDATE Customers SET CompanyName = @NewCompanyName, Address =
@NewAddress WHERE CustomerID = @OldCustomerID";

myUpdateCommand.Parameters.Add("@NewCompanyName",SqlDbType.NVarChar,40,"CompanyName");
myUpdateCommand.Parameters.Add("@NewAddress", SqlDbType.NVarChar,60,"Address");
myUpdateCommand.Parameters.Add("@OldCustomerID", SqlDbType.NChar,5,"CustomerID");
mySqlDataAdapter.UpdateCommand = myUpdateCommand;
```

Ažuriranje podataka u bazi

Ukoliko se radi o netipiziranom DataSetu mora se najpre definisati primarni ključ DataTable objekta na osnovu koga će se vršiti pretraživanje. Korišćenjem Find metode kolekcije DataRowCollection pronalazi se DataRow objekat sa željenom vrednošću primarnog ključa. U pomenutom redu se izvrši promena željenih atributa tako što se atributu unutar reda pristupa preko imena ili indeksa. Pozove se Update metoda DataAdaptera i prosledi joj se ažurirani objekat DataTable.

Primer:

```
// korak 1: postavi PrimaryKey svojstvo za objekat
DataTable mojaTabela.PrimaryKey = new DataColumn[]
{
mojaTabela.Columns["CustomerID"]
};
// korak 2: koriscenjem Find() metode lociraj DataRow
```

Delete komanda DataAdaptera

U ovom kodu je dat primer DELETE komande DataAdptera. Uobičajeno je da se pronalaženje vrste za brisanje vrši po primarnom ključu.

```
myDeleteCommand.CommandText =
"DELETE FROM Customers WHERE CustomerID = @OldCustomerID";
myDeleteCommand.Parameters.Add("@OldCustomerID",SqlDbType.N
Char,5,
"CustomerID");
mySqlDataAdapter.DeleteCommand = myDeleteCommand;
```

BRISANJE PODATAKA U BAZI

Prvi korak je definisanje primarnog ključa objekta DataTable ukoliko se radi o netipiziranom DataSetu.

Prvi korak je definisanje primarnog ključa objekta DataTable ukoliko se radi o netipiziranom DataSetu. Drugi korak je korišćenje Find metode kolekcije vrsta za pronalaženje vrste za specificirani ID. Treći korak je brisanje vrste unutar DataTable objekta tj. brisanje vrste iz lokalne kopije. Četvrti korak je ažuriranje izvora podataka koristeći Update metodu DataAdapter objekta kome se kao parametar proslenuje izmenjeni objekat DataTable. Primer:

```
// Korak 1: postavi PrimaryKey svojstvo za DataTable myDataTable.PrimaryKey = new DataColumn[]
{
myDataTable.Columns["CustomerID"]
};
// Korak 2: koriscenje Find() metode za pronalzenje vrste
DataRow myRemoveDataRow = myDataTable.Rows.Find(id);
// Korak 3: koriscenje metoda Delete() za brisanje vrste
try
{
myRemoveDataRow.Delete();
Console.WriteLine("myRemoveDataRow.RowState = " +
myRemoveDataRow.RowState);
// Korak 4: koriscenje Update() metode za brisanje vrste iz baze
podataka

mySqlConnection.Open();
int numOfRows = mySqlDataAdapter.Update(myDataTable); Console.WriteLine("brojVrsta = " + numOfRows);
Console.WriteLine("myRemoveDataRow.RowState = " +
myRemoveDataRow.RowState);
```

KREIRANJE DATAVIEW OBJEKTA

Objekat DataView sličan je objektu view u SQL Server-u s tim što objekat DataView prikazuje podatke iz samo jedne tabele.

Objekat DataView sličan je objektu view u SQL Server-u s tim što objekat DataView prikazuje podatke iz samo jedne tabele. Sadrži podskup podataka iz objekta DataTable nad kojim je definisan. Kreira se korišćenjem konstruktora DataView tako što mu se kao ulazni parameter prosledi objekat DataTable koji se nalazi u DataSetu:

Svaki objekat DataTable poseduje svojstvo DefaultView:

Ovo svojstvo vraća DataView objekat nad tabelom koji se može iskoristiti za filtriranje i sortiranje podataka u tabeli.

Karakteristike objekta DataView

DataView objekat omogućava filtriranje podataka korišćenjem svojstva RowFilter. Svojstvu RowFilter se pridružuje string koji odgovara WHERE klauzuli prilikom pisanja običnog SELECT upita sa filtriranjem podataka (npr. dvKorisnik.RowFilter = "Country = 'UK'").

Svojstvu Sort se pridružuje string koji odgovara klauzuli ORDER BY prilikom sortiranja podataka u običnom SELECT upitu. Svaka tabela ima DefaultView. Povezivanje podataka u Windows obrascima uvek se radi posredstvom odgovarajućeg view objekta iako se to ne naglašava uvek eksplicitno:

DataViewManager klasa predstavlja view celog DataSet-a. DataGrid je jedina Windows kontrola koja može da vrši povezivanje podataka iz celog DataSeta.

SVOJSTVA OBJEKTA DATAVIEW

Objekat DataView sličan je objektu view u SQL Server-u s tim što objekat DataView prikazuje podatke iz samo jedne tabele.

Svojstvo Table služi da se postavi ili pročita odgovarajući objekta DataTable za koji je View vezan. DataRowView predstavlja view definisan za objekat DataRow. Odnosno DataRowView se može shvatiti kao red unutar DataView objekta.Metoda Find() objekta DataView nalazi vrstu u odgovarajućem objektu DataView sa specificiranom vrednošću atributa po kome se sortira. Metoda Find vraća indeks vrste koja odgovara traženom kriterijumu.

U fragmentu koda ispod dat je primer upotrebe DataView objekta:

