VISUAL BASIC 6 - Početni kurs

S K R I P T A

Vladimir Tasić

Microsoft Certified Professional

Sadržaj

UVOD

Basic, kao programski jezik je prešao dug razvojni put. Zahvaljujući Microsoft-u postao je profesinalni razvojni alat i standard za razvoj aplikacija u Windows okruženju. U poslednjoj verziji 5.0 donosi mnogo noviteta i značajno poboljšane perfomanse.

Princip objektnog programiranja u Windows-u, koji je dosledno prenet na Visual Basic, se zasniva na tri ključna termina:

1. **Svojstva (Properties)**
2. **Metode (Methods)**
3. **Događaje (Events)**

Ako bi napravili poređenje sa čovekom (u ovom primeru objekt) mogla bi se povući paralela:
Svojstva čoveka bi bile njegove fizičke osobine; visina, težina, boja očiju ...
Metode bi bile njegove sposobnosti u stručnom smislu; sposobnost da projektuje zgradu, popravi automobil, napravi kompjuterski program.
Događaji bi bili načini na koje čovek reaguje na spoljnu sredinu i svoje okruženje.

Ova, pomalo slobodna paralela, se dokazuje u programerskoj praksi. Kada bi hteli da sakupite ekipu stručnjaka koja treba da završi složeni projekat, čiji ste Vi rukovodilac, izvršili bi ste njegovu analizu i zavisno od potreba formirali potrebnu ekipu stručnjaka, podelili im zadatke i snihronizovali ih. Potpuna je ista metodologija prilikom kreiranja programa u Visual Basic-u. Ovde stručnjake predstavljaju objekti sa svojim specijalizovanim skupovima svojstava i metoda, a Vi pišete programski kod koji vrši njihovu sinhronizaciju u međusobnom radu.

Kreirate formular, i postavljate potrebne objekte na njega. Potom za svaki od njih podešavate svojstva i kao odgovor na događaje (koje pokreće korisnik svojim akcijama u programu) pišete potrebni kod.

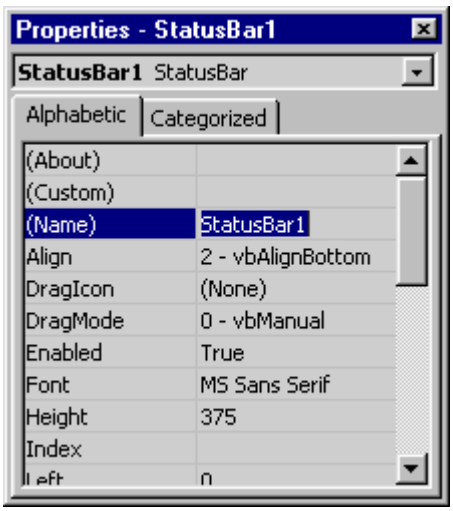
Programiranje vođeno događajima

U standardnim proceduralnim jezicima kao što je C, stareije verzije bejzika, fortran, clipper uvek je postojala neka manje ili više složena ulazna tačka programa. U njoj se ispituju akcije korisnika i u zavisnosti od toga pozivaju funkcije i/ili procedure za izvršenje zadatka. U modernim objektno orijentisanim jezicima u koje spada i Visual Basic, takva ulazna tačke ne postoji. U stvari, postoji i to je sam operativni sistem - Windows. Celokupni program je podeljen na mnoštvo delova koji se izvršavaju kada korisnik izvrši neku akciju.

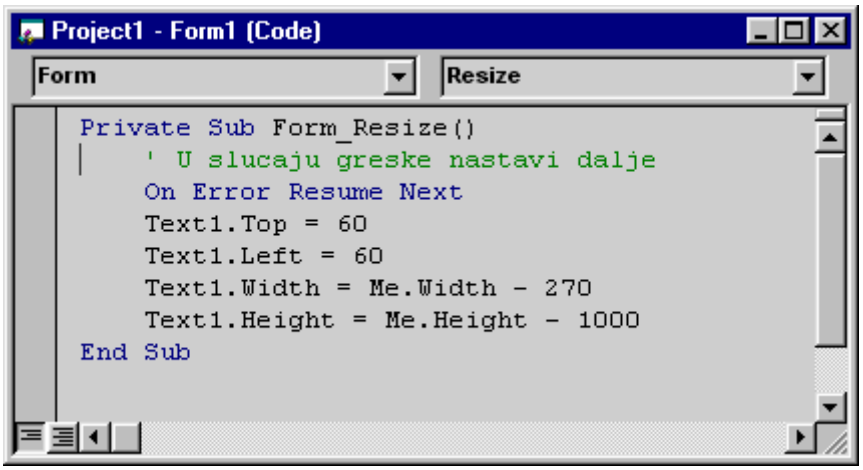
Na primer, akcija bi mogla biti klik na komandno dugme. U tom slučaju komandno dugme je objekt. U njegova svojstva spada tekst koji je ispisan na njemu. Dugme 'prepoznaje' kada korisnik klikne na njega (događaj) i mi za taj događaj (Click) tog objekta (CommandButton) pišemo kod. Ovaj kod će se startovati i izvršiti samo kada korisnik klikne mišem na komadno dugme.

Programiranje vođeno događajima je u stvari manji ili veći broj segmenata programa koje korisnik aktivira svojim akcijama.

Svaki objekt ima svoj set svojstava. Njih podešavamo u Properties prozoru koji dobijamo kada selektujemo željeni objekt i na tastaturi pritisnemo F4 taster kao što je prikazano na slici:



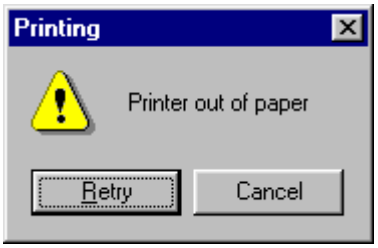
Takođe, svaki objekt prepoznaje neke događaje. Duplim klikom na objekt otvaramo prozor za pisanje koda (Code Window). U ovom prozoru pišemo program za događaj koji selektujemo iz liste (gore desno na slici), mogućih događaja koje podržava dati objekt:

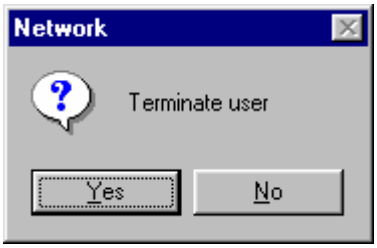


Počinjemo sa nekoliko elementarnih funkcija koje ćemo kasnijekoristiti u primerima.

MsgBox funkcija

Message Box (dijalog za poruke) je često korišćena funkcija Visual Basica. Ove dijaloge ste sigurno već videli u drugim aplikacijama:





Razlikujemo dva tipa dijaloga.
1 informacija korisniku (samo "OK" dugme), prva slika
2 pitanje korisniku (dva ili više dugmeta), od korisnika se očekuje odgovor i na osnovu njega dalje u programu vrši odgovarajuća akcija, druga i treća slika.

U prvom tipu MsgBox pozivamo kao proceduru jer nam nije potrebna informacija koje dugme je korisnik pritisnuo. Sintaksa:

```
MsgBox Prompt, Flags, Naslov
```

- Prompt je tekst ili varijabla koji će biti prikazan u tekstualnom delu dijaloga
- Flags određuju pojavu dijaloga
- Naslov određuje tekst u naslovnoj liniji dijaloga

Flags parametar određuje četiri stvari:
1. Tip ikone
   vbQuestion - znak pitanja
   vbExclamation - znak uzvika
   vbCritical - stop znak
   vbInformation - informacija
   0 - bez ikone

2. Dugniće koji će biti prikazani na dijalogu
   vbOKOnly
   vbOKCancel
   vbAbortRetryIgnore
   vbYesNoCancel
   vbYesNo
   vbRetryCancel

3. Koje dugme po redu, inicijlano ima fokus
   vbDefaultButton1
   vbDefaultButton2
   vbDefaultButton3
   vbDefaultButton4

4. Modalno stanje dijaloga
   vbApplicationModal - korisnik mora zatvoriti dijalog da bi nastavio rad na aplikaciji
   vbSystemModal - sve aplikacije su zaustavljene dok korisnik ne zatvori dijalog

Parametri se kombinuju znakom +. Na primer ako želite da prikažete dijalog kao treći na slici:

```
MsgBox "Terminate user", vbQuestion + vbYesNo, "Network"
```

U drugom tipu MsgBox vraća vrednost (ponaša se kao funkcija). Vraćena vrednost je kod dugmeta koje je korisnik izabrao. Sintaksa poziva je ista, samo parametre treba staviti u zagrade:

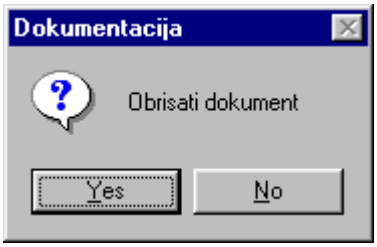
MsgBox (*Prompt*, *Flags*, *Naslov)*

Pošto sada MsgBox vraća vrednost, nephodno je tu vrednost dodeliti promenjivoj ili je iskoristiti u nekom izrazu. Vrednosti koje dobijamo su:

vbOK
vbCancel
vbAbort
vbRetry
vbIgnore
vbYes
vbNo

Na primer:

```
Dim Izlaz as Integer
Izlaz = MsgBox ("Obrisati dokument", vbQuestion + vbYesNo, "Dokumentacija")
If Izlaz = vbNo Then
        ' korisnik je izabrao No - ne brisemo dokument
End If

If Izlaz = vbYes Then
        ' korisnik je izabrao Yes - brisemo dokument
End If
```



InputBox funkcija

InputBox je sistemska funkcija koju obezbeđuje Windows i koja omogućava korisniku da unese odgovarajući tekst i potom potvrdi (OK dugme) ili odustane (Cancel dugme). InputBox je funkcija koja vraća tekst koji je korisnik uneo. U slučaju da nije uneo ništa ili odustao od dijaloga biće vraćen prazan tekst.

Sintaksa je:
`InputBox` (*`Prompt, Naslov, Inicijalni tekst`*)

- *Prompt* je tekst koji daje informaciju koriniku šta se traži od njega
- *Naslov* je tekst koji se pojavljuje na naslovnoj liniji dijaloga
- *Inicijalni tekst* je inicijalna (default) vrednost koja se nudi korisniku. Može se izostaviti, pa je tada inicijalna vrednost prazan tekst.

NAPOMENA:

Radi kompatibilnosti sa ranijim verzijama VB-a ostavljena je InputBox$ funkcija koja radi potpuno isto kao InputBox.

Varijable i njihovi tipovi

Visual Basic podržava sve standardne tipove varijabli. U narednoj tabeli su dati svi tipovi:

Tip	Zauzeće (bajtova)	Opseg
Byte	1	0 - 255 (celi brojevi)
Boolean	2	**True** ili **False (Tačno** ili **Netačno)**
Integer	2	-32,768 do 32,767
Long (long integer)	4	-2,147,483,648 to 2,147,483,647
Single	4	-3.402823E38 to -1.401298E-45 za negativne vrednosti 1.401298E-45 to 3.402823E38 za pozitivne vrednosti
Double	8	-1.79769313486232E308 do -4.94065645841247E-324 (-) 4.94065645841247E-324 do 1.79769313486232E308 (+)
Currency	8	-922,337,203,685,477.5808 do 922,337,203,685,477.5807
Decimal	14	+/-79,228,162,514,264,337,593,543,950,335 celi brojevi +/-7.9228162514264337593543950335 (28 decimala sa desne strane decimalne tačke)
Date	8	1 Januar 100 do 31 Decembar 9999
Object	4	32-bitna referanca na bilo koji objekt
String (varijabilne dužine)	10 + dužina stringa	do oko 2 milina karaktera
String (fiksne dužine)	dužina stringa	1 do 65400 karaktera
Variant (sa brojevima)	16	bilo koja numerička vrednost do opsega za Double tip
Variant (sa karakterima)	22 + dužina stringa	do opsega String-a varijabilne dužine

Korisnički definisani tipovi

Osim standardnih tipova promenjivih moguće je definisati korisnički tip koji je sastavljen od standardnih tipova. Definicija:

```
Type ImeKorisničkogTipa
      varijable
      ...
End Type
```

```
Na primer:
Type Racunar
      Procesor As String
      Frekvencija As Long
      Cena As Currency
End Type
```

Posle ovoga tip promenjive Racunar je ravnopravan sa ostalim tipovima i može se iskoristiti u deklaraciji varijable:

```
Dim MojRacunar As Racunar
```

Članovima složenog tipa Racunar se pristupa pomoću dot (tačka) operatora:

```
MojRacunar.Procesor = "486"
MojRacunar.Frekvencija = 50
MojRacunar.Cena = 0
```

Dužina složenog tipa varijable je jednaka zbiru dužina članova.

Odlučivanje

Visual Basic poseduje standardne strukture za odlučivanje u toku programa.

If ... Then

Sintaksa 1:
```
If Uslov Then Naredba
```

Sintaksa 2:
```
If Uslov Then
      Blok naredbi
end if
```

Sintaksa 3:
```
If Uslov 1 Then
      Blok naredbi 1
ElseIf Uslov 2 Then
      Blok naredbi 2
.
.
ElseIf Uslov N Then
      Blok naredbi N
Else
      Blok naredbi N+1
End if
```

Uslov predstavlja logički izraz koji može imati vrednost tačno (True) ili netačno (False). U slučaju da je Uslov tačan izvršava se naredba ili blok naredbi iza ispitivanja. U suprotnom prelazi se na sledeću liniju koda. Opciono, u višestrukim ispitivanjima, može postojati ELSE deo koji se izvršava samo ako ni jedan od predhodnih uslova nije tačan.

Jedan uslov može biti i složena kombinacija uslova koji se vezuju logičkim operatorima **Or** (ili), **And** (i), **NOT** (negacija).

Na primer:
```
If a>5 then
      ' kod ako je a veće od 5
Else if a<5 Or b>10
      ' kod kada je a manje od 5 i b veće od 10
Else
      ' kod kada nisu zadovoljena oba gornja uslova
End If
```

Select Case

Modernija varijanta If .. Then konstrukcije:

```
Select Case Parametar

      Case V1
            Blok naredbi 1
      Case V2
            Blok naredbi 2


      Case Vn
            Blok naredbi n

      Case Else
            Blok naredbi n+1
End Select
```

Zavisno od vrednosti parametra izvršiće se odgovarajući blok naredbi. Uz Case može stojati i kombinacija sa ključnim rečima **Is** i **To:**

```
Select Case Broj           ' Zavisno od broja
      Case 1 To 5
            ' Broj Između 1 i 5 (uključujući)
      Case 6, 7, 8     '
            ' Broj je ili 6 ili 7 ili 8
      Case Is > 8 And Broj < 11
            ' Broj je 9 ili 10

      Case Else
            ' Broj ima sve ostale vrednosti
End Select
```

iif funkcija

Jednolinijska funkcija odlučivanja. Vraća rezultat koji je neophodno prebaciti u varijablu ili izraz.

Sintaksa

```
      iif (Uslov, Tačni deo, Netačni deo)
```

Ako je uslov tačan vraća se vrednost Tačni deo, u suprotnom vraća se vrednost Netačni deo.
Primer:

```
a = iif (b>6,10,20)
```

Identično bi mogli napisati pomoću If .. Then strukture:

```
If b>6 Then
      a = 10
Else
      a = 20
end if
```

Petlje

For ... Next petlja

Sintaksa:
```
For varijabla = početna_vrednost To krajnja_vrednost [Step korak]
      telo petlje
Next [varijabla]
```

Primer:
```
Dim i as integer
For i=1 to 100 step 2
      debug.print i
next i
```

Step je opcioni parametar. Ako se ne navede podrazumeva se vrednost 1

Do ... Loop

Sintaksa
```
Do
      telo petlje
      [Exit Do]
Loop
```

- Do ... Loop petlja je 'mrtva petlja', tj. izvršavaće se neprekidno. Uvek (zavisnog od nekog kriterijuma) unutar petlje izvršavamo Exit Do i time prekidamo petlju.

Postoji nekoliko varijanti ove petlje:

Do Until...Loop

```
Do Until Uslov
      telo petlje
Loop
```

Petlja se izvršava sve dok je uslov = False tj. dok ne postane True.
Uslov se ispituje na početku petlje. Ako je tada tačan petlja se neće ni jednom izvršiti

Do While...Loop

```
Do While Uslov
      telo petlje
Loop
```

Petlja se izvršava sve dok je uslov = True tj. dok ne postane False.
Ispitivanje uslova je takođe na početku petlje.

Do ... Loop Until

```
Do
      telo petlje
Loop Until Uslov
```

Isto kao pod (1) samo se uslov ispituje na kraju petlje.

Do ... Loop While

```
Do
      telo petlje
Loop While Uslov
```

Isto kao pod (2) samo se uslov ispituje na kraju petlje.

U sve četiri varijante u telu petlje može postojati neograničen broj Exit Do naredbi koje prekidaju izvršavanje petlje i nastavljaju rad od sledeće linije program ispod nje.

Rukovanje greškama u Visual Basic-u

U Visual Basic-u razlikujemo tri vrste grešaka:

1. Sintaksna greška: nastaje u dizajn režimu kao rezultat pogrešno otkucane naredbe, funkcije, svojstva itd.. Program koji ima ovakvu vrstu greške nije moguće kompajlirati.
2. 'Run - time' greška: nastaje u toku izvršavanja programa. Najčešći izroci su deljenje sa nulom, prkoračenje opsega varijable ili niza, nedostupnost objekta u tom momentu itd... Ove greške treba obraditi i odgovoriti na njih jer mogu onemogućiti ispravni rad programa. U ovom poglavlju obrađujemo takve greške.
3. Logičke greške: program ispravno radi, ali daje pogrešne rezultate. Greška je u samoj logičkoj strukturi programa i spada pod popularno 'ljudske greške'

Run-time greške obrađujemo pomoću On Error naredbe i njenih varijacija. Generalno možemo razlikovati dva metoda.

1. Generalno reagovanje na greške na nivou procedure događaja i/ili korisnički definisane funkcije i subrotine. Koristimo: **On Error Goto *Labela***

Primer:

```
On Error Goto Obrada_Greske ' U slucaju greske idi na label Obrada_Greske
      ...
      ...   ' Telo procedure ili funkcije
      ...
Izlaz:
      Exit sub

Obrada_Greske:
      ...   ' Kod za obradu greske
      ...
      Err.Clear
      Resume Izlaz      ' Vrati se na labelu Izlaz
      ' ili Resume Next
```

Ovaj pristup se najčešće koristi kada ne znamo tačno u kojoj liniji koda može doći do greške, ili ako se greška može dogoditi na više linija. U predhodnom primeru posle prve linije, gde god dođe do greške, izvršenje programa se preusmerava na labelu Obrada_Greske. Ispod te labele obično ide kod kojim se korisnik informiše o grešci i preduzimaju potrebne akcije za njeno ispravljanje. Resume naredba prenosi tok programa na neku drugu labelu. Moguće je koristiti i **Resume Next** i tada se izvršavanje programa nastavlja od sledeće linije programa u odnosu na onu na kojoj je nastala greška. U primeru se koristi **Err** objekt.

Err objekt:

- Svojstvo **Description**: Err.Description daje opis nastale greške. Ako greška nije nastala svojstvo je postavljeno na prazan string
- Svojstvo **Number**: Err.Number daje numerički kod greške. Ako nema greške svosjtvo je postavljeno na nula.
- Metoda **Clear**: Err.Clear resetuje gresku, tj. Number svojstvo postavlja na nula, a Description na prazan string.

2. Drugi metod koristimo kada tačno znamo na kojoj linije će doći do greške. Posle te linije ispituje da li je zaista pokrenuta greška, pa ako jeste radimo njenu obradu.

Primer:

```
On Error Resume Next ' U slucaju greske idi dalje sa programom
...
...
... ' Linija gde ocekujemo pojavu greske
If Err.Number <> 0 Then ' Proveravamo da li je nastala greska
      ...   ' Ako jeste
      ...   ' Obrada greske
      Err.Clear
end if
... 'Nastavak programa
```

Napomena:
Postoje retke prilike gde je racionalno staviti samo On Error Resume Next. Tada u slučaju greške program nastavlja sa sledećom linijom, tj. ignoriše grešku.

Rad sa fajlovima

Visual Basic poseduje standardni Basic skup naredbi za rad sa fajlovima. U ovom kursu ćemo upoznati tri varijante Open funkcije. Generalna sintaksa ove funkcije je:

Open *ImeFajla* **For** *Pristup* **As** #*Identifikator*

ImeFajla: Parametar koji predstavlja puno ime fajla koji otvaramo

Pristup: Režim pod kojim otvaramo fajl
1. Input (vršimo čitanje sa fajla)
2. Output (pišemo na fajl)
3. Binary (čitamo sa binarnog - ne tekstualnog fajla)

Identifikator: Broj pod kojim otvaramo fajl. Kasnije služi kao identifikator. Ne možemo otvoriti dva fajla pod istim brojem.

Primer:

```
Open "C:\Windows\ReadMe.txt" For Input As #1
' Otvaramo dati fajl za čitanje sa brojem 1
```

Funkcije za rad sa fajlovima:

FreeFile
Vraća sledeći slobodan broj pod kojim možemo otvoriti fajl

LOF (Identifikator)
Vraća dužinu fajla u bajtovima koji smo predhodno otvorili pod zadatim identifikatorom.

Input (X, Identifikator)
Učitava X bajtova sa fajla koji ima dati identifikator i vraća ih kao string.

Print #Identifikator, Vrednost
Štampa zadatu vrednost na fajl koji je otvoren sa identifikatorom.
Close [#Identifikator]
Zatvara fajl sa navedenim idetifikatorom. Ako se identifikator ne navede, tj. koristi se samo Close, zatvaraju se svi predhodno otvoreni fajlovi.

Primer:

```
Dim Rezultat As String  ' Promenjiva koja cuva sadrzaj fajal
Dim Ident As Integer    ' Identifikator otvorenog fajla
Dim Duzina As Long      ' Duzina bajtova za citanje

Ident = FreeFile        ' Uzimamo slobodni broj i smestamoga u promenjivu
Ident

' Otvaramo fajl za citanje
Open "C:\Windows\ReadMe.txt" For Input As #Ident

' Uzimamo duzinu fajla i smetamo u promenjivu Duzina
Duzina = LOF(Ident)

' Citamo kompletni sadrzaj fajla i smestamo ga u promenjivu Rezultat
Rezultat = Input (Duzina,Ident)

Close #Ident
```

Forme

Forme predstavljaju elementarni nivo komunikacije sa korisnicima. Nazivaju se kontejenrima, zato što na njih postavljamo potrebne kontrole. Prilikom otvaranja novog projekta, inicijalno se kreira početna forma pod nazivom Form1. Forma poseduje set svojstava pomoću kojih podešavamo pojavu i ponašanje forme u programu. Sledi lista najčešće korišćenih sa objašnjenjima:

Svojstva forme:

Svojstvo	Moguće vrednosti i objašnjenje
Name	ključno svojstvo svakog objekta u VB-u. U jednom projektu svaka forma mora imati svoje jedinstveno ime, preko koga se referišemo na nju
BackColor	Boja pozadine forme
BorderStyle	Definiše vrstu okvira i ponašanje forme. Moguće vrednosti su: 0-None : Forma je bez okvira 1-Fixed Single: okvir jednostruke debljine 2-Sizable: okvir koji omogućava izmenu dimenzije forme (inicijalna vrednost) 3-Fixed Dialog: forma ima fiksni okvir kome se ne mogu menjati dimenzije 4-Fixed ToolWindow: forma ima izgled toolbox-a (manji font se koristi za naslov forme), i nije moguće menjati joj dimenzije 5-Sizable ToolWindow: isto kao 4 samo je moguće menjati dimenzije forme
Caption	Određuje tekst koji će biti ispisan u naslovu forme
ControlBox	Određuje da li forma ima control box (vrednosti su True ili False)
Icon	Definiše ikonu koja će predstavljati formu (vidi se u gornjem levom uglu). Ako je u pitanju glavna forma aplikacije, ta ikona najčešće predstavlja i samu aplikaciju
MaxButton	Da li će biti prikazano dugme za maksimizaciju forme (True ili False)
MinButton	Isto samo se odnosi na dugme za minimizaciju forme
Moveable	Određuje da li korisnik može da pomera formu. Vrednosti True (može) ili False (ne može je pomerati)
Picture	Omogućava da na formu postavimo sliku
ShowInTaskBar	Da li će forma, kada je otvorena, biti prikazana u Task Bar-u Windows-a
WindowState	Određuje dimenzije prozora prilikom otvaranja: 0-Normal: dimenzije prozora su iste kao one koje su postavljene u dizajn režimu 1-Minimized: prozor će biti prikazan minimizovan 2-Maximized: prozor će biti veličine celog ekrana

Metode forme:

Show
Prikazuje formu. frmGlavniMeni.Show će prikazati formu čije je ime (name svojstvo) frmGlavniMeni.

Hide
Uklanja formu sa ekrana, ali je ostavlja u memoriji. Sledeći Show metod je prikazuje znatno brže jer je forma već u memoriji i nema potrebe da je učitava sa diska.

Load (funkcija)
Load ImeForme učitava formu, ali je ne prikazuje na ekranu.

Unload (funkcija)
Unload ImeForme, uklanja formu i sa ekrana i iz memorije.

Događaji forme:

Naziv događaja	Kada se pokreće
Load	prilikom učitavanja forme sa diska, pre njenog prikazivanja
Unload	prilikom uklanjanja forme sa ekrana i iz memorije
Activate	prilikom aktiviranja forme. Pokreće se posle Load događaja. Takođe se može pokrenuti ako zatvorimo formu koja je naknadno otvorena i vratimo se nazad na predhodnu.
Resize	prilikom izmen dimenzije forme

Subrotine i funkcije

U slučaju da postoje isti segmenti koda koji se izvršavaju na više mesta u programu, te segmente odvajamo imenujemo ih i pišemo samo na jednom mestu. Kasnije kada nam je potrebno njihovo izvršenje jednostavo ih pozivamo. Ovo sprečava dupliranje koda, i značajno olakšava kasnije uklanjanje grešaka i održavanje programa. Zavisno od potrebe razlikujemo dve vrste ovakvih segmenata koda:
1. Subrotine (ili procedure): ne vraćaju vrednost
2. Funkcije: vraćaju vrednost

Zavisno od mesta pozivanja (da li ih pozivamo iz iste forme ili imamo potrebu da ih pozivamo iz svake forme u projektu), kreiramo ih ili na nivou forme ili u modulu. Tipično u modulu kreiramo segmente koji su nam potrebni u celom projektu.

Subrotine:

Procedura je segment koda koji direktno ne vraća vrednost, već izvrši svoju ulogu i kontrolu toka programa prenosi na mesto odakle je pozvana.

Sintaksa:

```
[Public][Private] Sub ImeSubrotine ([Par1],[Par2],...,[Par n])
      ...
      ...   ' Telo subrotine
      [Exit Sub]
      ...
End Sub
```

Objašnjenje:

[Public]
Označava da je subrotina javna, tj možemo je pozvati iz bilo kog dela projekta

[Private]
Znači da je vidljiva samo u mestu gde je i napisana (samo iz te forme)

ImeSubrotine:
Identifikator koji označava subrotinu i pomoću koga je kasnije pozivamo.

[Par1],[Par2],...,[Par n]
Ulazni parametri subrotine. Opciono. Ako ih nema ostavljamo samo otvorenu i zatvorenu zagradu. Ako ih ima više razdvajamo ih zarezom.

U telu subrotine možemo opciono imati jednu ili više **Exit Sub** naredbi koji odmah završavaju izvođenje i predaju kontrolu pozivajućem programu.

Ulazni parametri

Ulazne parametre možemo proslediti na dva načina: Po vrednosti (ByVal) i po referenci (ByRef) . Takođe je poželjno (nije neophodno) zadati tip promenjive koja je ulazni parmetar. Sintaksa definicije ulaznog parametra:

```
[ByRef]|[ByVal] NazivPromenjive [As Tip]
```

ByRef Prenos parametra po referenci. Ako u subrotini izmenimo vrednost parametra, takođe će biti izmenjena vrednost i orginalne varijable iz pozivajućeg koda. Kažemo da smo proceduri prosledili pokazivač na varijablu.

ByVal Prenos parametra po vrednosti. U subrotini se kreira lokalna kopija ulaznog parametra i sve izmene nad njom ne utiču na vrednost originalne varijable

Ako ne naglasimo način prosleđivanja ulaznih parametara podrazumeva se po referenci ByRef.

As Tip Tip parametarske varijable. Ako ga definišemo, varijabla koju prenosimo prilikom pozivanja subrotine mora biti istog tipa. Ako ga ne definišemo, podrazumevan tip je Variant.

Pozivanje subrotine možemo vršiti na dva načina:

Call ImeSubrotine (parametri)

ImeSubrotine parametri

Subrotina može imati i opcione parametre. To su parametri koje nije neophodno proslediti prilikom pozivanja. Opcioni parametri se definišu ključnom reči Optional i moraju se staviti na kraju liste ulaznih parametara.

U subrotini, funkcijom IsMissing (ime_opcionog_parametra) možemo proveriti da li je parametar prosleđen ili ne. IsMissing funkcija u tom slučaju vraća vrednosti False ili True respektivno. Ova funkcija radi samo ako su opcioni parametri deklarisani kao tipovi Variant.

Primer subrotine:

```
Private Sub KalkulacijaTabele (ByVal Red As Integer, ByRef Kolona As Integer, Optional Koef as Variant)

        If IsMissing (Koef) Then
                ' Opcioni parametar nedostaje
                ...
        End If
End Sub
```

Funkcije:

Za funkcije važe sve što i za subrotine. Razlika je u tome što one vraćaju vrednost, pa se razlikuje deklaracija i pozivanje.

```
Sintaksa:

[Public][Private] Function ImeFunkcije ([Par1],..,[Par n]) As Tip
      ...
      ...   ' Telo funkcije
      [Exit Function]
      ...
      ImeFunkcije = Vrednost
End Function
```

Pošto funkcija vraća vrednost, u deklaraciji se mora navesti tip podataka te vrednosti. Sama povratna vrednost se dodeljuje u telu funkcije tako što ime funkcije tretiramo kao varijablu. Poslednja postavljena vrednost pre izlaska iz funkcije će biti vrednost koju funkcija vraća.

Prilikom pozivanja funkcije, vrednost koju ona vraća moramo smestiti u varijablu ili svojstvo objekta.

Primer deklaracije funkcije:

```
Private Function Kalkulacija (ByVal Ulaz As Single) As Single
      Kalkulacija = 3.14 * Ulaz
End Function
```

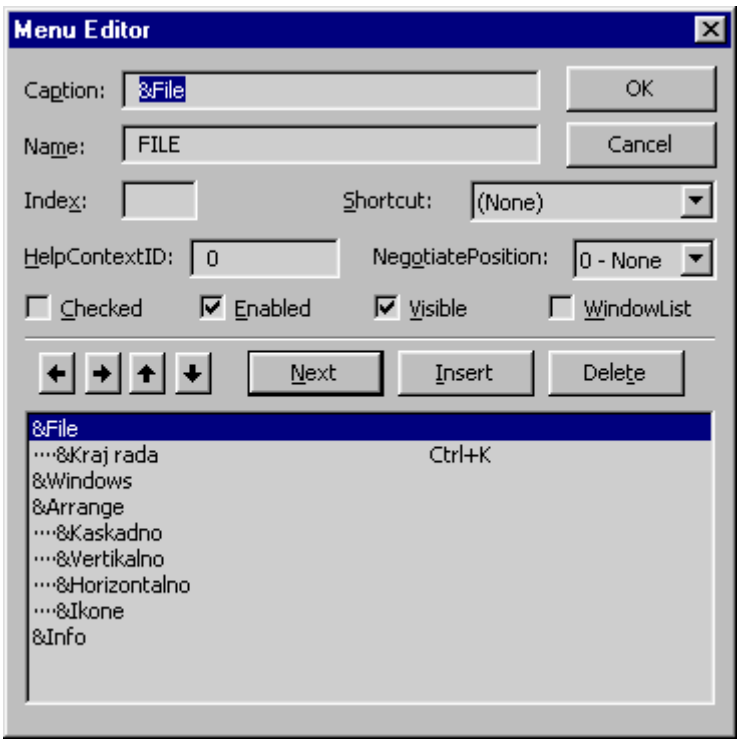
Primer poziva funkcije:

```
Dim a As Single
Dim b As Single
a = 7.4

b = Kalkulacija (a)
MsgBox b
```

Meniji

Kriranje menija u Visual Basic-u je izuzetno jednostavno. Meni je vezan za formu i svaka forma može imati svoj meni po potrebi. Moeni kreiramo pomoću Meni Editor-a, koji se pokreće iz tool box-a. Svaka stavka iz glavne linije menija može (ali i ne mora imati podstavke - pull down meni). U VB-u možemo imati menije u maksimalno četiri nivoa. U praksi dva ili retko tri, zadovoljavaju sve potrebe. Svaku stavku menija definiše njen nivo, svojstvo Caption i svojstvo Name. Ovo su tri neophodna parametra.

Izgled meni editora:



Značenje drugih svojstva stavke menija:

Index:	kreiranje indeksiranog niza meni kontrola
Shortcut:	dodeljivanje skraćenica za stavku menija.
HelpContexId:	identifikacija stavke Help fajla koji je dodeljen aplikaciji, za datu stavku
NegotiatePosition:	u vezi sa prikazom stavke menija kod OLE povezivanja
Checked:	mogućnost da stavka menija bude čekirana ili ne
Enabled i Visible:	kao za ostale kontrole
WindowsList:	Prikazuje listu otvorenih MDI Child formi

Strelice Levo i Desno:	Podešavanje nivoa stavke menija
Strelice gore i dole:	Pomeranje nivoa stavke menija
Next:	Nova stavka ili pomeranje na narednu
Insert:	Insertovanje nove stavke menija iznad selektovane
Delete:	Brisanje selektovane stavke menija

Stavke menija prepoznaju samo klik (Click) događaj, gde i pišemo odgovarajući kod.

PopUp Meniji

Pop up meniji su često korišćeni u Windows aplikacijama. U domaćoj literaturi se nazivaju "otrgnuti meniji" jer zaista i prikazuju delove već kreiranih menija na formi. Najčešće se otvaraju klikom desnog tastera miša na formi.

Sintaksa:

```
PopUpMenu ImeMenija, Flags, x, y, DefaultMeni
```

ImeMenija:	Naziv (Name svojstvo) menija na glavnoj liniji
Flags:	Definiše dve stvari: Pozicija menija u odnosu na kurzor miša: 0 - Levo od kurzora 4 - Centrirano 8 - Desno od kurzora Određuje na koje tastere reaguju stavke menija 0 - Reaguje samo na levi (primarni) taster miša 3 - Reaguje na oba tastera miša
x, y:	kordinate na kojoj se prikazuje meni. Ako se izostave meni se prikazuje na poziciji miša
DefaultMeni	naziv stavke menija koja će biti prikazana podebljano

Standardne Kontrole

U ovom delu će biti govora o standardnim Windows kontrolama koje Vam stoje na raspolaganju. Prikazaćemo svojstva, metode i događaje ovih kontrola uz objašnjenje i tamo gde je primereno, uz kratak kod kao ilustraciju. Ove kontrole se inicijalno nalaze u svakom Visual Basic projektu, nije ih potrebno dodavati pomoću opcije Project -> Components, a takođe ih nije ni moguće ukloniti sa ToolBox-a. Ove kontrole su:

- Label (oznaka)
- TextBox (polje za unos teksta)
- CommandButton (komandno dugme)
- CheckBox (polje za potvrđivanje)
- OptionButton (dugme za izbor opcija)
- Frame (okvir)
- ListBox (lista)
- ComboBox (padajuća lista)
- HScrollBar (horizontalna traka za pomeranje)
- VScrollBar (vertikalna traka za pomeranje)
- Timer (sat, štoperica)
- DriveListBox (lista disk jedinica u sistemu)
- DirListBox (lista direktorijuma aktivne disk jedinice)
- FileListBox (lista fajlova aktivnog direktorijuma)
- Shape (geometrijski oblici)
- Line (linija)
- PictureBox (okvir za sliku i crtanje)
- Image (okvir za sliku)

Uz originalni (engleski), naziv svake od kontrola je dat prevod koji je odomaćen u našoj literaturi, radi Vaše reference. Mi ćemo, ipak zbog lakšeg snalaženja u Visual Basic okruženju koristiti engleska imena ovih kontrola. Neka svojstva se ponavljaju u više kontrola. Imaju isti naziv i funkciju, tako da ćemo ih objasniti samo kod prvog pojavljivanja, a kasnije samo napomenuti da postoje. Na primer **Name** svojstvo je zajedničko za sve kontrole. Ono predstavlja jedinstveni identifikator kontrole na nivou forme.

Ovo znači da na istoj formi ne možemo imati dve kontrole bilo kog tipa sa istim **Name** svojstvom. Izuzetak je indeksirani niz kontrola, kada slično indeksiranom nizu promenjivih možemo imati više kontrola sa istim imenom, ali svaka od njih ima jedinstveni indeks identifikator. Svojstvo koje ovo određuje se, logično, zove **Index**. Svaka kontrola koja u sebi sadrži bilo kakav tekst ima i **Font** svojstvo koje postavlja atribute tipa slova koji se koriste.

LABEL

Namena:

Label kontrola u sebi sadrži najčešće fiksni tekst kao pokazatelj šta je namena neke druge kontrole koja sledi. U tom smislu najčešće se koristi u kombinaciji sa kontrolama **TextBox**, **ListBox**, **ComboBox** i ostalima. Ova kontrola ne može dobiti fokus, tj prilikom izvršavanja programa ne možete kurzor postaviti na nju i menjati joj sadržaj. Što se tiče korisnika ona je samo za čitanje (*Read-Only*). Naravno, sadržaj teksta i ostala svojstva ove kontrole možemo programski menjati u vreme izvršavanja programa, zavisno od potrebe. Treba znati da u sadržaju teksta ove kontrole možemo naznačiti *Hot-Key* kombinaciju za ovu kontrolu. Pošto ona ne može dobiti fokus, fokus se postavlja na prvu narednu kontrolu (zavisno od tab redosleda kontrola na formi), koja može dobiti fokus. Na primer, ako je prva kontrola labela sa tekstom Kupac a naredna TextBox, pritiskom na kombinaciju tastera Alt+K, postavićemo fokus na TextBox kontrolu. Ovo se često koristi pri kreiranju formi, gde posebno treba obratiti pažnju na tab redosled kontrola.

Label kontrola može biti vezana, (*bound*) kontrola, u smislu rada sa bazama podataka. Ovo znači da može prikazati vrednost nekog polja iz baze podataka, naravno bez mogućnosti izmena, pa je u tom smislu treba i koristiti.

Svojstva:

Svojstva Label kontrole koja se najčešće koriste su:

(Name):

Jedinstveno ime kontrole na nivou forme pomoću kojeg se referišemo na svojstva i metode ove kontrole. Inicijalno ime prve Label kontrole na formi je **Label1**, sledeće **Label2** i tako redom.

Appearance:

Svojstvo koje određuje grafički izgled kontrole. Moguće vrednosti su

0-Flat	kontrola ima standardni okvir
1-3D	kontrola ima trodimenzionalni okvir

Ovo svojstvo ima funkciju samo ako je svojstvo **BorderStyle** postavljeno na **1-FixedSingle**

BorderStyle:

Određuje da li se iscrtava okvir oko kontrole. Moguće vrednosti su:

0-None	kontrola je bez okvira
1-Fixed single	kontrola ima tip okvira zavisno od vrednosti svojstva Appearance

Autosize:

Ako je vrednost ovog svojstva postavljeno na *True*, veličina kontrole će biti prilagođena tekstu koji kontrola sadrži. Inicijalno ova vrednost je postavljena na *False*.

BackStyle:

Određuje način iscrtavanja kontrole na formi.

1-Opaque	kontrola prekriva sadržaj ispod sebe
0-Transparent	kontrola je transparentna

Ako na formu postavljate bilo kakvu bitmapiranu ili vektorsku sliku i želite da se ona "providi" ispod label kontrole ovo svojstvo treba postaviti na 0-Transparent

Caption:
Tekstualni sadržaj kontrole. Ako želite da definišete *Hot-Key* kombinaciju, ispred željenog slova stavite znak "&". Na primer, ako vrednost ovog svojstva postavite na "Ime &kupca", na labeli će biti ispisan tekst "Ime kupca". Podvučeno slovo k označava Hot-Key kombinaciju (Alt+K) koja će postaviti fokus, ali ne na labelu, već na prvu narednu kontrolu koja može dobiti fokus. Naravno, nema smisla postaviti više od jednog karaktera za Hot-Key. U svakom slučaju ako to uradite, Visual Basic će uzeti u obzir samo poslednji označen karakter. Inicijalno, **Caption** svojstvo kontrole je jednako imenu kontrole (name svojstvo).

UseMnemonic:
Svojstvo od koga zavisi da li se karakter "&" tretira kao oznaka za Hot-Key kombinaciju, ili će se literalno ispisivati na kontroli. Ako želite da se karakter "&" prikaze na kontroli ovo svojstvo treba postaviti na False. Ako ovo uradite, tekst iz predhodnog slučaja će biti prikazan na labeli kao "Ime &kupca". Inicijalna vrednost ovog svojstva je True. U ranijim verzijama Visual Basica nije bilo moguće na regularan način ispisati karakter "&" na Label kontroli.