```
DataView  dvKorisnik   = new  DataView(myDataSet.Tables["Customers"]);
dvKorisnik.RowFilter   = "Country  =  'UK'";
dvKorisnik.Sort   = "CustomerID   ASC";
int  indeksVrste   = dvKorisnik.Find("ISLAT");    Console.WriteLine(dvKorisnik[indeksVrste][0]);
Console.WriteLine(dvKorisnik[indeksVrste][1]);
foreach  (DataRowView   mojDataRowView    in dvKorisnik)
{
for  (int count  = 0;  count  < dvKorisnik.Table.Columns.Count;    count++)
{
Console.WriteLine(mojDataRowView[count]);
}
Console.WriteLine("");
}
```

Kreiran je objekat DataView nad tabelom Customers koja se nalazi u DataSetu. DataView prikazuje samo one redove tabele Customers kod kojih atribut Country ima vrednost UK. Redovi se prikazuju sortirani po atributu CustomerID u rastućem poretku. Korišćenjem metode Find objekta DataView traži se indeks reda u objektu DataView kod koga je vrednost atributa CustomerID jednaka ISLAT.

Odgovarajućem DataRowView objektu se pristupa korišćenjem izraza dvKorisnik[indeksVrste] gde je indeksVrste vrednost koju je vratila metoda Find. Prvom atributu ovog DataRowView objekta pristupa se korišćenjem izraza dvKorisnik[indeksVrste][0], drugom dvKorisnik[indeksVrste][1] itd. Broj kolona tabele nad kojom je View definisan može se dobiti korišćenjem izraza dvKorisnik.Table.Columns.Count jer svojstvo Table vraća odgovarajući DataTable objekat koji ima Column kolekciju. Broj članova kolekcije Columns se dobija korišćenjem svojstva Count a to je ujedno i broj kolona te tabele odnosno objekta DataView koji je nad njom definisan.

METODA FINDROWS

Objekat DataView sličan je objektu view u SQL Server-u s tim što objekat DataView prikazuje podatke iz samo jedne tabele.

Ova metoda vraća niz objekata DataRowView koji odgovaraju zadatoj vrednosti atributa po kome se vrši sortiranje objekta DataView. Za kretanje kroz ovaj skup DataRowView objekata može se iskoristiti foreach naredba na način predstavljen na slajdu. U primeru ispod se štampa atribut CustomerID za sve vrste iz DataView-a koje odgovarju kriterijumu traženja.

```
DataView pogled2 = new DataView(myDataSet.Tables["Customers"]);
pogled2.Sort = "Country";
DataRowView[] vrste = pogled2.FindRows("Germany");
if (vrste.Length == 0)
{
Console.WriteLine("Nema pronadjenih vrsta.");
}
else
{
foreach (DataRowView vrsta in vrste)
{
Console.WriteLine(vrsta["CustomerID"].ToString() + " odgovara.");
}
}
```

Dodavanje DataRowView objekta u objekat DataView
Dodavanje redova u DataTable objekat je moguće uraditi kroz DataView. To se postiže dodavanjem objekta DataRowView u objekat DataView. Pozivanjem metode AddNew objekta DataView kreira se objekat DataRowView čija je šema identična šemi odgovarajućeg DataView objekta.

Objekat DataRowView ima metodu BeginEdit koja ozbačava da se objekat nalazi u režimu editovanja. Ukoliko želimo da sačuvamo promenu u DataRowView objektu na kraju editovanja pozivamo metodu EndEdit ili CancelEdit ukoliko želimo da poništimo promene.

Primer:

```
DataView dvKorisnik = new DataView(); dvKorisnik.Table = tabelaKorisnik;
dvKorisnik.RowFilter = "Country = 'UK'";
// dodaj novi DataRowView
//-------------------------------------
DataRowView korisnikDRV = dvKorisnik.AddNew();
korisnikDRV.BeginEdit(); korisnikDRV["CustomerID"] = "J7COM";
korisnikDRV["CompanyName"] = "J7 Company"; korisnikDRV["Country"] = "UK";
Console.WriteLine("korisnikDRV.IsNew = " + korisnikDRV.IsNew);
Console.WriteLine("korisnikDRV.IsEdit = " + korisnikDRV.IsEdit);
korisnikDRV.EndEdit();
prikazi(dvKorisnik);
```

EDITOVANJE DATAROWVIEW OBJEKTA KOJI SE NALAZI U DATAVIEW OBJEKTU

Odgovarajućem DataRowView objektu koje se nalazi unutar DataView objekta pristupamo na osnovu indeksa vrste.

Odgovarajućem DataRowView objektu koje se nalazi unutar DataView objekta pristupamo na osnovu indeksa vrste. Indeks može biti definisan ukoliko se unapred zna indeks vrste koju treba editovati ili se može naći na osnovu nekog kriterijuma koristeći metodu Find objekta DataView na već opisani način. Posle pozicioniranja na odgovarajući red pristupa se editovanju željenih atributa npr. dvKorisnik[0]["CompanyName"] ="Link group." tj. atribut CompanyName prve vrste u DataView objektu dobija novu vrednost Link group. Pre početka editovanja poziva se metoda BeginEdit a po završetku editovanja metoda EndEdit.

Primer:

```
// EDITOVANJE PRVE VRSTE dvKorisnik[0].BeginEdit();
dvKorisnik[0]["CompanyName"] = "Link group.";
Console.WriteLine("dvKorisnik[0][\" CustomerID\"] = " +
dvKorisnik[0]["CustomerID"]);
Console.WriteLine("dvKorisnik[0][\" CompanyName\"] = " +
dvKorisnik[0]["CompanyName"]);
Console.WriteLine("dvKorisnik[0].IsNew = " +
dvKorisnik[0].IsNew);
Console.WriteLine("dvKorisnik[0].IsEdit = " +
dvKorisnik[0].IsEdit);
dvKorisnik[0].EndEdit();
prikazi(dvKorisnik);
```

Brisanje DataRowView objekta

Da bi se izbrisao DataRowView objekat iz objekta DataView poziva se Delete metoda objekta DataView i proslenuje joj se kao parametar indeks vrste koju treba obrisati. Indeks može biti definisan ukoliko se unapred zna indeks vrste koju treba obrisati ili se može naći na osnovu nekog kriterijuma koristeći metodu Find objekta DataView na već opisani način.

Primer:

```
dvKorisnik.Delete(1);
Console.WriteLine("dvKorisnik[1].IsNew = " +
dvKorisnik[1].IsNew);
Console.WriteLine("dvKorisnik[1].IsEdit = " +
dvKorisnik[1].IsEdit); prikazi(dvKorisnik)
```

ZADACI ZA SAMOSTALNI RAD

Kreirati bazu podataka po izboru i napraviti DataGridView koji se povezuje na istu

Kreirati bazu podataka po izboru i napraviti DataGridView koji se povezuje na istu. Kreirati klase za svaku od tabela u bazi i napraviti listu podataka koju ćete napuniti podacima iz baze.

Zaključak

ZAKLJUČAK

Zaključak

1.Visual C# je izvanredan za rukovanje podacima. Visual C# ima

ugradjenu *database engine,* mašinu tj alat za kreiranje baze podataka

takodje drajvere za različite servere baza podataka

podaci memorisani pomoću Visual C# mogu biti korišćeni od strane drugih aplikacija npr Microsoft Access ili Microsoft Word

Cilj nam je ovde da naučimo da

pomoću kreiramo aplikacije sa bazama podataka,

i da povezujemo Visual C# forme sa baze podataka, u cilju pretrage i pisanja izveštaja

2. Visual C# omogućuje lako

kreiranje, i rukovanje malih I velikih baza podataka.

Naime, pomoću Visual C# mogu se kreirati #forme za

-pretragu, ažuriranje podataka, ili izveštavanje o podacima.

3. Takodje, može se koristiti programski kod u C#, da se obave proračuni, ili procesiraju veliki nizovi podataka.

ZAKLJUČAK

Zaključak

4.***Data Grid***,

to je veoma korisna i veoma korišćena kontrola tj objekat tj alat koji je raspoloživ u Toolbox-u,

koja prikazuje podatke u obliku mreže podataka, slično kao spreadsheet,

a i omogućuje editovanje tih podataka

5. **ADO.NET** je biblioteka za baze podataka u okviru Visual C#, i ***Visual C#*** obezbedjuje skup drajvera pomoću kojih se može pristupiti raznim tehnologijama baza podataka, npr. *SQL Server*, MySQL, ili **Oracle**, itd.