Alignment:
Poravnanje teksta na kontroli. Moguće opcije su:

0-Left Justify	poravnanje ulevo
1-Right Justify	poravnanje udesno
2-Center	tekst u kontroli je centriran u odnosu na okvir kontrole

WordWrap:
Svojstvo koje određuje da li će tekst unet u kontrolu biti raspoređen horizontalno ili vertikalno. Ima uticaja samo ako je **Autosize** svojstvo postavljeno na true.

DataSource, DataField:
Label može biti vezana (bound) kontrola. Svojstva koja određuju vezu ove kontrole sa bazom podataka i koja su posebno obrađena u naprednom kursu "Visual Basic i baze podataka".

Događaji:

Inicijalni događaj ove kontrole je **Click događaj**. Izvršava se kada korisnik klikne mišem na kontrolu.

Sumarno:

Label kontrolu ćete načešće koristiti kao statični tekst iza koga sledi TextBox kontrola. Takođe je možete povezati za polje iz baze podataka za Read-Only pregled. Metodi i događaji ove kontrole se ređe koriste.

U toku izvršavanja programa tekst ispisan na ovoj kontroli menjamo konstrukcijom:

```
Label1.Caption = "Novi sadrzaj"
```

TEXTBOX

Text box kontrola omogućava unos i izmenu teksta u vreme izvršavanja programa. Kontrola može dobiti fokus i najčešće se postavlja posle Label kontrole. TextBox je vezana kontrola i daje mogućnost pregleda ili izmena vrednosti polja iz baze podataka. Inicijalno, naziv prve kontrole je postavljen na Text1, sledeće Text2 itd.

Svojstva:

Alignment, Appearance, BorderStyle imaju istu funkciju kao kod kontrole Label. Jedina razlika je da svojstvo Alignment ima uticaja samo ako je svojstvo MultiLine postavljeno na True.

MultiLine:
Svojstvo koje određuje da li se unet tekst prostire u više linija (True), ili samo u jednoj (False).

ScrollBars:

Određuje da li će na kontroli biti postavljene trake za pomeranje tekstualnog sadržaja ove kontrole. U funkciji je samo ako je svojstvo MultiLine postavljeno na True. Moguće vrednosti su:

0-None	bez traka za pomeranje
1-Horizontal	samo horizontalna
2-Vertical	samo vertikalna
3-Both	i horizontalna i vertikalna

Ako ovo svojstvo postavite na 1-Horizontal ili 3-Both, tekst koji unosite će se pomerati udesno sve dok ne pritisnete Enter taster, čime ste otvorili novi red za unos teksta. U suprotnom ako je svojstvo postavljeno na 0-None ili 2-Vertical, tekst koji unosite će biti automatski prelomljen u naredni red čim dođete do desne ivice polja za unos.
Najčešće, se ovo svojstvo postavlja na 2-Vertical i u tom slučaju širina same kontrole određuje maksimalnu širinu linije teksta koja se unosi.

HideSelection:

Skrivanje selekcije. Standardni način selekcije teksta tj. dela teksta u Windows-ima je prevlačenje miša preko željenog teksta uz pritisnuti levi taster ili pomoću kurzorskih tastera uz istovremeno pritisnut Shift taster. Ovako označeni tekst je markiran drugom bojom. Ovo svojstvo određuje da li će se označeni tekst ostati markiran pošto kontrola izgubi fokus (True) ili će ostati markiran (False). Inicijalna vrednost ovog svojstva je True.

MaxLength:

Svojstvo određuje maksimalni mogući broj unetih karaktera. Ako je postavljeno na 0 onda nema ograničenja. U suprotnom, maksimalni broj karaktera koji možete upisati odgovara vrednosti ovog svojstva. Inače, ograničenje TextBox kontrole je da maksimalno može da sadrži 65535 karaktera, pa samim tim ni vrednost ovog svojstvo ne može biti veće. Inicijalno vrednost je postavljena na 0, tj nema ograničenja u broju unetih karaktera.
Ovo svojstvo dolazi do izražaja pri radu sa bazama podataka, kada je neophodno da ograničimo broj unetih karaktera na definisanu dužinu tekstualnog polja u bazi. U suprotnom može doći do prekoračenja dužine unetog teksta i do generisanja greške prilikom ažuriranja polja.

SelStart, SelLength, SelText:

SelStart je svojstvo koje nam omogućava da pročitamo ili postavimo poziciju početka selekcije teksta. Imajte u vidu da vrednost 1 znači početak selekcije od prvog karaktera ne uključujući prvi karakter. Ako želimo da postavimo početak selekcije na prvi karakter SelStart treba postaviti na 0.
SelLength čita ili postavlja dužinu selektovanog teksta u karakterima.
SelText vraća selektovan sadržaj. Prilikom postavljanja ovog svojstva moguća su dva slučaja:

Ako je predhodno selektovan deo ili ceo tekst, ono što je selektovano biće prebrisano i zamenjeno novim sadržajem.

Ukoliko nije ništa selektovano, nov sadržaj će biti dodat na početak teksta. Sledi kratak primer:

Ako je sadržaj TextBox-a "**Primer**" naredni kod radi sledeće

```
Text1.SelStart = 2              ' početak selekcije od karaktera i (uključivo)
Text1.SelLength = 3             ' selekcija naredna tri karaktera
msgbox Text1.SelText            ' selektovan tekst je "ime"

' ako sada postavimo vrednost selektovanog teksta:
Text1.SelText = "12345"
' sadržaj kontrole će biti " Pr12345r"
' sada uklanjamo selekciju
Text1.SelStart = 0
Text1.SelLength = 0
Text1.SelText = "xyz"
' posle ovog sadržaj teksta u kontroli će biti "xyzPr12345r"
```

Ako želite da selektujete celokupni tekst u kontroli Text1, to možete uraditi na dva načina:

```
Text1.SelStart = 0
Text1.SelLength = 65535          ' maksimalna moguća dužina teksta
```

ili

```
Text1.SelStart = 0
Text.SelLength = Len(Text1.Text)
```

Bitno je znati da svojstva SelStart i SelLength mogu imati vrednosti u intervalu od 0-65535. Ako ste postavili SelStart na npr 100, a SelLength na maksimalnu vrednost 65535 proizilazi da je ukupni broj karaktera u polju 65535+100=65635 što prevazilazi gore navedeni maksimalni broj karaktera koji može da prihvati ova kontrola. Međutim. a ako Visual Basic u ovoj situaciji ne generiše nikakvu grešku, ograničenja ostaju, samo znači da treba pažljivo koristiti ova svojstva.
Ako isprobavate ovaj primer, obavezno svojstvo HideSelection postavite na False, kako bi na kontroli videli označeni tekst.

PasswordChar:

U slučaju da želite da sakrijete tekst koji unosite u kontrolu, tj da umesto njega ispisujete jedan isti karakter, u ovo svojstvo unesite taj karakter. Dozvoljeno je uneti samo jedan karakter. Najčešće se koristi prilikom unosa šifri, kada ne želimo da neko slučajno ili namerno vidi šta kucamo na ekranu. Standardni karakter za ovu namenu je "*". Posle ovoga šta god ukucavali u ovu kontrolu umesto unetih karaktera će biti prikazana *. Naravno, ovo se samo prikazuje, stvarno unet tekst možete dalje normalno obrađivati. Ako ovo svojstvo ostavite prazno, TextBox se ponaša standardno.

Locked:

Svojstvo koje zaključava tekst za bilo kakve izmene. Za razliku od varijante Enabled = False, omogućava da kontrola dobije fokus, ako postoje trake za pomeranje, one normalno funkcionišu. Ovo je jedini način da pregledate višelinijski tekst koji ne vidite ceo na kontroli, a da onemogućite njegovu izmenu. Korisno pri radu sa bazama podataka, kada za vreme izvršavanja programa možete kontrolisati vrednost Locked i/ili Visible svojstva, zavisno od na primer, nivoa pristupa korisnika.

Metode:

SetFocus:

Metoda kojom postavljamo fokus na datu TextBox kontrolu. Na primer ako želite programski da postavite fokus na TextBox kontrolu pod nazivom Text1, uradite ovo:

```
Text1.SetFocus
```

Događaji:

Click:

Aktivira se kada korisnik klikne mišem na kontrolu.

Change:

Ovaj događaj se aktivira prilikom bilo kakve izmene teksta u TextBox-u. U praksi se koristi za validaciju unetog teksta za vreme unosa. Na primer, možete proveriti koji tekst je unet i zavisno od toga izvršiti neku akciju.

LostFocus:
Aktivira se kada kontrola izgubi fokus. Koristi se najčešče za post validaciju unetog teksta. Naredni primer proverava da li je uneti tekst u kontrolu pod nazivom Text1 validni datum:

```
Private Sub Text1_LostFocus()
      ' da li je unet tekst pogresan datum
      If Not IsDate(Text1.Text) Then
            ' ako jeste pogresan datum izbaci poruku
            MsgBox "Pogresan datum", vbCritical, "Test datuma"
            ' vrati fokus na Text1
            Text1.SetFocus
      End If
End Sub
```

KeyDown, KeyPress:
Aktivira se prilikom pritiska na bilo koji taster, dok kontrola ima fokus. Razlika između ova dva događaja je u ulaznim parametrima.
KeyDown za ulazne parametre ima KeyCode (kod pritisnutog tastera) i Shift (stanje shift tastera).
KeyPress za ulazne parametre ima KeyAscii (Ascii kod pritisnutog tastera). Svi ulazni parametri su tipa Integer.

U slučaju da želite da ispitate kod alfanumeričkih tastera, koristićete KeyPress događaj, jer on ne reaguje na specijalne tastere (funkcijski tasteri, NumLock, CapsLock, NumLock, Insert, Delete, PageUp, PageDown). Ako želite da ispitate stanje svih tastera sa tasture koristićete KeyCode događaj. On vraća kod svih pritisnutih tastera i stanje shift tastera: Ako je pritisnut parametar Shift će imati vrednost 1, u suprotnom vrednost je 0. Sledi kratki primer koji ograničava unos u TextBox pod nazivom Text1 na alfa skup znakova (samo slovni karakteri):

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
      ' pod uslovom da ASCII kod karaktera nije u granicama 65-122
      If KeyAscii < 65 Or KeyAscii > 122 Then
            ' postavi ASCII kod tastera na nulu
            ' Na ovaj način ignorišemo pritisnut taster i on se ne ispisuje
            KeyAscii = 0
      End If
End Sub
```

Ovaj primer nam omogućava da presretnemo sistemsku Windows rutinu za obradu tastature, proverimo i po potrebi promenimo ASCII kod pritisnutog tastera. Ako postavimo kod na 0, praktično poništavamo pritisak na taster (uklanjamo njegov kod iz bafera za tastaturu).

Naredni primer će bez obzira koji taster je zaista pritisnut, njegov ASCII kod promeniti na 65 – slovo "A"

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
      KeyAscii = 65
End Sub
```

Pomoću ovog događaja možete kompletno predefinisati ponašanje tastature za određenu TextBox kontrolu.

COMMAND BUTTON (Komandno dugme)

Kontrola koja korisnuku nudi vuzuelni interfejs za pokretanje određenih akcija. Na sebi može imati ili tekst ili sliku. Koristeći Click događaj, za ovu kontrolu vezujemo kod.

Svojstva:

Style:

Postoje dve varijante:

0-standard; na dugmetu je ispisan tekst
1-graphical; na dugmetu se nalazi slika i tekst ako je unet u svojstvo Caption

Caption:

Svojstvo koje određuje tekst ispisan na dugmetu. Za definicuju Hot-Key kombinacije koja aktivira ovo dugme važe ista pravila kao kod Label kontrole. Nedostaje svojstvo UseMnemonic, tako da je nemoguće na regularan način, na dugmetu ispisati karakter "&".

Picture:

Određuje koja slika će biti iscrtana na dugmetu. U funkciji je samo ako je svojstvo Style postavljeno na 1-graphical. Podržani tipovi slika su:

Bitmapirane slike	(*.bmp)
Ikone	(*.ico)
GIF slike	(*.gif)
JPEG slike	(*.jpg)
Metafajlovi	(*.wmf)

Standardno se koriste *.bmp i *.ico formati slika. Ne stavljajte suviše velike slike bez potrebe jer se one ugrađuju u izvršni fajl, povećavaju ga i usporavaju učitavanje forme na kojoj se nalaze. Ako je na primer, dužina izvršnog fajla 30 Kb, a potom na dugme stavite sliku veličine 300 Kb, posle kompajliranja izvršni fajl će biti 330 Kb. Ako koristite slike sa punom paletom boja, proverite da li ciljni računar podržava ovakav prikaz, inače će se slika prikazati sa redukovanom paletom i potpuno degradirati Vaš pažljivo dizajniran korisnički interfejs. Osim ovog svojstva postoje još dva sa sličnom namenom:

DisabledPicture; **koja slika će biti prikazana ako je svojstvo Enabled** postavljeno na False
DownPicture; **koja slika će biti prikazana kada je dugme pritisnuto**

Default, Cancel:

Dijalozi u Windows-ima standardno imaju dva dugmeta: OK za potvrdu i Cancel za poništavanje izmena i/ili odustajanje od operacije. Nepisano pravilo je da sa tastature aktivirate ove dugmiće pritiskom na Enter i Esc, respektivno. Ako želite da komandno dugme reaguje na Esc taster, postavite Cancel svojstvo na True. Ako želite da reaguje na Enter taster postavite svojstvo Default na True. Na istoj formi samo jedno dugme može imati Cancel svojstvo postavljeno na True i Default svojstvo postavljeno na True.

Metode:

SetFocus:

Ovom metodom postavljamo focus na komandno dugme.

Događaji:

Click:

Gotovo u svakoj aplikaciji jedini korišćeni događaj. U proceduri ovog događaja pišemo kod koji se izvršava kada korisnik klikne na dugme ili ga aktivira Hot-Key kombinacijom.

CHECKBOX

Ova kontrola služi kao prekidač za neku opciju u programu. Za razliku od prekidača koji ima dva stanja, ova kontrola ih ima tri, zbog čega je obično zovu 3-State control (kontrola sa tri stanja). Omogućava vezivanje za polje iz baze podataka, tipično za polje tipa Da/Ne. Ako imamo više CheckBox kontrola na formi, one su međusobno nezavisne, tj. ne isključuju se međusobno.

Svojstva:

Style:

Slično komandnom dugmetu određuje način prikaza kontrole:

0-Standard; ispisan tekst i okvir za čekiranje
1-Graphical; slika i tekst ako je unet u svojstvo Caption.

Caption:

Tekst koji će biti ispisan na kontroli. Važe ista pravila u vezi Hot-Key kombinacije, kao za Label kontrolu.

Picture:

Isto kao kod komandnog dugmeta, uz dodatna svojstva DisabledPicture i DownPicture sa istim funkcijama i napomenama.

Alignment:

Određuje poziciju okvira za čekiranje u odnosu na tekst koji je unesen u Caption svojstvo. Moguće vrednosti su:

0-Left Justify; sa leve strane teksta
1-Right Justify; sa desne strane teksta

Ovo svojstvo je u funkciji samo ako je vrednost svojstva Style postavljeno na 0-Standard. U suprotnom tekst je ispisan centrirano na kontroli.

Value:

Svojstvo koje vraća ili postavlja stanje kontrole. Moguća su tri stanja:

0-Unchecked kontrola nije čekirana
1-Checked kontrola je čekirana
2-Grayed kontrola je u nedefinisanom stanju i nije ažurirana

Koristeći ovo svojstvo možete u dizajn režimu postaviti inicijalni status kontrole, takože za vreme izvrašavanja programa, možete po potrebi menjati njen status.

Metode:

SetFocus:

Metoda postavlja fokus na kontrolu.

Događaji:

Click:

Izvršava se kada korisnik klikne na kontrolu i u isto vreme je izmeni status. Tipično se u tom smislu i koristi. Zavisno od stanja kontrole izvršavate neku akciju, postavljate vrednost promenjive i slično. Sledi primer:

```
Private Sub Check1_Click()
      ' Zavisno od vrednosti svojstva Value
      Select Case Check1.Value
            ' u slucaju da je cekirana
            Case vbChecked
                  MsgBox "Cekirano"

            ' u slucaju da nije cekirana
            Case vbUnchecked
                  MsgBox "Nije cekirano"

            ' u slucaju nedefinisanog stanja
            Case vbGrayed
                  MsgBox "Nedefinisano"
      End Select
End Sub
```

Obratite pažnju da kontrola u ovom primeru nikada ne dostiže nedefinisano (Grayed) stanje. Jednostavno, čim ste kliknuli na nju, vi postavljate stanje na ili Checked ili na Unchecked. Ako je inicijalno, u dizajn ili izvršnom režimu, stanje kontrole postavljeno na Grayed, sledećim klikom ga prevodite na Unchecked.

OPTION BUTTON

Kontrola koja služi za izbor opcija koje se međusobno isključuju. Ako na formi imate na primer tri ovakve kontrole, uvek je moguće samo jednu čekirati, ostale se automatski dečekiraju. Grupu OptionButton kontrola koje se međusobno zavisne nazivamo OptionGroup. Ako imate samo jednu grupu možete sve njene članove postaviti direktno na formu. Postavlja se pitanje kako napraviti više grupa ovih kontrola. U tu svrhu koristimo kontrolu **Frame** (okvir) koja je opisana u narednom tekstu. Kontrola OptionButton ima ista svojstva sa istom funkcijom kao CheckBox kontrola. Jedina razlika je u svojstvu Value.

Svojstva:

Value:

OptionButton je kontrola koja može imati samo dva stanja: True (tačno) i False (netačno). Samim tim svojstvo Value može imati vrednosti True i False. Obratite pažnju na ovo i razlikujte stanja ovog svojstva od istoimenog za CheckBox.

Događaji:

Click:

Kao kod CheckBox kontrole, ovaj događaj koristimo za proveru stanja u jednoj grupi OptionButton kontrola.

FRAME:

Kontrola koja služi za grupisanje drugih kontrola. Ona može da sadrži ostale kontrole i zbog toga je nazivamo kontejner kontrolom. Tipično služi za grupisanje OptionButton kontrola. Sve kontrole koje su postavljene na Frame, se pomeraju istovremeno sa pomeranjem Frame kontrole i zadržavaju svoje relativne pozicije u odnosu na Frame kontrolu. Koordinate kontrola (Top i Left), postavljenih u okvir se ne računaju u odnosu na formu, već u odnosu na okvir. Vodite računa o ovome u slučajevima kada u vreme izvršavanja programa programski menjate pozicije kontrola na formi. Takođe, vodite računa da brisanjem Frame kontrole u dizajn režimu brišete i sve kontrole koje su na nju postavljene.

Svojstva:

BorderStyle:

Određuje da li kontrola ima iscrtan okvir ili ne. Moguće se dve varijante:
0-None; bez okvira
1-Fixed Single; sa okvirom

Appearance:

Ako je svojstvo BorderStyle postavljeno na 1-Fixed Single, određuje da li će okvir biti iscrtan trodimanzionalno ili ne.

Caption:

Svojstvo koje određuje tekst ispisan na vrhu okvira. Za ovo svojstvo u vezi Hot-Key kombinacija važi isto što i za CommandButton kontrolu. Ako ne želite naslov, jednostavno ostavite ovo svojstvo prazno.

LISTBOX

Ova kontrola omogućava prikaz više tekstualnih stavki u vidu liste. Ovu listu "punimo" u vreme izvršavanja programa. Moguće je naknadno dodavati ili brisati stavke, proveriti koliko ih ima u listi, proveriti da li je neka i koja selektovana. Listu možemo prikazati u jednoj ili više kolona, zavisno od potreba. U ranijim verzijama Visual Basic-a postojalo je ograničenje u vezi ukupne dužine svih stavki koje se nalaze u ListBox kontroli. Ovo ograničenje više ne postoji, jedino ste ograničeni resursima računara. Međutim, nemojte ovo zloupotrebljavati, jer samo "punjenje" liste može potrajati. Ako želite da u listi vidite polja iz baze podataka, isključivo koristite sličnu kontrolu pod nazivom DBList koje je detaljno objašnjena na kursu "Visual Basic i baze podataka".

Svojstva:

Columns:

Određuje broj kolona u listi. Inicijalno ima vrednost 0, što u ovom slučaju znači jedna kolona, uz vertikalnu traku za pomeranje. Sve stavke u listi su prikazane u jednoj koloni, a ako postoje kolone van opsega kontrole, automatski se postavlja vertikalna traka za pomeranje. Ako ovo svojstvo postavite na vrednost 1, onda će takođe postojati jedna kolona, ali će po potrebi biti postavljen horizontalni scroll bar, pomeranje liste je horizontalno. Za sve ostale vrednosti svojstva Columns, biće postavljen zadati broj kolona i po potrebi horizontalna traka za pomeranje.

ItemData:

Svaku stavku u listi karakteriše njen sadržaj i opciono numericki identifikator. Metodom ItemData(x)=vrednost za stavku na poziciji x, vezujemo numerički podatak vrednost. ItemData predstavlja niz numerika tipa long integer, čija je dimenzija jednaka broju stavki u listi umenjinih za jedan. Najčešće ga koristimo kao indeks niza podataka ili struktura smeštenih u listu. U ovom smislu, što se podataka tiče, listu možete tretirati kao dvodimenzionalni niz strukture podataka čiji su članovi sam sadržaj stavke (tip podataka string) i ItemData (tip podataka je long integer).

NAPOMENA:

Pozicije stavke u listi počinju od 0. Dakle prva stavka ima poziciju 0, druga poziciju 1, a poslednja poziciju N-1, gde je N ukupan broj stavki u listi.

MultiSelect:

Svojstvo koje određuje kako se ponaša lista i da li omogućava izbor više stavki od strane korisnika. Postoje sledeće mogućnosti:

0-None nije moguće izabrati više stavki, već samo jednu (inicijalna vrednost)

1-Simple moguć je izbor više stavki, klikom levog tastera miša. Ako kliknete na već izabranu stavku, izvršićete deselekciju te stavke

2-Extended moguć je izbot više stavki i to na dva načina.

Ako držite pritisnut Ctrl taster i kliknete mišem, selektovali ste stavku ili je deselektovali ako je već ranije selektovana.

Ako kliknete na stavku, potom pritisnete Shift taster i kliknete na neku drugu, sve stavke počev od prve do druge će biti selektovane (uključujući prvu i drugu).

U vezi selekcije je naredno opisano svojstvo Selected.

Selected:

Vraća vrednost True ili False zavisno od toga da li je stavka u listi selektovana ili ne. Ovo svojstvo se može iskoristiti za pregled izabranih stavki u listi ili za programsko selektovanje stavki. Izraz List1.Selected(5)=True će selektovati šestu stavku u listi. Izraz If List1.Selected(3) Then ... će biti tačan ako je četvrta stavka u listi selektovana.

SelCount:

Svojstvo koje vraća broj izabranih stavki u listi.

Sorted:

Ako je vrednost ovog svojstva u listi postavljeno na True, stavke u listi će biti sortirane po abecednom redosledu, u suprotnom za False, stavke će biti prikazane onim redosledom kojim su i unete.

Ovo svojstvo se može postaviti samo u dizajn režimu. U izvršnom režimu je samo za čitanje. Treba obratiti pažnju da se sortiranje vrši u rastućem nizu i to tako što se brojevi tretiraju kao alfa (slovni) karakteri. Zbog ovoga, ako u listi imate brojeve, oni neće biti pravilno sortirani jer se tretiraju kao slovni znakovi. Na primer, ako u listi imate brojeve od 1 do 100, i ako je svojstvo Sorted postavljeno na True, dobićete ovakav redosled: 1,10,100,11,12,...,19,2,20 itd.

Style:

Svojstvo koje odeđuje način prikaza stavki u listi. Postoje dve mogućnosti:

0-Standard standardna lista

1-CheckBox lista u kojoj se levo od svake stavke nalazi checkbox. U ovom slučaju svojstvo MultiSelect mora biti postavljeno na 0-None. Korisnik selektuje stavku ili više stavki, tako što čekira checkbox sa leve strane stavke.

Ovo svojstvo je moguće postaviti samo u dizajn režimu. U izvršnom režimu je samo za čitanje.

ListCount:

Vraća broj stavki u listi. Izraz msgbox List1.ListCount će u dijalogu za poruke prikazati broj stavki u listi pod nazivom List1. ListCount svojstvo je samo za čitanje.

ListIndex:

Vraća ili postavlja indeks aktivne stavke u listi. Izraz List1.ListIndex = 6 će postaviti fokus na sedmu stavku liste. Tipično se koristi za Click događaj liste. Ovaj događaj se odigrava kada korisnik mišem klikne na neku od stavki sa liste. Tada nam najčešće treba ili vrednost ItemData svojstva ili sadržaj izabrane stavke, kako bi zavisno od toga nešto dalje uradili u programu. Ako za ovaj događaj napišete:

```
msgbox List1.ListIndex
```

svaki put kada kliknete na neku stavku liste, u dijalogu za poruke će biti prikazan indeks izabrane stavke. Obratite pažnju da prva stavka ima indeks 0.

NAPOMENA: Ako u vreme izvršavanja programa želite da pomoću ListIndex postavite fokus na stavku iz liste, pazite da ne dođe do prekoračenja, tj. da indeks stavke bude u opsegu od 0 do broj stavki – 1. Takođe predhodno proverite pomoću ListCount da li uopšte ima stavki u listi. Ako je lista prazna, onda će ListCount svojstvo imati vrednost 0.

List:

Svojstvo koje vraća tekstualni sadržaj stavke sa zadatim indeksom. Izraz:
msgbox List1.List (8) će vratiti sadržaj devete stavke po redu. Što se prekoračenja tiče, važi isto što i za ListIndex svojstvo.

ListBox kontrola se takođe, može vezati za bazu podataka. Ali ne u smislu da izlistava vrednosti nekog polja, već samo kao mogućnost da se izabrana stavka iz liste veže za jedno polje aktivnog sloga.

Metode:

Clear:

Metoda primenjena nad ListBox kontrolom briše sve stavke liste. Izraz:
List1.Clear briše sadržaj liste pod nazivom List1

AddItem:

Metoda koja služi za dodavanje novih stavki u listu. Sintaksa je:
ListObject.Additem Item, Index

Item je podatak tipa string koji predstavlja tekst stavke koja se dodaje.

Index je opcioni parametar tipa long integer. Ako je izostavljen, stavka se dodaje na dno liste, a ako je postavljen na neku vrednost, nova stavka se umeće na to mesto, a ostale stavke ispod nje su potisnute za jedno mesto naniže. Ovde je takođe neophodno da vodite računa o prekoračenju. Validne vrednosti za Index parametar su od 0 do ListCount, pod uslovom da imate stavke u listi u suprotnom validna vrednost je samo 0.

RemoveItem:

Metoda koja brise određenu stavku u listi. Sintaksa je:
ListObject.RemoveItem Index

Index je parametar tipa Long Integer. Validne vrednosti su od 0 do ListCount-1, pod uslovom da u listi uopšte ima stavki. Posle ove upotrebe metode, vodite računa da se menjaju indeksi stavki u listi koje se nalaze ispod obrisane stavke.

Događaji:

Click:

Događaj koji se najčešće koristi da bi se odredio indeks i sadržaj stavke na koju je korisnik kliknuo mišem.

Primer za ListBox kontrolu:

Otvorite novi projekt, i na inicijalnu formu smestite ListBox kontrolu. Takođe kreirajte dva komandna dugmeta. Inicijalno, ime (name svojstvo), ListBox kontrole je postavljeno na List1, a komandnih dugmadi na Command1 i Command2. Želimo da pritiskom na komandno dugme Command1 napunimo listu sa 100 stavki. Sadrzaj svake stavke će biti broj od 1 do 100. Pritiskom na komandno dugme Command2, želimo da u dijalogu za poruke dobijemo sve stavke iz liste koje smo selektovali. Konačno ako uradimo dupli klik miša na neku stavku u listi, želimo da je uklonimo iz liste. Ovaj kratki primer će ilustrovati metode **Clear**, **AddItem**, **RemoveItem**, svojstva **ListCount**, **ListIndex**, **List** i događaj **Click** za komandnu dugmad i ListBox.
NAPOMENA:

Konstrukcije koje se koriste u primeru mogu biti nejasne ako zadržimo inicijalnu vrednost svojstva Name za ListBox. Konstrukcija List1.List (List1.ListIndex) svakako nije baš razumljiva na prvi pogled. Međutim, ako svojstvo name ListBox kontrole promenimo na (na primer), Brojanje ova konstrukcija postaje Brojanje.List (Brojanje.ListIndex) i svakako je razumljivija. Radi ovoga promenite vrednost Name svojstva za ListBox na Brojanje. Takođe svojstvo MultiSelect ListBox kontrole postavite na 2-Extended kako bi korisniku omogućili višestruku selekciju stavki u listi.

Procedura događaja Click za komandno dugme Command1:

```
Private Sub Command1_Click()
      ' deklaracija pomocne brojacke promenjive
      Dim i As Integer

      For i = 1 To 100
            ' dodaj stavku u listu
            ' naziv stavke je tipa string, tako da vrsimo konverziju.
            ' ovaj korak nije neophodan jer ce Visual Basic svejedno uraditi
            ' interno konverziju tipova promenjivih, ali je svakako
            ' preporucljiva
            ' praksa konverzije u svim slucajevima uraditi eksplicitno
            Brojanje.AddItem "Stavka No" & Str$(i)
      Next
End Sub
```

Procedura događaja Click za komandno dugme Command2:

```
Private Sub Command2_Click()
      ' deklaracija pomocne brojacke promenjive
      Dim i As Integer

      ' brojac od 0 do broj stavki u listi -1
      For i = 0 To Brojanje.ListCount - 1
            ' ako je stavka sa indeksom i selektovana
            If Brojanje.Selected(i) Then
                  ' poruka, sadrzaj stavke
                  MsgBox Brojanje.List(i), _
                        vbInformation, "SELEKTOVANO"
                  ' vrsi deselekciju stavke
                  Brojanje.Selected(i) = False
            End If
      Next i
End Sub
```

Procedura događaja Click za ListBox Brojanje:

```
Private Sub Brojanje_DblClick()
      ' brise stavku na koju je korisnik kliknuo misem
      Brojanje.RemoveItem (Brojanje.ListIndex)
End Sub
```

NAPOMENA:

Događaj Click ListBox kontrole se izvršava jedino u slučaju da se u listi nalazi bar jedna stavka. U suprotnom ovaj događaj se neće izvršiti.

Konstrukcija ListBoxObjekt.List (ListBoxObjekt.ListIndex) se često koristi kako bi dobili sadržaj stavke na koju je kliknuto mišem. ListBoxObjekt je svojstvo ime (Name) ListBox kontrole, u našem slučaju Brojanje.

U ovom primeru varirajte svojstva **Columns**, **MultiSlect** i **Style** ListBox kontrole kako bi videli efekte ovih svojstava.

ComboBox:

Standardna Windows kontrola koja je kombinacija dve kontrole: TextBox-a i ListBox-a. Inicijalno je prikazana kao TextBox koji sa desne strane ima strelicu (osim u jednom slučaju). Pritiskom na strelicu otvarate ListBox. Izborom stavke iz liste, prenosite je u TextBox. Sva svojstva, metode i događaji koji su navedeni za ListBox kontrolu, važe i za ComboBox kontrolu. Međutim, svojstvo Style se bitno razlikuje, i može uzeti naredne vrednosti:

0 - Dropdown Combo	inicijalna vrednost. ComboBox prikazuje strelicu za otvaranje liste, a u tekstualni deo je moguće upisati tekst.
1 - Simple Combo	ComboBox ne prikazuje strelicu i lista se ne može otvoriti. Izbor iz liste se vrši kurzorskim tasterima gore i dole. U tekstualni deo je moguće upisati tekst.
2 - DropDown List	ComboBox prikazuje strelicu za otvaranje liste, ali u tekstualni deo nije moguće upisati tekst. Moguće je samo preneti stavku iz liste. Ovo je korisno kada želite da ograničite unos na samo određene, unapred definisane vrednosti.

ComboBox ne poseduje svojstvo MultiSelect, a samim tim ni svojstva Selected i SelectCount. Najčešće su u upotrebi događaji Click i Change. Poslednji se izvršava prilikom izmene teksta u tekstualnom delu i obično se u zavisnosti od sadržaja preduzimaju određene akcije.

HORIZONTAL SCROLL BAR & VERTICAL SCROLL BAR

Obe kontrole su identične osim po horizontalnoj ili vertikalnoj orijentaciji na ekranu. Najčešće se koristi za vizalno postavljanje neke vrednosti, pomeranje sadržaja ekrana i/ili kontrola po ekranu. Moguće je podesiti minimalnu i maksimalnu vrednost, korak izmene vrednosti u dizajn režimu i u vreme izvršavanja programa. Takođe je moguće postaviti i pročitati aktuelnu vrednost (poziciju) klizača za vreme pomeranja ili po završenom pomeranju. Prilikom izvršavanja programa vrednost kontrole možete pomoću miša podešavati na tri načina:

- pritiskom na strelice levo i desno, odnosno gore i dole zavisno da li je u pitanju horizontalna ili vertikalna traka za pomeranje. Ovaj način se zove small change (mala izmena)
- klikom na levu ili desnu, odnosno gornju ili donju stranu kontrole u odnosu na aktuelnu poziciju klizača. Ovaj način se zove large change (velika izmena)
- klikom miša na sam klizač, držite taster pritisnuut i povlačite miša levo-desno, odnosno gore-dole. Ovaj način se zove scroll (pomeranje)

Pomoći tastature, kada kontrola ima fokus, malu izmenu vršite pomoću kurzorskih tastera, veliku pomoću tastera PageUp i PageDown. Na početnu (Min) vrednost se postavljate pritiskom na taster Home, a na krajnju (Max) pritiskom na taster End.

Svojstva:

Min

Svojstvo koje određuje minimalnu vrednost kontrole, kada je klizač u krajnjem levom odnosno gornjem položaju. Validne vrednosti za ovo svojstvo su celi brojevi od -32768 do 32767.

Max

Svojstvo koje određuje maksimalnu vrednost kontrole, kada je klizač u krajnjem desnom odnosno donjem položaju. Validne vrednosti su iste kao za Min svojstvo.

Value

Svojstvo pomoću koga postavljamo ili čitamo vrednost (poziciju) klizača kontrole. Vrednost svojstva Value se kreće od predhodno postavljenog svojstva Min do svojstva Max.

NAPOMENA:

Ako postavljate vrednost svojstva Value u vreme izvršavanja programa, vodite računa da ova vrednost ne bude veća od svojstva Max i ne bude manja od svojstva Min. Ako prekoračite ove granice, Visual Basic će generisati grešku ***Invalid Property Value*** i zaustaviti program.

Nije pogrešno postaviti svojstvo Min veće od svojstva Max. U ovom slučaju svojstvo Value će rasti u obrnutom smeru od uobičajenog.

SmallChange

Podešavanjem ovog svojstva regulišete korak za koji se pomera klizač kontrole kada vršite malu izmenu. Inicijalno ova vrednost je postavljena na 1.

LargeChange

Reguliše korak pomeranja klizača prilikom velike izmene. Inicijalno je takođe postavljen na 1. Zavisno od konkretne potrebe, tipično se vrednost ovog svojstva postavlja na 10% opsega vrednosti kontrole.

Na primer ako su svojstva Min=0, Max=200, ima smisla svojstvo LargeChange postaviti na (Max-Min)/10 = 20.

Događaji:

Change

Događaj se aktivira prilikom izmene svojstva Value, tj prilikom pomeranja klizača. U slučaju da klizač kontrole pomeramo direktno, scroll načinom, ovaj događaj se aktvira u momentu prestanka pomeranja, odnosno onda kada otpustimo taster miša.

Scroll

Događaj se aktivira samo prilikom direktnog pomeranja klizača. U ostalim načinima izmena vrednosti Value kontrole se ne aktivira. Zgodno se može iskoristiti na primer za interaktivno skaliranje slike na formi, kada korisniku želimo da pružimo vizualni prikaz onoga što radi.

Najčešće nam je potrebno da kontrolišemo bilo kakvu izmenu ScrollBar kontrole. U tom slučaju je zgodno napraviti jednu subrotinu (ili funkciju zavisno od potrebe), i pozivate je kod oba događaja i Change i Scroll. Naravno možete isti kod napisati u oba događaja, ali bi to značilo rasipanje resursa.

Primer:

Otvorite nov projekt i na formu HScrollBar kontrolu. Inicijalni naziv kontrole je HSCroll1. Postavite Label kontrolu, inicijalni naziv je Label1. Potom otvorite prozor svojstava HScroll1 kontrole (F4) i postavite sledeća svojstva:

```
Min             0
Max             200
LargeChange     20
```

Za proceduru događaja Change kontrole HScroll1 napišite sledeći kod:

```
Private Sub HScroll1_Change()
      Label1.Caption = HScroll1.Value
End Sub
```

Pokrenite program (F5). Prilikom pomeranja HorizontalScrollBar kontrole tekst u Label kontroli prikazuje aktuelnu vrednost svojstva Value HorizontalScrollBar kontrole. Obratite pažnju da prilikom direktnog pomeranja klizača labela prikazuje vrednost kontrole tek kada otpustite taster miša. Primera radi, napišite istu liniju za Scroll događaj kontrole HScroll1. Sada će labela prikazivati vrednost kontrole stalno, i za vreme direktnog pomeranja klizača.

TIMER kontrola

Timer kontrola se koristi kada želimo da u jednakim vremenski intervalima izvršavamo neku akciju. Ta akcija može biti trivijalna, na primer ispis tekućeg vremena i datuma na formi, a može se iskoristiti za mnogo složenije svrhe: pravljenje rezervne kopije (backup podataka), alarm u programu tipa rokovnik i raspored, vremenski kontrolisano štampanje periodičnih izveštaja, slanje faksova i slično. Timer kontrola nam omogućava da postavimo vremenski interval nakon koga će se aktivirati određeni programski deo. Takođe možemo programski uključivati i isključivati ovu kontrolu. Zahvaljujući ovoj kontroli postaje jednostavno pisanje "rezidentnih" rutina i kompletnih programa koji su stalno aktivni u memoriji, a aktiviraju se na vremenskoj osnovi ili ih aktivira korisnik nekom svojom akcijom. Pisanje ovakvih programa u DOS okruženju je bilo relativno složeno i zahtevalo je poznavanje BIOS prekida i raznorazna žongliranja sa registrima i stekom procesora.

NAPOMENA:

Prilikom intenzivnih operacija sa diskom, računskih operacija i sličnih koje znatno upošljavaju procesor, Timer kontrola neće tačno odbrojavati vreme. Zbog ovoga ona nije podesna za pisanje nekih vremenski intenzivnih ispitivanja i analiza procesa.

Svojstva:

Interval:

Određuje nakon kog vremenskog intervala će kontrola pokrenuti proceduru. Validne vrednosti za ovo svojstvo su od 0 to 65535. Ako je svojstvo postavljeno na 0, Timer kontrola nije aktivna. Svaki drugi broj u ovom intervalu predstavlja broj milisekundi (1/1000 sekundi) nakon čega se procedura događaja vezana za Timer kontrolu aktivira. Dakle, ako želimo da se timer aktivira nakon svake 2 sekunde, postavićemo ovo svojstvo na 2000.

Enabled

Svojstvo koje aktivira i deaktvira Timer kontrolu. Shodno ovome može imati dve vrednosti True i False.

Oba ova svojstva možete postaviti i čitati u dizajn režimu i u vreme izvršavanja programa.

Događaji:

Timer kontrola posedije samo jedan događaj pod nazivom Timer. Kod koji upišete u ovu proceduru događaja se izvršava prilikom aktiviranja Timer kontrole.

NAPOMENA:

U slučaju da se kod ovog događaja izvršava dugo (duže nego što je postavljeno svojstvo Interval Timer kontrole), neće se dogoditi nikakav nasilni prekid izvršavanja procedure, pa ponovni početak. Timer kontrola će uvek potpuno izvršiti proceduru događaja Timer. Međutim može se dogoditi da se vreme potrebno za izvršenje procedure i podešeni interval Timer kontrole prekriju. Rezultat ovoga je da se procedura događaja Timer stalno izvršava, što može usporiti ili čak potpuno zaustaviti izvršavanje Vaše i ostalih aplikacija koje su pokrenute. Da bi ste ovo sprečili koriste se dva načina:

- na početku procedure događaja Timer deaktivirajte Timer kontrolu, a na kraju je aktivirajte ponovo. Ako Timer kontrola ima naziv Timer1 ovo ćete uraditi ovako:

Deaktiviranje:

```
Timer1.Enabled = False
```

Aktiviranje:

```
Timer1.Enabled = True
```

Ovim se postiže da vremenski interval između završetka Timer procedure događaja i ponovnog početka bude zaista vremenski interval koji je postavljen u svojstvu Interval.

- na ključnim mestima u Timer proceduri događaja vršite pozive sistemske Windows funkcije DoEvents(). Ova funkcija je detaljno objašnjena kasnije.

Primer:

U primeru se Timer kontrola koristi za periodično ispisivanje tekućeg vremena na naslovu forme. Otvorite nov projekt i na formu postavite jednu Timer kontrolu. Inicijalni naziv je Timer1. Otvorite Prozor svojstava Timer1 kontrole (F4) i postavite sledeće vrednosti svojstava:

```
Enabled          True
Interval         1000
```

Potom u proceduri događaja Timer, Timer1 kontrole napišite sledeće:

```
Private Sub Timer1_Timer()
      Me.Caption = Format$(Now, "hh:mm:ss")
End Sub
```

Pokrenite primer (F5). Primetićete da po isteku jedne sekunde (Interval = 1000), Timer procedura događaja Timer1 kontrole ispisuje tekuće vreme na naslovu forme u formatu sati:minuta:sekundi.

DRIVE LIST BOX

Kontrola koja prikazuje sve disk jedinice u sistemu uključujući i mapirane mrežne diskove bilo da je u pitanju Novel, Windows 3.11, 95, NT ili neka druga mreža u pitanju. Sa leve strane oznake disk jedinice se nalazi odgovarajuća ikona zavisno od toga da li je u pitanju flopi drajv, hard disk, CD Rom uređaj ili mapirani mrežni disk. Sa desne strane oznake diska se nalazi njegova labela u slučaju hard diska, odnosno ime mrežnog kompjutera sa deljenim nazivom (*shared name*) diska i/ili direktorijuma.

Ova kontrola se ponaša kao ComboBox kontrola kod koje je svojstvo Style podešeno na 2 - Dropdown List. Logično, ova lista je samo za čitanje.

Svojstva:

Drive:

Svojstvo pomoću kojeg možemo čitati i postavljati aktivnu disk jedinicu ove kontrole. Tip ovog svojstva je string, a validne vrednosti su imena disk jedinica iza kojih sledi dvotačka. Na primer:

```
Drive1.Drive = "a:"
Drive1.Drive = "c:"
Drive1.Drive = "\\net3\deljen"
```

Ako postavite vrednost ovog svojstva na disk koji ne postoji ili nije trenutno dostupan, Visual Basic će generisati grešku "*Device unavailable*".

ListCount:

Svojstvo identično istoimenom za ListBox i ComboBox. U ovom slučaju daje broj svih trenutno raspoloživih diskova u sistemu i mreži. Samo za čitanje.

ListIndex:

Isto kao kod ListBox-a i ComboBox-a. Služi za čitanje i postavljanje aktivne stavke u listi disk jedinica. Obratite pažnju da prva stavka ima indeks 0, a poslednja n-1 gde je n ukupni broj stavki u kontroli (ListCount).

List:

Vraća sadržaj određene stavke liste disk jedinica. Sintaksa je List(Index) gde je index vrednost od 0 do broja stavki u listi - 1. Konstrukcija Drive1.List(0) će vratiti prvi disk u sistemu (najčešće flopi disk a:). Tip podataka koje vraća ovo svojstvo je string. Svojstvo je samo za čitanje.

Događaji:

Change:

Događaj koji se aktivira prilikom izmene aktivne disk jedinice bilo akcijom korisnika ili programski.

DIR LIST BOX:

Kontrola koja daje listu direktorijuma tekuće disk jedinice. Ikone mogu predstavljati otvoren ili zatvoren direktorijum i izgledaju isto kao u Windows Explorer-u. Ova kontrola se najčešće koristi u sprezi sa Drive kontrolom.

Svojstva:

Path:

Svojstvo koje služi sa čitanje ili postavljanje aktivnog diska i/ili direktorijuma koji će biti otvoren i čiji sadržaj će biti prikazan u listi. Tip podataka ovog svojstva je string. Konstrukcija `Dir1.Path = "c:\windows"` će postaviti aktivni disk na disk c: i otvoriće direktorijum \Windows na njemu.

NAPOMENA:

Ako je aktivna staza neki direktorijum na disku, svojstvo Path će vratiti string u kojem se nalaze disk i staza. Na primer "c:\windows\system", ali bez poslednjeg backslash ("\") karaktera. Sa druge strane, ako je aktivna staza koren (root) nekog diska npr diska c:, svojstvo Path će vratiti "c:\" , dakle poslednji karakter će biti backslash. O ovome treba voditi računa, i ispitati da li je poslednji karakter svojstva Path backslash, pa zavisno od potrebe ga dodati ili oduzeti.

ListCount, Listindex, List:

Svojstva su identična kao kod predhodne kontrole uz jednu razliku. **ListIndex** otvorenog direktorijuma će uvek biti -1. Indkesi svih direktorijuma iznad otvorenog će imati indekse za po jedan manje, znači -2, -3 ... Indeksi direktorijuma ispod otvorenog imaju regularne vrednosti počev od nule pa na dalje. Ovo daje bolju kontrolu, jer u svakom momentu možemo utvrditi koji direktorijum je otvoren (aktivan), jer ima indeks -1. Potom možemo u pozitivnom delu indeksa (uključujući i nulu), pročitati sve direktorijume ispod, a u negativnom delu sve direktorijume iznad tekućeg.

Događaji:

Change

Događaj koji se aktivira prilikom izmene aktivnog direktorijuma odnosno njegovim otvaranjem. Sa korisničke strane u vreme izvršavanja programa, to se postiže duplim klikom na naziv direktorijuma.

Click

Aktivira se kada korisnik uradi klik mišem na bilo koji stavku u listi.

FILE LIST BOX

Kontrola koja daje listu fajlova aktivnog (otvorenog) direktorijuma. Najčešče se koristi u sprezi sa predhodne dve kontrole. Moguće je podesiti koji fajlovi će biti prikazani u zavisnosti od njihovih atributa i ekstenzije.

Svojstva:

Archive, Hidden, Normal, System, ReadOnly

Svojstva koja određuju da li će fajlovi sa odgovarajućim atributima biti prikazani ili ne. Sva četiri svojstva mogu imati vrednosti True ili False. Inicijalno, svojstva Archive i Normal su postavljena na True, a svojstva Hidden i System su postavljena na False.

Pattern:

Svojstvo koje nam daje mogućnost da prikažemo fajlove koji po imenu i ekstenziji zadovoljavaju zadati kriterijum. Tip podataka za ovo svojstvo je string. Na primer, ako ovo svojstvo postavite na "*.exe" u listi će se naći samo fajlovi sa ekstenzijom "exe" - izvršni programi. U slučaju da želite da kombinujete više kriterijuma, potrebno je da ih razdvojite znakom ";".

"*.exe;*.bat;*.com" će izlistati sve fajlove sa ekstenzijama exe, bat, i com u tekućem direktorijumu. Ovaj kriterijum nije ograničen samo na ekstenzije. Setite se DOS komande DIR. Sve varijante važe i ovde. Na primer, "W*.exe;*.d?t" će izlistati sve fajlove koji počinju sa "w" i imaju ekstenziju exe i sve fajlove sa ekstenzijama .dat, .dbt, dct ... , znak ? zamenjuje bilo koji karakter.

List, ListIndex, ListCount, MultiSelect, Selected

Ova svojstva su identična kao kod ListBox kontrole.

PRIMER:

Sledi jednostavan primer u kome je omogućen pregled, disk jedinica sistema, direktorijuma i fajlova u njima. Demonstrira povezivanje DriveList, DirList i FileList kontrola. Takođe je moguće zadati kriterijum po kome će biti prikazani fajlovi u FileList kontroli u smislu naziva, ekstenzije i atributa. Ako izvršite dupli klik mišem na listu fajlova, otvoriće se prozor za poruke u kome će biti prikazan pun naziv fajla, sa stazom i samim njegovim imenom. Ovde je vođeno računa o tome da Path svojstvo na kraju ima BackSlash karakter samo ako je u pitanju koren diska. U proceduri događaja DblClick (dupli klik), FileList kontrole se ispituje da li na kraju svojstva Path postoji BackSlash karakter, i ako ne postoji programski ga dodaje.

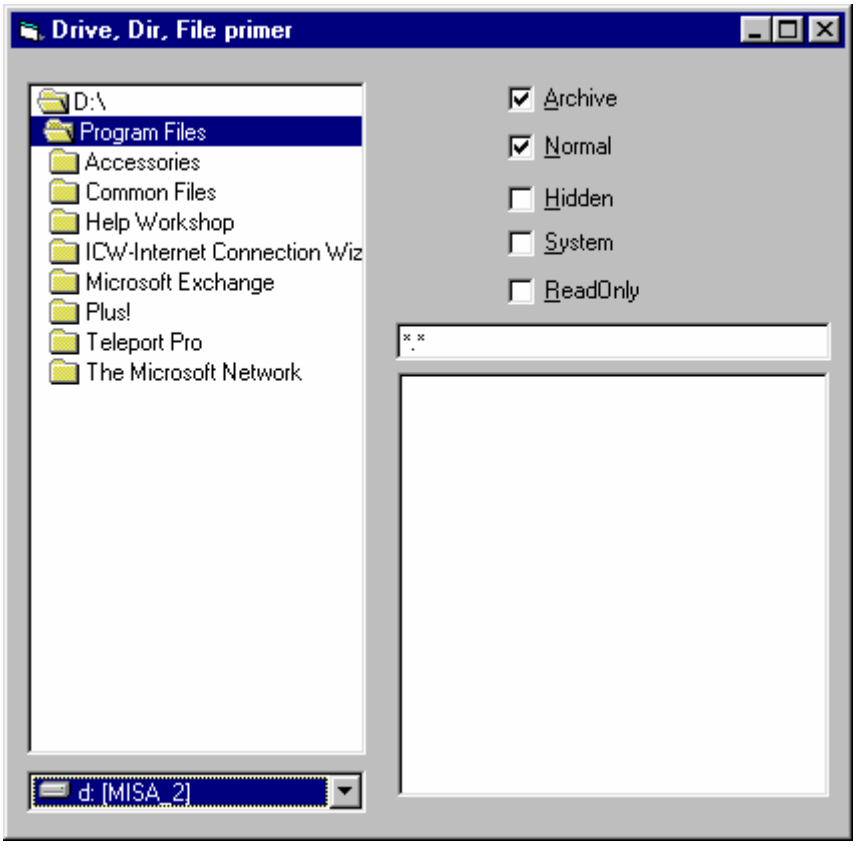
Otvorite nov projekt i na formu dodajte sledeće komponente:
- DriveListBox
- DirLIstBox
- FileListBox
- pet CheckBox kontrola (služiće nam za postavljanje atributa fajlova)
- TextBox (za postavljanje filter kriterijuma za prikaz fajlova)

Za svaku od pet CheckBox kontrola postavite sledeća svojstva:

Svojstvo	Name	Caption	Value
1	cArchive	&Archive	1 - Checked
2	cNormal	&Normal	1 - Checked
3	cHidden	&Hidden	0 - Unchecked
4	cSystem	&System	0 - Unchecked
5	cReadOnly	&ReadOnly	0 - Unchecked

Svojstvo Text, TextBox kontrole postavite na ***.***

Za sve ostale kontrole ostavite inicijalna imena i vrednosti svojstava.
Slike forme sa raspoređenim kontrolama:



Sledi kod koji povezuje kontrole:
Za Change proceduru događaja DriveList kontrole:

```
Private Sub Drive1_Change()
      Dir1.Path = Drive1.Drive
End Sub
```

Za Change proceduru događaja DirListKontrole:

```
Private Sub Dir1_Change()
      File1.Path = Dir1.Path
End Sub
```

Za DblClick proceduru događaja FileListkontrole:

```
Private Sub File1_DblClick()
      ' Promenjiva u koju smestamo stazu i ime fajla
      Dim sNaziv As String

      ' Proveravamo da li Path svojstvo na kraju ima
      ' BackSlash "\" karakter
      If Right$(File1.Path, 1) <> "\" Then
            ' Ako nema dodajemo ga
            sNaziv = File1.Path & "\"
      Else
            ' Ako vec postoji samo vrsimo dodelu vrednosti
            sNaziv = File1.Path
      End If

      ' Dodajemo ime fajla koji je selektovan
      sNaziv = sNaziv & File1.filename
      ' U dijalogu za poruke prikazujemo pun naziv fajla
      MsgBox sNaziv, vbInformation, "Primer"
End Sub
```

U ovom primeru, korisnik može u TextBox ukucati jedan ili više filtera kao kriterijum za prikaz fajlova u listi fajlova. Više kriterijuma se razdvaja znakom tačka zarez; na primer *.exe;*.dll. Nakon upisanog kriterijuma, zgodno je koristiti KeyDown događaj TextBox kontrole i ako je korisnik pritisnuo Enter taster, onda primeniti filter.
KeyDown događaj TextBox kontrole:

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)
      ' Ako je korisnik pritisnuo Enter taster (kod 13), primeni filter
      If KeyCode = 13 Then
            File1.Pattern = Text1.Text
      End If
End Sub
```

Na kraju sledi kod koji postavlja filter u smislu atributa fajlova koji će biti prikazani u listi fajlova. Za svaku od pet CheckBox kontrola je napisan kod u proceduri događaja Click. Ispitujemo svojstvo Value i zavisno od njega postavljamo odgovarajuće svojstvo File1 FileList kontrole na True ili False. Ne gubite iz vida da svojstvo Value CheckBox kontrole može imati tri vrednosti:

0- Nečekirano
1- Čekirano
2- Nedefinisano

Numerički, konstanta False ima vrednost 0, a konstanta True ima bilo koju vrednost različitu od 0. Na osnovi ovoga vrednosti 1-Ćekirano i 2-Nedefinisano imaju vrednost True. Zbog toga je neophodno ispitati vrednost svojstva Value CheckBox kontrole, pa zavisno od toga postaviti dogovarajuća svojstva FileList kontrole.

```
Private Sub cArchive_Click()
      If cArchive.Value = 1 Then File1.Archive = True
      If cArchive.Value = 0 Then File1.Archive = False
End Sub

Private Sub cNormal_Click()
      If cNormal.Value = 1 Then File1.Normal = True
      If cNormal.Value = 0 Then File1.Normal = False
End Sub

Private Sub cHidden_Click()
      If cHidden.Value = 1 Then File1.Hidden = True
      If cHidden.Value = 0 Then File1.Hidden = False
End Sub

Private Sub cSystem_Click()
      If cSystem.Value = 1 Then File1.System = True
      If cSystem.Value = 0 Then File1.System = False
End Sub

Private Sub cReadOnly_Click()
      If cReadOnly.Value = 1 Then File1.ReadOnly = True
      If cReadOnly.Value = 0 Then File1.ReadOnly = False
End Sub
```

Ilustracije radi je korišćen ovakav pristup, međutim efikasnije bi bio da smo CheckBox kontrole postavili kao niz kontrola sa istim imenom i indeksom od 0 do 4, i napisali kod samo na jednoj proceduri događaja koristeći linijski IIf uslov. Na primer ako je naziv niz CheckBox kontrola cAtributi procedura događaja Click bi izgledala ovako:

```
Private Sub cAtributi_Click(Index As Integer)
      Select Case Index
            Case 0
                  File1.Archive = IIf(cAtributi(0).Value = 1, True, False)
            Case 1
                  File1.Normal = IIf(cAtributi(1).Value = 1, True, False)
            Case 2
                  File1.Hidden = IIf(cAtributi(2).Value = 1, True, False)
            Case 3
                  File1.System = IIf(cAtributi(3).Value = 1, True, False)
            Case 4
                  File1.ReadOnly = IIf(cAtributi(4).Value = 1, True, False)
      End Select
End Sub
```

SHAPE

Jednostavna kontrola koja služi za crtanje geometrijskih oblika. Moguće je izabrati tip geometrijskog oblika, boju, ispunu, tip linije kojom se crta, kao i nekoliko predefinisanih vrsta šrafiranja. Može se koristiti prilikom dizajniranja formi i dijaloga. Na žalost, ova kontrola nema svojih događaja. Nije moguće u izvršnom režimu jednostavno detektovati klik miša na nju. Kada bi ovo postojalo bilo bi veoma jednostavno napraviti program za vektorsko crtanje. Nadajmo se da će Microsoft u narednoj verziji Visual Basica u ovom smislu unaprediti Shape kontrolu.

Svojstva

Shape:

Određuje vrstu geometrijske figure. Moguće vrednosti su:

0 - Rectangle	(pravougaonik)
1 - Square	(kvadrat)
2 - Oval	(elipsa)
3 - Circle	(krug)
4 - Rounded Rectangle	(pravougaonik sa zaobljenim ivicama)
5 - Rounded Square	(kvadrat sa zaobljenim ivicama)

BackStyle:

Određuje da li će pozadina geometrijskog oblika biti vidljiva ili ne. Moguće vrednosti ovog svojstva su:

0 - Transparent	(oblik je providan, sve kontrole ispod njega se vide)
1 - Opaque	(oblik nije providan)

BackColor:

Unutrašnja boja oblika. Ovo svojstvo ima smisla samo ako je svojstvo BackStyle postavljeno na 1 - Opaque

BorderColor:

Boja okvira geometrijskog oblika.

BorderStyle:

Način iscrtavanja okvira geometrijskog oblika. Moguće su sledeće predefinisane vrednosti:

0 - Transparent	(okvir je providan, ne iscrtava se)
1 - Solid	(puna linija)
2 - Dash	(crtice)
3 - Dot	(tačke)
4 - Dash-Dot	(crta-tačka)
5 - Dash-Dot-Dot	(crta-tačka-tačka)
6 - InsideSolid	(spoljna ivica okvira je istovremeno i spoljna ivica oblika)

BorderWidth:

Debljina okvira. Inicijalna vrednost je 1. Validne vrednosti ovog svojstva su od 1 to 8192 i označavaju debljinu u twips jedinicama. Zavisno od svojstva BorderStyle, širina okvira se različito ponaša:

Ako je BorderStyle=0 (Transparent), debljina okvira naravno nema uticaja jer se okvir ni ne vidi.

Ako je BorderStyle od 1 do 5 (uključivo), širina okvira raste od centra (ili bolje rečeno od ose) okvira geometrijskog oblika podjednako sa spoljne i unutrašnje strane.

Ako je BorderStyle=6 (InsideSolid), debljina okvira raste samo ka unutrašnjoj strani geometrijskog oblika.

DrawMode:

Svojstvo koje definiše kako se geometrijski oblik ponaša u odnosu na pozadinu ili neku drugu kontrolu sa kojom se seče. Inicijalna vrednost je 13 - CopyPen, što znači standardno iscrtavanje preko ili ispod drugih kontrola zavisno od svojstva ZOrder. Moguće je izabrati jednu od ukupno 16 drugih varijanti koje predstavljaju kombinacije logičkih izraza Not, Or, Xor ... koje se izvode nad pojedinačnim pikselima kontrole i pozadine ili druge kontrole. Ovde nećemo ulaziti u sve varijante ovog svojstva. Ako želite, postavite na formu dva geometrijska oblika i variranjem ovog svojstva uočite efekte.

FillColor:

Boja ispune geometrijskog oblika.

FillStyle:

Način ispune geometrijskog oblika. Moguće su sledeće predefinisane vrednosti:

0 - Solid	(puno bojenje oblika sa FillColor bojom)
1 - Transparent	(ne vrši se punjenje oblika, on preuzima boju definisanu BackColor svojstvom, pod uslovom da je svojstvo BackStyle postavljeno na 1 - Opaque, u suprotnom oblik je providan)

U narednim varijantama u pitanju su razni tipovi šrafura. Potrebno je imati u vidu da se šrafura iscrtava u boji FillColor, dok je ispuna oblika data bojom BackColor pod uslovom da je svojstvo BackStyle postavljeno na 1 - Opaque. Ako je ovo svojstvo postavljeno na 0 - Transparent, iscrtava se samo šrafura u FillColor boji dok je pozadina transparentna.

2 - Horizontal Line
3 - Vertical Line
4 - Upward Diagonal (dijagonalne linije \\\)
5 - Downward Diagonal (dijagonalne linije ///)
6 - Cross
7 - Diagonal Cross

LINE

Takođe jednostavna kontrola za iscrtavanje linije. Nedostaju procedure događaja kao i kod Shape kontrole. Takođe nedostaje 3D opcija iscrtavanja linije, tako da je postižemo iscrtavanjem dve iste linije od kojih je jedna postavljena odmah ispod druge i obično u tamno sivoj boji.

Svojstva:

BorderColor:

Boja iscrtane linije.

BorderStyle, BorderWidth, DrawMode:

Ista uloga i predefinisane vrednosti kao kod istoimenih svojstva Shape kontrole.

X1,Y1; X2,Y2

Koordinate početka (X1,Y1) i kraja (X2,Y2) linije. Koordinate su date relativno u odnosu na kontejner objekat na kome je nacrtana linija. Ako je kontejner objekat sama forma (linija je nacrtana na formi), u pitanju su apsolutne koordinate. Takođe je zavisna od ScaleHeight, ScaleWidth, ScaleLeft i ScaleTop svojstva forme na kojoj je nacrtana.

Jedinica mere ovih svojstava je zavisna od ScaleMode svojstva forme na kojoj se nalazi linija i inicijalno je postavljena na twips-ove. Ostale mogućnosti su objašnjenje u poglavlju Visal Basic Forme.

PICTUREBOX

Veoma kompleksna i korisna kontrola. Omogućava prikaz slike snimljene u nekoliko standardnih grafičkih formata:

Bitmapirane slike	(*.bmp; *.dib)
Gif format	(*.gif)
JPEG	(*.jpg)
Windows metafile	(*.wmf; *.emf)
Ikone, kurzori	(*.ico; *.cur)

Zavisno od broja raspoloživih boja sistemski postavljenih u Windows-u, prikazaće slike neizmenjene ili će po potrebi svesti boje na sistemsku paletu. Sliku je moguće postaviti ili u dizajn režimu ili u vreme izvršavanja programa. Vodite računa da će, ako sliku postavite u dizajn režimu, dužina izvršne verzije vaše aplikacije biti povećana za veličinu slike. U većini situacija pogodnije je sliku učitavati u kontrolu za vreme izvršavanja. Takođe treba znati da se najbrže učitavaju bitmapirane slike, a najduže slike u JPEG formatu, zbog neophodne dekompresije. Inače sam Visual Basic sve slike interno čuva u bitmap formatu. Ako Vam ova kontrola služi isključivo za prikaz slike, bolje je koristiti Image kontrolu koja će biti obrađena u narednom delu, jer je ona brža, manje složena i samim tim zauzima manje resursa računara.

Drugi, verovatno značajniji aspekt ove kontrole je mogućnost crtanja i pisanja po njoj. Često se koristi za prikaz linijskih grafikona kreiranih iz baze podataka, inženjerskih proračuna i slično. Mogućnost skaliranja, podešavanja korisničkih kordinata, definicija koordinatnog početka je čine veoma fleksibilnom i jednostavnom za upotrebu. Prilikom crtanja, možete, ako želite predhodno postaviti sliku i raditi preko nje.

Konačno ova kontrola može biti bound (vezana) za bazu podataka. Tipično se vezuje za polje tipa OLE Object (Long Binary), i služi za prikaz slika koje smeštamo direktno u bazu podataka. O ovome više na kursu "Visual Basic i baze podataka".

Svojstva:

Picture:

Svojstvo koje određuje sliku sa Vašeg diska koju želite da prikažete. Moguće je ovo svojstvo postaviti u dizajn i/ili izvršnom režimu. Ako je slika definisana u dizajn režimu ona je ugrađena (embeed) u aplikaciju. Sve daljnje izmene na orginalu, ne utiču na sliku u Visual Basic aplikaciji. U vreme izvršavanja sliku učitavamo pomoću funkcije LoadPicture. Sintaksa ove funkcije je:

LoadPicture (PunoImeFajla)

PunoImeFajla je disk, direktorijum i ime slike sa ekstenzijom. Funkcija se koristi za učitavanje slike u bilo koji objekat ili kontrolu koja može sadržati sliku. Podržani formati su prikazani na početku prikaza PictureBox kontrole. Primeri upotrebe:

```
Set Picture1.Picture = LoadPicture ("c:\windows\winlogo.bmp")
Set Form1.Picture = LoadPicture ("d:\slike\ccindy1.jpg")
```

NAPOMENA:

Ako želite da izbrišete sliku učitanu bilo u dizajn ili u vreme izvršavanja programa, metoda Cls (objašnjena kasnije u ovom poglavlju), neće pomoći. Ovo možete izvršiti na dva načina:

1. Koristite LoadPicture funkciju sa praznim stringom kao argumentom:

```
Set Picture1.Picture = LoadPicture ("")
```

2. Postavite referencu za sliku na Nothing (ništa):

```
Set Picture1.Picture = Nothing
```

Nothing je ključna reč Visual Basica koja služi da poništi referencu objektne varijable u našem slučaju, pokazivač na sliku.

Align:

Svojstvo koje određuje poziciju kontrole u odnosu na formu. Zahvaljujući postojanju ovog svojstva PictureBox kontrola se može postaviti na MDI formu, a potom na nju ostale kontrole koje ne poseduju Align svojstvo. U ovom kontekstu nam PictureBox kontrola služi kao kontejner. Moguće vrednosti ovog svojstva su:

0 - None	(kontrola je bez poravnanja)
1 - Align Top	(nalazi se na vrhu forme, širine iste kao forma)
2 - Align Bottom	(na dnu forme)
3 - Align Left	(sa leve strane)
4 - Align Right	(sa desne strane forme)

Prilikom izmene dimenzija forme u vreme izvršavanja širina (1 i 2) ili visina (3 i 4) PictureBox kontrole se automatski podešava zavisno od novih dimenzija forme.

AutoRedraw:

Svojstvo koje određuje kada će Visual Basic ponovo iscrtati (osvežiti) sadržaj PictureBox kontrole. Svojstvo može imati dve vrednosti True ili False sa sledećim značenjem:

True: Omogućava automatsko iscrtavanje sadržaja kontrole kada je to potrebno. Slika se čuva i u kontroli i u memoriji računara. U ovom slučaju PictureBox kontrola ne prima Paint događaj. Međutim zbog dvostrukog čuvanja sadržaja slike, povećavaju se potrebe za resursima.

False: Sadržaj slike se čuva samo u kontroli. Kontrola prima Paint događaj i Visual Basic ga aktivira kada je to potrebno. Ovo je inicijalna vrednost svojstva AutoRedraw.

AutoSize:

Određuje da li će kontrola automatski prilagoditi svoje dimenzije zavisno od dimenzija slike koja je učitana u kontrolu. Samim tim svojstvo može imati vrednosti True ili False.

BackColor:

Svojstvo ima uticaja samo prilikom crtanja po kontroli. Ako učitate sliku, postavljena boja pozadine se ignoriše.

DrawMode, DrawStyle, DrawWidth:

Identično istoimenom svojstvu Shape kontrole.

ForeColor:

Boja koja se koristi prilikom crtanja ili ispisa teksta.

FontTransparent:

Određuje da li je tekst ispisan na kontroli sa transparentnom pozadinom (True) ili ne (False).

ScaleWidth, ScaleHeight:

Svojstva koja određuju logiču širinu (x koordinata) i visinu (y koordinata) kontrole. Inicijalno vrednosti ovih svojstava su jednake širini (Width) i visini (Height) same PictureBox kontrole. Ako želite da bez obzira na stvarne dimenzije kontrole postavite opseg koordinatnog sistema onako kako Vam najviše odgovara to učinite na sledeći način:

Na primer potreban Vam je koordinatni sistem PictureBox kontrole pod nazivom Picture1, gde vrednosti za X osu idu od 0 do 100, a za Y od 0 do 50:

```
Picture1.ScaleWidth = 100
Picture1.ScaleHeight = 50
```

Od ovog momenta koordinata leve gornje tačke je (0,0), a donje desne tačke (100,50). Ako promenite bilo koje od ova dva svojstva, obratite pažnju da ScaleMode svojstvo automatski prelazi na 0-User.

Ova mogućnost je izuzetno korisna. Na primer korišćenjem iste rutine za crtanje, izmenom ScaleWidt i ScaleHeight svojstva možete vršiti skaliranje slike, deformacije po X i/ili Y osi, efekte zoom-in i zoom-out.

ScaleTop, ScaleLeft:

Svojstva koja određuju poziciju koordinatnog početka u odnosu na gornju levu tačku PictureBox kontrole. Sledeća sekvenca postavlja koordinatni početak u sredinu kontrole, tako da obe ose imaju pozitivan i negativan smer, a potom crta krug sa centrom na koordinatama (0,0) i poluprečnikom 20.

```
Picture1.ScaleWidth = 100
Picture1.ScaleHeight = 100
Picture1.ScaleTop = -50
Picture1.ScaleLeft = -50
Picture1.Circle (0, 0), 20
```

ScaleTop i ScaleLeft svojstva, ukratko rečeno, rade translaciju koordinatnog početka i veoma je jednostavno istovremeno pomerati i crtež ili sliku koja se nalazi na kontroli.

CurrentX, CurrentY:

Svojstva koja čitaju ili postavljaju aktuelnu poziciju od koje će kasnije krenuti crtanje ili ispis teksta na PictureBox kontroli. Najčešće se koristi prilikom ispisa teksta. Prvo se postave CurrentX i CurrentY svojstva, a potom se metodom Print odštampa tekst počev od te pozicije. Jedinice mere su zavisne od vrednosti ScaleWidth, ScaleHeight i ScaleMode svojstava

Metode:

Cls:

Metoda koja briše sadržaj PictureBox kontrole, ali samo elemente koji su nacrtani ili ispisani za vreme izvršavanja programa. Slike učitane za vreme izvršavanja programa ili u dizajn režimu ostaju neizmenjene.

PSet:

Metoda koja crta tačku na datoj koordinati i boji. Sintaksa metode je:
objekt.PSet [Step] (x, y), [color]
- objekt je naziv objekta koji predstavlja PictureBox kontrolu
- [Step] je opcioni parametar i označava da koordinate koje slede nisu apsolutne od koordinatnog početka, već su relativne od poslednje posećene koordinate koje se mogu dobiti ili postaviti pomoću CurrentX i CurrentY svojstva.
- (x,y) obavezni parametri koji označavaju apsolutne ili relativne koordinate nacrtane tačke, zavisno da li je postavljen [step] parametar.
- [color] opcioni parametar koji određuje boju tačke. Može se koristiti RGB ili QBColor funkcija. Ako je izostavljen koristi se boja postavljena u ForeColor svojstvu PictureBox kontrole.

Point:

Pomoću ove metode (koja je samo za čitanje), možemo dobiti boju tačke na zadatoj koordinati u PictureBox kontroli. Ulazni parametri su tipa Single, a vraćena vrednost je tipa LongInteger koji predstavlja RGB prezentaciju boje. Sintaksa je:
Varijabla = object.Point (x, y)
- (x,y) je koordinata tačke

Umereno korisna metoda. Ako želite da nešto što ste iscrtali prenesete na štampač, možete proći tačku po tačku PictureBox kontrole i svaku ponaosob poslati na štampač. Ovo je naravno sporo, ali je još uvek brže od štampača, pa se u tom smislu može koristiti.

Line:

Metod za crtanje linije, praznih i popunjenih okvira. Sintaksa je:

```
objekt.Line [Step] (x1, y1) [Step] - (x2, y2), [color], [B][F]
```

- [Step] isto značenje kao za PSet metod
- (x1,y1) opcioni parametar koji označava koordinate početka linije. Ako se izostavi podrazumevan početak je poslednja posećena koordinata (CurrentX i CurrentY)
- (x2,y2) obavezni parametri, označavaju koordinate kraja linije
- [color] isto značenje kao kod PSet metode
- [B] opcioni parametar Block. Označava da će umesto linije biti nacrtan pravougaonik (ili kvadrat), čija je diagonala data linija. Oblik će biti popunjen bojom određenom sa FillColor i tipom ispune određenim sa FillStyle svojstvima PictureBox-a
- [F] opcioni parametar, označava da će oblik biti popunjen istom bojom kojom je nacrtan okvir oblika. Ne možete staviti parametar F bez parametra B.

Naredni kod crta pravougaonik od koordinate (10,10) do koordinate (1000,2000):

```
Picture1.Line (10, 10)-(1000, 2000), , B
```

Obratite pažnju da je parametar [color] izostavljen, ali je neophodno označiti njegovo mesto pošto smo upotrebili parametar [B] koji sledi iza njega.

Naredni kod crta poligon od dve linije. Početak druge linije se nastavlja na kraj prve pa smo radi toga izostavili početne koordinate:

```
Picture1.Line (100, 100)-(1000, 2000)
Picture1.Line -(3000, 4000)
```

Circle:

Crtanje kruga ili elipse. Sintaksa je:

objekt.Circle [Step] (x, y), radius, [color, start, end, aspect]

- [step] je opcioni parametar sa poznatim značenjem
- (x,y) obavezan parametar koji predstavlja centar kruga ili elipse
- radius je obavezan parametar koji predstavlja prečnik kruga
- [color] opcioni parametar, kao kod PSet metode
- start opciono, označava početnu tačku iscrtavanja kruga ili elipse
- end opciono, označava krajnju tačku iscrtavanja kruga ili elipse. Pomoću parametara start i end, možemo crtati lukove. Oba parametra su izražena u radijanima. Ako su izostavljeni, podrazumevaju se vrednosti 0 radijana za start i 2pi radijana za kraj. Validne vrednosti za oba parametra su u opsegu od -2pi do 2pi.
- aspect, opcioni parametar koji postavlja odnos visine i širine kruga. Ako se izostavi podrazumeva se da je njegova vrednost 1, čime se dobija krug. Ako stavite vrednosti veće od 1 dobićete elipsu čija je visina veća od širine, obrnuto ako stavite vrednost manju od jedan.

Naredni kod crta elipsu čija je visina 1000,a širina 500 sa centrom na koordinatama (2000,2000):

```
Picture1.Circle (2000, 2000), 1000, , , , 2
```

Scale:

Metod pomoću koga možete definisati vlastiti koordinatni sistem slično kao što je prikazano pomoću ScaleWidth i ScaleHeight svojstava. Sintaksa je:

object.Scale (x1, y1) - (x2, y2)

- (x1,y1) gornja leva koordinata
- (x2,y2) donja desna koordinata

Posle ovoga, svojstvo ScaleMode svojstvo se automatski postavlja na 0-User. Ako želite da resetujete ovako kreiran koordinatni sistem, upotrebite ovo svojstvo bez oba para parametar. Sada se ScaleMode svojstvo postavlja na 1-Twips.

TextHeight, TextWidth:

Metode koje vraćaju visinu i širinu teksta. Koriste se kako za PictureBox kontrolu, tako i za formu i Printer objekt. Sintaksa je:

```
objekt.TextHeight(string)
objekt.TextWidth(string)
```

- objekt može biti forma, PictureBox ili Printer objekt
- string, ulazni parametar tipa string čiju širinu i visinu želimo da odredimo

Font koji se pri tome uzima u obzir je aktivni font podešen u objektu. Ovo svojstvo je izuzetno korisno, pogotovu prilikom potrebe da se formatizuje izlaz na štampač. Pošto su TrueType fontovi uglavnom proporcionalni (širina svakog znaka nije ista), bez ove metode bi bilo veoma teško odrediti da li neki tekst može ili ne može da stane na predviđenu poziciju prilikom štampanja.

NAPOMENA:

U svojstvima i metodama se koriste koordinate i grafičke dimenzije. Imajte u vidu da njihova jedinica mere zavisi od ScaleMode svojstva. Takođe vodite računa da prilikom skaliranja ili translacije koordinatnog sistema ScaleMode svojstvo automatski prebacuje na vrednost 0-User. Može se dogoditi da kod, koji je do malopre savršeno crtao dijagram, odjednom više ne radi korektno, ili se čak ništa ne vidi. ScaleMode svojstvo postavljeno na 1-Twips ima red veličine 1000 za koordinate; s druge strane ako je postavljeno na 7-Centimeter to je već red veličine 20-tak jedinica, zavisno od veličine monitora. Sve grafičke elemente Visual Basic će iscrtati bez poruke o grešci, makar one daleko izlazile van vidnog opsega.

Primer za picture box kontrolu:

Jednostavan primer koji demonstrira upotrebu metoda za crtanje elementarnih geometrijskih oblika, omogućava učitavanje slike sa diska i njen prikaz na PictureBox kontroli. Prilikom crtanja koristimo MouseDown događaj. Kod izbora slike sa diska koristimo CommonDialog kontrolu koja je obrađena kasnije.

Otvorite nov projekt i na formu postavite PictureBox kontrolu, pet OptionButton kontrola, tri komandna dugmeta i CommonDialog kontrolu. Pošto CommonDialog kontrola ne postoji u osnovnom setu Visual Basic kontrola moramo je dodati u projekat. Iz menija Project izaberite stavku Components. U narednom dialogu čekirajte stavku pod nazivom "Microsoft Common Dialog Control 5.0" i potom pritisnite taster **OK**. Nakon ovoga u VB toolbox-u je na raspolaganju CommonDialog kontrola i možete je postaviti na formu.
Potom postavljamo svojstva OptionButton kontrola kao što je prikazano u tabeli:

Name	**Caption**	**Value**
oTacka	&Tacka	True
oLinija	&Linija	False
oKrug	&Krug	False
oElipsa	&Elipsa	False
oTekst	T&ekst	False

Svojstva tri komandna dugmeta:

Name	**Caption**
bUcitajSliku	&Ucitaj sliku
bBrisiSliku	&Brisi sliku
bBrisiNacrtano	Brisi &nacrtano

Slika prikazuje izgled gotove forme:

Kod ovog primera se nalazi na Click proceduri događaja svakog od ova tri komandna dugmeta i MouseDown događju PictureBox kontrole. Takođe u FormLoad događaju postavljamo neka svojstva PictureBox kontrole. Za sada zanemarite kod u vezi CommonDialog kontrole. Ona će biti kasnije obrađena.

FormLoad događaj:

```
Private Sub Form_Load()
      ' Logičke koordinate PictureBox kontrole postavljamo na
      ' širina 100, visina 100
      Picture1.ScaleWidth = 100
      Picture1.ScaleHeight = 100
End Sub
```

Učitavanje slike:

NAPOMENA: U sledećem poglavlju je objašnjena CommonDijalog kontrola koja se koristi u ovom primeru

```
Private Sub bUcitajSliku_Click()
      On Error Resume Next
      With CommonDialog1
            .CancelError = True
            .filename = ""
            .DialogTitle = "Otvori sliku"
            .Filter = "Sve slike|*.bmp;*.gif;*.jpg;*.ico;*.cur;*.wmf;|" & _
            "Bitmap slike|*.bmp|" & _
            "JPEG slike|*.jpg|" & _
            "GIF slike|*.gif|" & _
            "Windows metafile|*.wmf;*.emf|" & _
            "Ikone i kurzori|*ico;*cur|"
            .ShowOpen
      End With
      If Err.Number Then
            Err.Clear
            Exit Sub
      End If

      ' U slučaju greške idi na labelu Greska_ucitavanja
      On Error GoTo Greska_ucitavanja

      ' Postavljamo pokazivač miša na "wait" jer učitavanje velike
      ' kompresovane slike može trajati par sekundi
      Screen.MousePointer = vbHourglass

      ' Učitavamo sliku koji je korisnik izabrao
      Picture1.Picture = LoadPicture(CommonDialog1.filename)

Izlaz:
      ' Vraćamo pokazivač miša na "default"
      Screen.MousePointer = vbDefault

      ' Izlazimo iz subrotine
      Exit Sub

Greska_ucitavanja:
      ' U slučaju greške ispisujemo poruku korisniku i opis nastale greške
      MsgBox "Greska prilikom ucitavanja slike" & Err.Description _
            , vbCritical, "LoadPicture funkcija"

      ' Dalje izvršavanje programa nastavljamo od labele Izlaz
      Resume Izlaz

End Sub
```

Brisanje slike:

```
Private Sub bBrisiSliku_Click()
      ' Postavljamo Picture svojstvo na Nothing
```

```
      Picture1.Picture = Nothing

      ' može i ovako: Picture1.Picture = LoadPicture ("")
End Sub
```

Brisanje nacrtanih elemenata:

```
Private Sub bBrisiNacrtano_Click()
      ' Cls metodom
      Picture1.Cls
End Sub
```

Za crtanje elemenata koristimo MouseDown događaj. Za razliku od Click događaja, MouseDown nam daje i koordinate pozicije miša relativno u odnosu na kontrolu na koju smo kliknuli, u našem slučaju PictureBox. Boju crtanja elemenata ćemo izabrati pomoću generatora slučajnih brojeva. Validne vrednosti za boju u RGB obliku su od 0 do 16777215. Kao seme generatora slučajnih brojeva koristimo trenutno sistemsko vreme Vašeg računara.

```
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, _
      X As  Single, Y As Single)

      ' Promenjiva u kojoj čuvamo boju crtanja
      Dim Boja As Single

      ' Boju crtanja dobijamo pomoću generatora slučajnih brojeva
      Boja = Rnd(Now) * 16777215

      ' Postavljamo boju crtanja PictureBox kontrole
      Picture1.ForeColor = Boja

      ' Proceravamo šta je korsnik izabrao da crta zavisno od svojstva Value
      ' OptionButton-a

      ' Crta tačku
      If oTacka.Value = True Then
            Picture1.PSet (X, Y)
            Exit Sub
      End If

      ' Crta liniju od koordinatno početka do pozicije miša
      If oLinija.Value = True Then
            Picture1.Line (0, 0)-(X, Y)
            Exit Sub
      End If

      ' Crta krug sa centrom na poziciji miša i fiksnim poluprečnikom
      If oKrug.Value = True Then
            Picture1.Circle (X, Y), 20
            Exit Sub
      End If

      ' Crta elipsu sa centrom na poziciji miša i fiksnim poluprečnikom
      ' odnos x i y koordinate je 2
      If oElipsa.Value = True Then
            Picture1.Circle (X, Y), 30, , , , 2
            Exit Sub
      End If

      ' Ispisuje tekuće vreme na poziciji miša
      If oTekst.Value = True Then
            ' Postavlja aktivnu X koordinatu
            Picture1.CurrentX = X
```

```
            ' Postavlja aktivnu Y koordinatu
            Picture1.CurrentY = Y

            ' Ispisuje tekuće vreme
            Picture1.Print Time
      End If
End Sub
```

IMAGE:

Kontrola koja nema skup metoda i svojstava kao PictureBox kontrola. Namenjena je uglavnom za prikazivanje gotovih slika. Zahteva manje resursa i brža je od predhodne. Podržava isti set grafičkih formata i može se povezati sa bazom podataka. Nema skup svojstava i metoda za skaliranje, translaciju koordinatnog sistema i crtanje geometrijskih oblika. Međutim poseduje jedno vrlo korisno svojstvo koje nije na raspolaganju kod PictureBox kontrole:

Svojstva:

Stretch:

Svojstvo određuje da li će učitana slika biti skalirana tako da cela stane u dimenzije kontrole. Moguće vrednosti su True ili False. Ako svojstvo postavite na True, bez obzira na dimenzije učitane slike, one će biti automatski skalirane tako da cela slika stane na kontrolu, po potrebi smanjene ili povećane.

Common Dialog

Comon dialog (standardni, uobičajeni dijalog) je kontrola koja Vam omogućava korišćenje standardnih sistemskih Windows dijaloga i to: Open, Save/Save As, Font, Color, Printer. Ove dijaloge možete naravno, kreirati i sami korišćenjem forme, kontrola i uz dosta kodiranja, ali svakako je preporučljivo upoznati i koristiti već postojeće kako bi umanjili neophodni kod i svojim aplikacijama dali konzistentan izgled u skladu sa drugim Windows aplikacijama. U narednom tekstu će biti objašnjene metode i svojstva potrebne za rad sa svakom vrstom dijaloga ponaosob.

Pre svega, svojstvo zajedničko za svaki dijalog je CancelError. Pošto nemamo načina da saznamo da li je korisnik otkazao (Cancel) ili prihvatio (OK) dijalog, koristimo ovo svojstvo. Vrednosti koje može imati su True ili False. Ako je postavljeno na True, i ako korisnik otkaže dijalog, u vreme izvršavanja programa će se generisati greška. Sistem je dakle sledeći:

- pre otvaranja dijaloga postavljamo potrebna svojstva
- pozivamo dijalog
- ispitujemo da li je odmah posle linije poziva došlo do greške. Ako jeste znači da je korisnik otkazao dijalog (Cancel), u suprotnom korisnik je prihvatio dijalog (dugme OK na dijalogu) i u skladu sa ovim dalje vodim program

Još jedno zajedničko svojstvo dijaloga je Flags svojstvo. Predstavlja numeriču vrednost. Upisom odgovarajućih vrednosti u Flags svojstvo fino podešavamo ponašanje odgovarajućeg dijaloga. Postoji zaista mnogo različitih vrednosti za Flags svojstvo, a mi ćemo upoznati najčešće korišćene, zavisno od tipa dijaloga.

File Open dijalog

Dijalog koji služi za izbor fajla sa diska. Korisniku nudi standardni interfejs, mogućnost kreiranja direktorijuma, brisanje fajlova i ostale manipulacije. Po zatvaranju dijaloga korisnik je izabrao željeni fajl ili odustao od dijaloga.

Svojstva:

DialogTitle

Tekstualno svojstvo, definiše naslov koji će se ispisati na otvorenom dijalogu. Ako se ne navede ima inicijalnu vrednost Open.

FileName:

Tekstualno svojstvo, postavlja ili čita ime fajla koji je korisnik izabrao u dijalogu. Vraća korektno formiranu punu putanju do fajla (disk, direktorijum, ime fajla)

Filter:

Svojstvo koje omogućava postavljenje jednog ili više filtera koje korisnik može primeniti pri izboru fajlova. Svojstvo tipa tekst. Jedan filter se sastoji od dva dela: Opis i sam filter. Na primer:

CommonDialog1.Filter = "Tekst fajlovi|*.txt"

Ovaj filter će prikazati sve fajlove sa txt ekstenzijom, a u samom dijalogu će stajati opis "Tekst fajlovi". Što se samog filtera tiče mogu figurirati džoker znak (*) i / ili (?). Funkcija je ista kao u DOS-u. Ako želite da navedeti više filtera, jednostavno ih daljeređate vodeći računa da prvo postavite opis, a potom sam filter. Separator je "pipe" karakter. Na primer sledeća vrednost filter svojstva će nuditi tri filtera:

Opis	Filter
Tekst fajlovi	*.txt
Dokumenti	*.doc
Svi fajlovi	*.*

```
CommonDialog1.Filter = "Tekst fajlovi|*.txt|Dokumenti|*.doc|Svi fajlovi|*.*"
```

Flags:

U File Open dijalogu korisnik može ili izabrati postojeći fajl u listi fajlova, ili otkucati njegivu putanju i ime ako mu je poznata. U drugoj varijanti dijalog će vratiti ono što je korisnik otkucao, i ako taj direktorijum i / ili datoteka možda ne postoji. Da bi ovo sprečili Flags svojstvu treba dodeliti vrednost **CdlOFNFileMustExist.** Ako fajl ne postoji na disku, korisnik neće moći da prihvati dijalog.

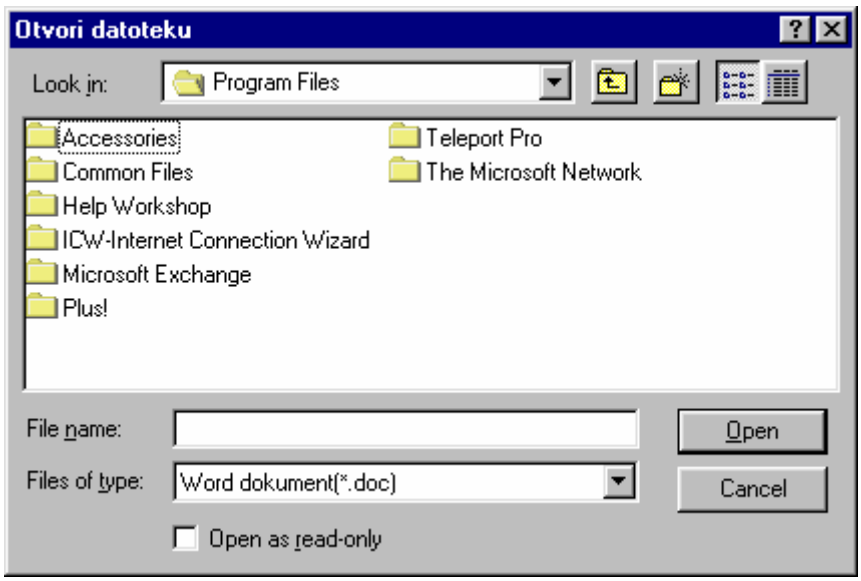
Metoda ShowOpen

Ova metoda otvara File Open dijalog.

Primer za File Open dijalog:

```
On Error Resume Next
With CommonDialog1
      .DialogTitle = "Otvori datoteku"
      .Flags = cdlOFNFileMustExist
      .Filter = "Word dokument(*.doc)|*.doc|Svi fajlovi(*.*)|*.*"
      .ShowOpen
End With
If Err.Number <> 0 Then
      MsgBox "Otkazan dijalog", vbInformation, "Open dijalog"
      Err.Clear
      Exit Sub
End If
Msgbox CommonDialog1.FileName, vbInformation, "Otvoren fajl"
```

Slika Open dijaloga:



File Save dijalog

Svojstva FileName, DialogTitle, Filter imaju isto značenje.

DefaultExt

Postavlja inicijalnu ekstenziju datoteke za snimanje pod uslovom da korisnik ne navede nikakvu ekstenziju. U suprotnom prihvata se ona koju određuje korisnik. Svojstvo je tekstualnog tipa.

Flags:

U Save / Save As dijalogu potrebno je upozoriti korisnika ako ime fajla pod kojim nešto snima već postoji na disku. Vrednost Flags svojstva koja ovo čini je **cdlOFNOverwritePrompt** . Ako fajl već postoji korisnik dobija upozorenje i pitanje da li želi da prebriše postojeći fajl. Inicijalno je **No.**

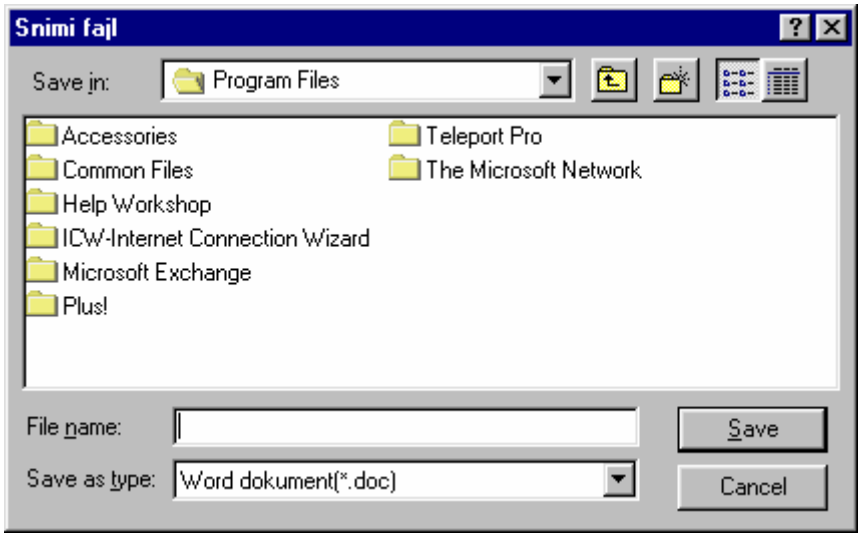
Metoda ShowSave

Otvara Save File dijalog.

Primer za File Save dijalog:

```
On Error Resume Next
With CommonDialog1
      .DialogTitle = "Snimi fajl"
      .Flags = cdlOFNOverwritePrompt Or cdlOFNHideReadOnly
      .Filter = "Word dokument(*.doc)|*.doc|Sve fajlovi(*.*)|*.*"
      .ShowSave
End With
If Err.Number <> 0 Then
      MsgBox "Otkazan dijalog", vbInformation, "Save dijalog"
      Err.Clear
      Exit Sub
End If
MsgBox CommonDialog1.FileName, vbInformation, "Save file"
```

Slika Save dijaloga:



Font Dijalog

Dijalog služi za izbor fonta i njegovih atributa. Svojstva su istoimena kao kod bilo koje druge kontrole koja ima tekst u sebi (TextBox, Label, ListBox,...):

Svojstva:

FontName	- ime sistemskog fonta (tekst)
FontSize	- veličina (integer)
FontBold	- boldiran (True/False)
FontItalic	- italic (True/False)
FontUnderline	- podvućen (True/False)
FontStrikeout	- precrtan (True/False)
Color	- boja fonta (long, RGB format)

Flags:

CdlCFForceFontExist	- Font koji korisnik ukuca mora postojati
CdlCFScreenFonts	- Samo ekranski fontovi u izboru
CdlCFPrinterFonts	- Samo printerski fontovi u izboru
CdlCFBoth	- Svi fontovi u izboru
CdlCFTTOnly	- Samo TrueType fontovi
CdlCFEffects	- Dodaje mogućnost za atribute Underline, StrikeOut i Color

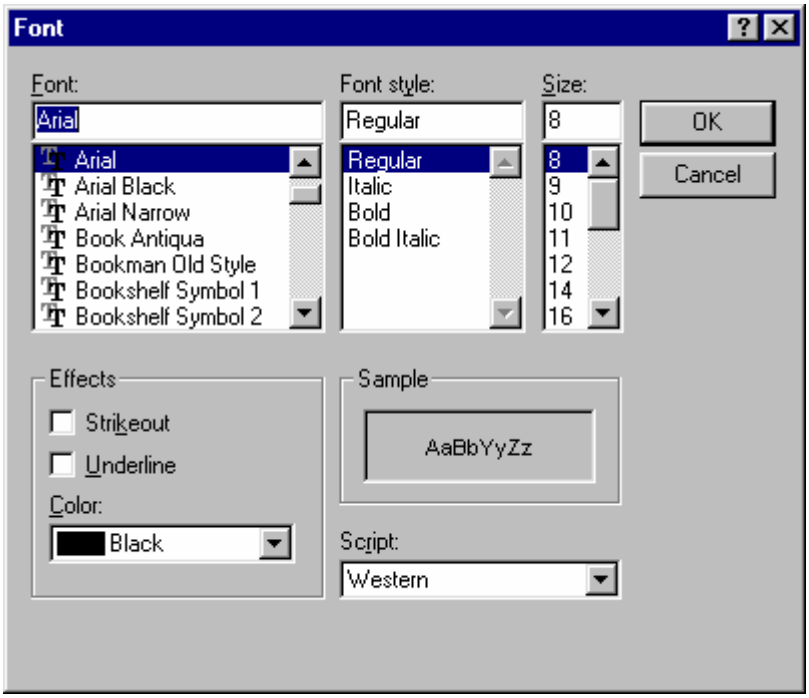
Kada kombinujemo više vrednosti za Flags svojstvo vezujemo ih pomoću ključne reči **OR**. Na primer želimo da vidimo samo ekranske TrueType fontove:
CommonDialog1.Flags = CdlCFScreenFonts Or CdlCFTTOnly

Ako želimo da izmenimo font na text box-u koji se zove tText kod je sledeći:

```
On Error Resume Next
With CommonDialog1
      .Flags = cdlCFForceFontExist _
            Or cdlCFScreenFonts _
            Or cdlCFTTOnly _
            Or cdlCFEffects
      ' Postavljamo svojstva dijaloga na ona koja su vec postavljena
      ' u text box-u, kako bi pri otvaranju dijaloga odgovarajuci font sa
```

```
        ' svojim atributima vec bio postavljen
        .FontName = tFont.FontName
        .FontSize = tFont.FontSize
        .FontBold = tFont.FontBold
        .FontItalic = tFont.FontItalic
        .FontUnderline = tFont.FontUnderline
        .FontStrikethru = tFont.FontStrikethru
        .Color = tFont.ForeColor
        .ShowFont
End With
If Err.Number Then
       Err.Clear
       Exit Sub
End If
With tFont
        .FontName = CommonDialog1.FontName
        .FontSize = CommonDialog1.FontSize
        .FontBold = CommonDialog1.FontBold
        .FontItalic = CommonDialog1.FontItalic
        .FontUnderline = CommonDialog1.FontUnderline
        .FontStrikethru = CommonDialog1.FontStrikethru
        .ForeColor = CommonDialog1.Color
End With
```

Slika Font dijaloga:



Color Dialog

Omogućava izbor sistemske boje i, ako je omogućeno, kreiranje korisnički definisanih boja.

Color svojstvo služi za postavljanje i dobijanje boje koju je izabrao korisnik. Da bi prilikom otvaranja dijaloga bila inicijalno selektovana boja koju želimo potrebno je postaviti Color svojstvo i Flags na **cdlCCRGBInit.**

ShowColor

Metoda otvara Color dijalog.

Primer postavljanja boje pozadine kontrole PictureBox1 pomoću Color dijaloga:

```
On Error Resume Next
With CommonDialog1
      .Flags = cdlCCRGBInit
      .Color = Picture1.BackColor
      .ShowColor
End With
If Err.Number Then
      Err.Clear
      Exit Sub
End If
Picture1.BackColor = CommonDialog1.Color
```

Slika Color dijaloga:

Print Dijalog

Print dijalog može prikazati Print Setup dijalog u kome se podešava default printer i njegova svojstva i standardni print dijalog u kome se podešava broj primeraka za štampu, štampanje od-do strane, štampanje selekcije i ostalo. Postavljanjem odgovarajuće vrednosti Flags svojstva određujemo koji od ova dva tipa se prikazuje.

ShowPrinter

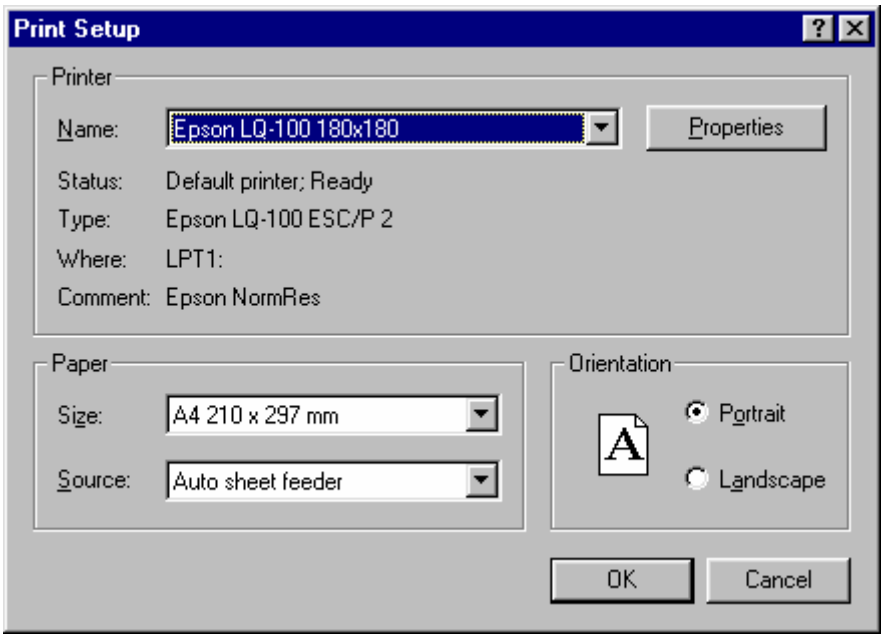
Metod koji otvara Printer dijalog

Primer otvaranja Print Setup dijaloga:

```
On Error Resume Next
With CommonDialog1
      .Flags = cdlPDPrintSetup Or cdlPDReturnDC
      .ShowPrinter
End With
If Err.Number <> 0 Then
      Err.Clear
      Exit Sub
End If
```

Posle ovoga postavljen je default printer u Windows-ima, i njegova svojstva (veličina papira rezolucija i ostalo zavisno od veznika instaliranog printera)

Slika Print Setup dijaloga



Print dijalog

je kompleksniji i poseduje sledeća svojstva:

Min

Minimalni broj strane od koje se može štampati

Max

Maksimalni broj strane do koje se može štampati

FromPage

(From Page) vrednost print djaloga

ToPage

(To Page) vrednost print dijaloga

Flags:

Svojstvo koje u Print djalogu najčešće čitamo, jer prikazuju koje vrednosti je korisnik izabrao:

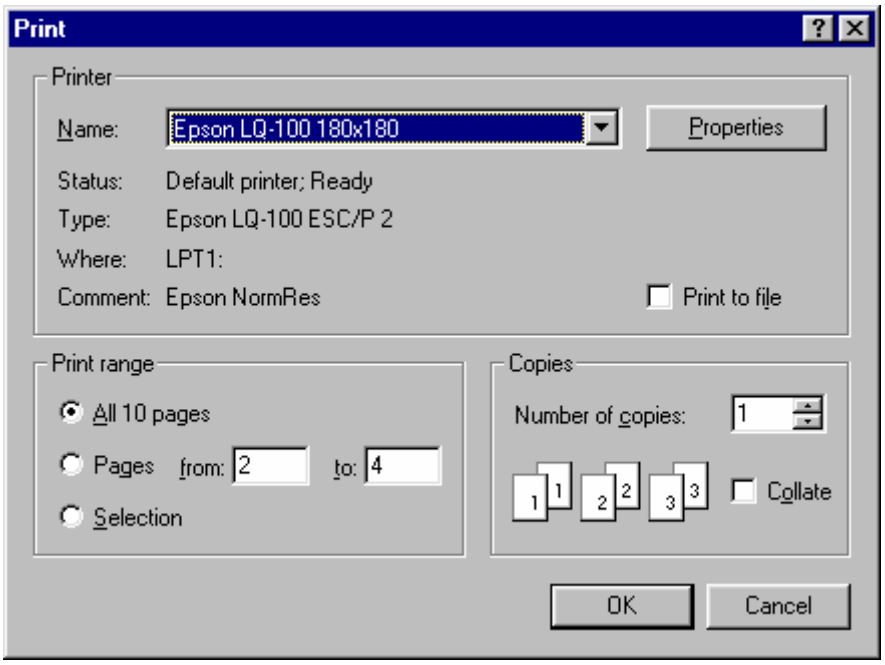
CdlPDAllPages	- izabrano štampanje svih strana (od Min do Max)
CdlPDPageNums	- izabrano štampanje grupe strana (od FromPage do ToPage)
CdlPDSelection	- izabrano štampanje selektovanog teksta

Primer otvaranja print dijaloga:

```
On Error Resume Next
With CommonDialog1
      .Min = 1
      .Max = 10
      .FromPage = 2
      .ToPage = 4
      .ShowPrinter
End With
If Err.Number <> 0 Then
```

```
      Err.Clear
      Exit Sub
End If

With CommonDialog1
      If .Flags = cdlPDAllPages Then
            MsgBox "Stampanje svih strana"
      ElseIf .Flags = cdlPDPageNums Then
            MsgBox "Stampanje strana od " & .FromPage & " do " & .ToPage
      ElseIf .Flags = cdlPDSelection Then
            MsgBox "Stampanje selekcije"
      End If
End With
```

Slika Print dijaloga:



MDI Forme

MDI (Multiple Document Interface) - interfejs za rad sa više dokumenata je standardni način na koji rade većina komercijalnih programa. U ovom sistemu postoji jedna roditeljska (Parent) forma koja služi kao kontejner za ostale forme (child-deca) koje se otvaraju unutar nje. Primer je MSWord aplikacija. Glavna (Parent) forma je glavni ekran Word-a, a svaki dokument koji kreirate se nalazi u svom prozoru unutar glavnog prozora. Treba primetiti da su sve akcije (meniji, toll bar-ovi) vezani za glavnu formu. Koristeći reference na aktivnu child formu akcije se preusmeravaju sa glavne - parent forme na child formu.

Roditeljska MDI forma je specijalna forma u Visual Basic projektu. Dodaje se sa menija Project - Add MDI Form. Dozvoljena je samo jedna MDI parent forma u projektu. S druge strane nema ograničenja u broju Child i standardnih formi.

MDI parent forma se razlikuje od standardne forme. Prva uočljiva razlika je u boji pozadine koja je sada tamno siva. Druga razlika je u tome da na MDI parent formu ne možete postaviti sve kontrole već samo one koje imaju align svojstvo (npr. Image i PictureBox). Izuzetak od ovoga je kontrola Timer i CommonDialog. Zbog ovoga ako želite da postavite na primer komandno dugme na MDI parent formu, prvo postavite Image kontrolu, pa potom na nju ostale. Uostalom tako se i formira toolbar.

MDI parent kontrola takođe ima metode i svojstva koje nema standardna forma:

ActiveForm

Svojstvo koje vraća referencu na aktivnu (onu koja ima fokus), MDI child formu otvorenu u okviru MDI parent forme. Na primer:

```
MDIForm1.ActiveForm.Caption = "Aktivna forma"
```

će postaviti naslov aktivne MDI child forme na "Aktivna forma".

Pošto je ActiveForm referenca na formu (slično kao Me ključna reč), daljnjim referenciranjem možemo postaviti svojstva i koristit metode svih kontrola koje se nalaze na MDI child formi. Na primer ako na MDI child formi postoji kontrola ListBox sa nazivom List1, validni su naredni izrazi:

```
MDIForm1.ActiveForm.List1.Additem "Nova stavka"
MDIForm1.ActiveForm.List1.FontName = "Arial"
...
```

Arrange

Mmetod: Pošto se u okviru MDI parent forme može nalaziti više otvorenih MDI child formi, pomoću arrange metode moguće je preurediti (aranžirati) njihov raspored. Arrange metod ima jedan ulazni parametar koji definiše način aranžiranja:

```
Arrange n
```

n:

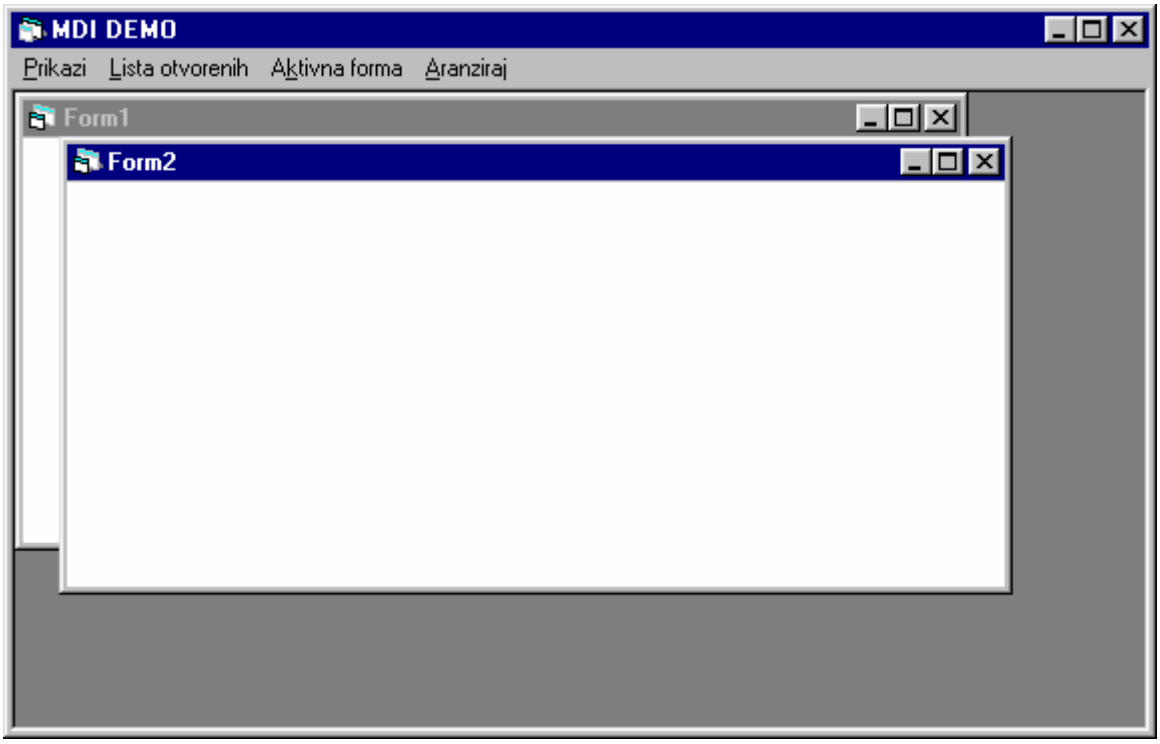
1 - Cascade	kaskadni raspored
2 - Tile horizontal	horizontalni
3 - Tile vertical	vertikalni
4 - Arrange Icons	aranžira ikone

MDI Child forme

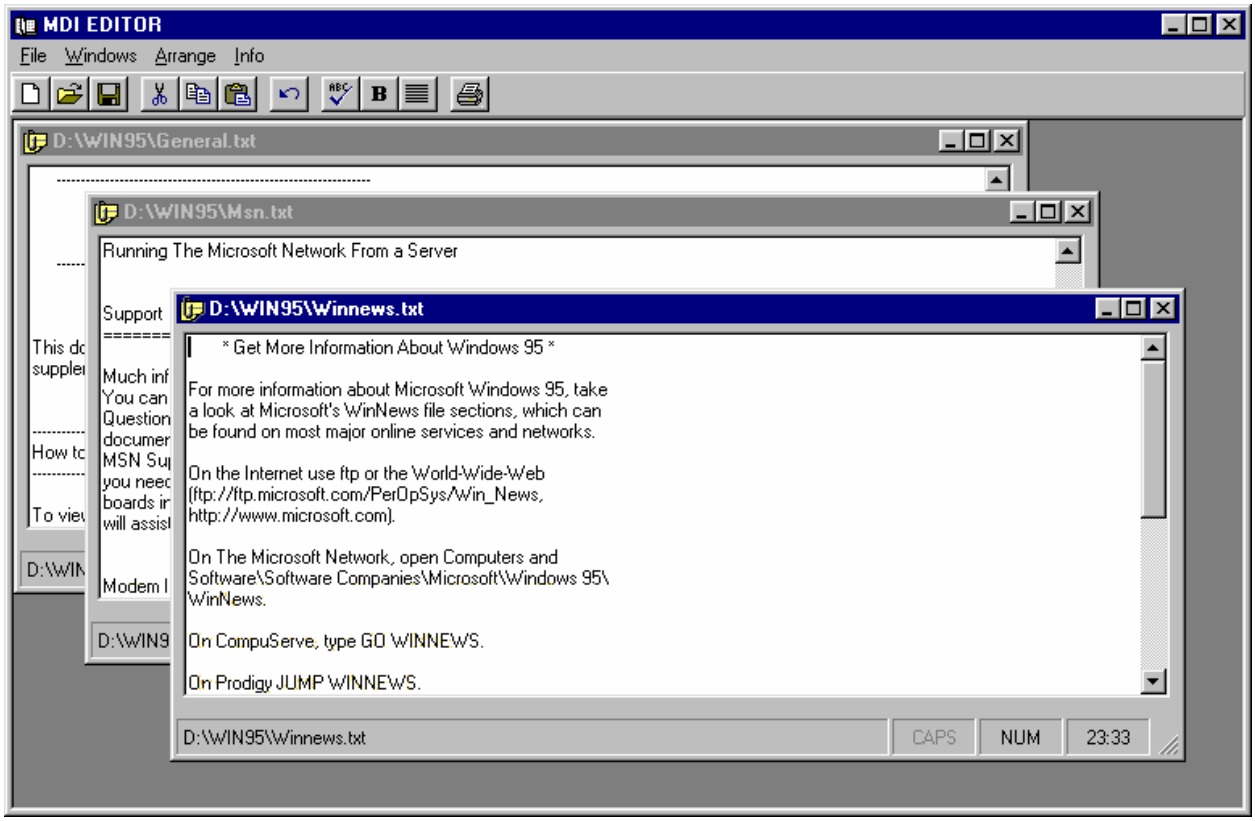
Standardna forma postaje MDI child forme ako se **MDIChild** svojstvo forme postavi na **True**. Ovo podrazumeva da MDI parent forma postoji u projektu. MDI child forma se po ponašanju razlikuje od standardne u dve stvari:

- kakve god dimenzije postavili u dizajn režimu, prilikom otvaranja u izvršnom režimu, ova forma će izmeniti dimenzije i to na oko 2/3 visine i 2/3 širine MDI parent forme. Ako želite da ovo prevaziđete potrebni kod za izmenu dimenzija se standardno stavlja u Form_Load i/ili u Form_Resize događaj.
- MDI child formu ne možete otvoriti u modalnom režimu. Ostale standardne forme u projektu naravno možete.

Slika: primer MDI parent forme sa otvorenim Child Formama



U narednom primeru ćemo demonstrirati jednostavan tekst editor po MDI receptu. Omogućeno je otvaranje tekstualnih ASCII dokumenata sa diska i njihov prikaz u više MDI child formi.
Gotova aplikacija:



Dakle, potrebna nam je jedna MDI forma i proizvoljni broj MDI Child formi u kojima će se raditi sa tekstom. Ovo poslednje nam postavlja pitanje koliko MDI Child formi treba da stavimo na raspolaganju

korisniku, tj. koliko maksimalno dokumenata može da obrađuje istovremeno. Odgovor na ovo pitanje je jednostavan: broj istovremeno otvorenih dokumenata mora biti ograničen jedino resursima ciljnog računara. Za rešenje ovog problema možemo iskoristiti dva pristupa. Prvi pristup – gruba sila: kreirati više potpuno istih MDI Child formi, koje imaju na sebi potpuno iste kontrole, osobine i obrađuju iste događaje. Ovaj pristup je apsolutno pogrešan ! Na ovaj način sebi pravimo dodatni posao, program je glomazniji i težak za realizaciju i održavanje. Definitivno preskočite ovaj pristup.

Ako imate u vidu da je u Visual Basic-u forma objekt, da je moguće u vremenu izvršavanja (run time), kreirati nove forme koje se baziraju na jednoj ranije kreiranoj i da sve te nove forme preuzimaju svojstva, procedure događaja i metode matične forme, to nas dovodi do pravog rešenja.

Kreiraćete samo jednu MDI Child formu. Postaviti na nju potrebne kontrole (u našem slučaju samo text box), postaviti svojstva same forme i kontrola na njoj i to će predstavljati šablonsku formu (template), na osnovu koje ćete u vremenu izvršavanja programa, po potrebi kreirati proizvoljni broj formi. U C++ terminologiji ovaj postupak bi se zvao nasleđivanje objekta, u našem slučaju šablon forme.

Takođe, sve procedure događaja (učitavanje teksta, snimanje teksta, izmena fonta itd...) ćete vezati za MDI formu. Pomoću svojstva ActiveForm je lako ove procedure preusmeriti na bilo koju aktivnu MDI Child formu.

Na ovaj način smo dobili kompaktnu, optimalnu aplikaciju koja je laka za realizaciju i održavanje. Sada naš zadatak izgleda ovako:

- Jedna MDI forma
- Jedna template MDI child forma

NAPOMENA: Uočite da se u primeru zapravo nigde ne otvara sama MDI Child forma, već samo njeni klonovi (instance).

Posao ćemo podeliti na nekoliko koraka.

Korak I: Kreiranje MDI Forme komponenti i koda za nju

Otvorite nov projekt u Visual Basicu (Standard EXE). Inicijalno postoji forma Form1 od koje ćemo kasnije napraviti template.

Dodajte MDI formu projektu (Project –> Add MDI form).

Da bi ste koristili Commondialog morate dodati ovu komponentu projektu. Kao što je ranije objašnjeno, to uradite pomoći menija Project -> Components (ili pritiskom na tastere Ctrl+T). Dobijate otvoren dijalog sa komponentama. Budite sigurni da CheckBox Selected Items Only nije čekiran. U listi nađite komponente Microsoft Common Dialog Control 5.0 označite CheckBox sa leve strane komponente. Na kraju pritisnite dugme OK.

Otvorite formu MDIForm1. Otvorite prozor sa svojstvima ove forme. Prvo izmenite name svojstvo u EDITOR, svojstvo caption u MDI EDITOR i na kraju svojstvo WindowState u 2 – Maximized.

Na MDI formu postavite Image kontrolu kako bi kreirali tollbar. On treba da ima 11 dugmića koji redom vrše sledeće funkcije:

- nov dokument
- otvori dokument
- snimi dokument
- cut (iseci)
- copy (kopiraj)
- paste (zalepi)
- undo (poništi izmene)
- izbor fonta
- izbor boje teksta
- izbor boje pozadine
- štampanje teksta

Postavite 11 komadnih dugmića na Image kontrolu. Sva trebaju da imaju postavljena svojstva Style na Graphical i Caption na prazan string. Potom opstavite sliku za svako komandno dugme (picture svojstvo). U primeru smo iskoristili bitmape koje se standardno dobijaju uz Visual Basic i nalaze se u direktorijumu \Graphics\Bitmaps\Tlbr_w95. Vi naravno možete iskoristiti druge slike, možete ih čak napraviti i sami uz pomoć nekog programa za kreiranje ikona ili bitmapiranih slika, ali vodite računa da

sve budu istih dimenzija (tipično 32x32 ili 16x16 piksela). U tabeli su date vrednosti svojstava Name i ToolTipText za svako komandno dugme.

Name	**ToolTipText**
bNew	Nov dokument
bOpen	Otvori dokument
bSnimi	Snimi dokument
bCut	Iseci
bCopy	Kopiraj
bPaste	Zalepi
bUndo	Ponisti izmene
bFont	Izbor fonta
bForeColor	Boja teksta
bBackColor	Boja pozadine
bPrint	Stampaj dokument

Takođe kreirajte meni na MDI parent formi koji ima sledeću strukturu.

Caption	**Name**	**ShortCut**	**Nivo menija**
&File	File		1
&Kraj rada	Izlaz	Ctrl+K	2
&Windows	Wndws		1
&Arrange	Arr		1
&Kaskadno	A (index 0)		2
&Vertikalno	A (index 1)		2
&Horizontalno	A (index 2)		2
&Ikone	A (index 3)		2
&Info	Info		1

U meniju &Windows (Wndws) obavezno čekirajte CheckBox WindowList !

NAPOMENA: Obratite pažnju da prilikom kreiranja menija ne dodeljujete nekoj stavki ShortCut jednak gore pomenutim. U nekim situacijama (editujete tekst u TextBox-u) Windows niju u stanju da odrediti da li, na primer sa Ctrl+X, želite da isečete označeni tekst ili da izvršite stavku iz menija čija je skraćenica (ShortCut) isto Ctrl+X. U tom slučaju se ne izvodi ni jedna od ove dve akcije, pa se dobija utisak da aplikacija ne radi ispravno.

KOD ZA MDI PARENT FORMU:

Nov dokument:

Private Sub bNew_Click()

```
    ' Procedura koja otvara novi dokument
    ' U slučaju greške idi na lebelu GRESKA
    On Error GoTo GRESKA
    ' Pokazivač miša na "wait"
    MouseOff
    ' Deklarise objektnu varijablu tipa FORM
    Dim NovaForma As Form

    ' Dodeljuje ovoj varijabli vrednost "šablon" forme Form1
    Set NovaForma = New Form1

    ' Postavlja svojstva novo kreirane forme
```

```
     With NovaForma
          ' Postavlja naziv forme
          .Caption = "Nov dokument"
          .StatusBar1.Panels(1).Text = "Bez imena"
          ' Prikazuje formu
          .Show
     End With

Izlaz:
     MouseOn
     Exit Sub

GRESKA:
     MsgBox "Ne mogu da kreiram nov dokument" & Chr$(13) _
     & Err.Description, vbCritical, "MDI EDITOR"
     Resume Izlaz
End Sub
```

Otvaranje dokumenta:

Private Sub bOpen_Click ()

```
     ' Procedura za otvaranje dokumenta

     ' Deklaracija objektne promenjive tipa FORM
     Dim A As Form
     ' U slučaju greške nastavi dalje
     On Error Resume Next

     ' Postavljamo svojstva Common Dialog kontrole
     With CommonDialog1
          ' Naslov dijaloga
          .DialogTitle = "Otvori dokument"

          ' Postavljanje filtera
          .Filter = "Tekst fajlovi (*.txt)|" & _
               "*.txt|Svi fajlovi|*.*|"
          ' Postavljanje Flags parametra kako bi korisnik
          ' morao da ukuca ime postojećeg fajla.
          ' U suprotnom Common Dialog kontrola prijavljuje
          ' grešku
          .Flags = cdlOFNFileMustExist
          .filename = ""
          ' Otvaramo Open dijalog
          .ShowOpen
     End With

     ' Ako je korisnik pritisnuo Cancel dugme na Open
     ' dijalogu, generiše se greška
     ' Proveravamo da li je ovo slučaj i ako jeste...
     If Err Then
          ' Brišemo grešku
          Err.Clear
          ' Izlazimo iz subrotine
          Exit Sub
     End If

     ' Ako je korisnik izabrao fajl sa diska
     ' U slučaju greške idi na labelu greška
     On Error GoTo GRESKA
     ' Dodeljuje ovoj varijabli vrednost "šablon" forme Form1
     Set A = New Form1
     ' Naslov forme na ime otvorenog fajla
     A.Caption = CommonDialog1.filename
```

```
     ' Isto i za status bar forme
     ' Otvaramo izabrani fajl za čitanje
     Open EDITOR.CommonDialog1.filename For Input As #1
     ' Učitavamo celokupni sadržaj fajla
     ' LOF(1) funkcija vraća dužinu fajla u bajtovima
     A.Text1 = Input(LOF(1), 1)
     ' Prikazujemo novokreiranu formu
     A.Show

Izlaz:
     ' Zatvaramo fajl
     Close
     ' Izlaz iz subrotine
     Exit Sub

GRESKA:
     ' Poruka o grešci sa njenim opisom
     MsgBox "Greska prilikom ucitavanja" & Chr$(13) & _
          Err.Description, vbCritical, "EDITOR"
     ' "Isčitavamo" formu
     Unload A
     ' Oslobađamo memoriju od objekta
     Set A = Nothing
     ' Nastavljamo od labele IZLAZ
     Resume Izlaz
End Sub
```

Snimanje teksta:

```
Private Sub bSnimi_Click ()
     ' Subrotina za snimanje aktivnog dokumenta
     ' U slučaju greške idi dalje
     On Error Resume Next
     ' Postavljamo svojstva Common Dialog kontrole
     With CommonDialog1
          ' U slučaju da fajl već postoji postavlja se
          ' pitanje da li želimo da ga prebrišemo
          .Flags = cdlOFNOverwritePrompt

          ' Naslov dijaloga
          .DialogTitle = "Snimi dokument"

          ' Postavljanje filtera
          .Filter = "Tekst fajlovi (*.txt)|*.txt|" & _
               "Svi fajlovi (*.*)|*.*|"

          ' Ukoliko korisnik ne upiše ekstenziju fajla
          ' inicijalna ekstenzija je txt
          .DefaultExt = "txt"
          .filename = ""
          ' Prikazujemo Save dijalog
          .ShowSave
     End With

     ' Ako je korisnik pritisnuo Cancel dugme
     If Err.Number Then
          ' Brišemo grešku
          Err.Clear
          ' Izlazimo iz subotine
          Exit Sub
     End If

     ' Ako je izabran fajl za snimanje ...
```

```
     ' U slučaju greške idi na labelu GRESKA
     On Error GoTo GRESKA
     ' Otvaramo fajl za čitanje
     Open EDITOR.CommonDialog1.filename For Output As #1

     ' Upisujemo sadržaj text kontrole na izabrani fajl
     Print #1, EDITOR.ActiveForm.Text1.Text

     ' Postavljamo naslov forme na
     ' naziv snimljenog dokumenta
     EDITOR.ActiveForm.Caption = _
          EDITOR.CommonDialog1.filename

     ' Zatvaramo fajl
     Close
     ' Informišemo korisnika o uspešnom snimanju
     MsgBox "Dokument snimljen pod imenom" & Chr$(13) & _
     EDITOR.CommonDialog1.filename, vbInformation, "EDITOR"

Izlaz:
     ' Izlazimo iz subrotine
     Exit Sub

GRESKA:
     ' Informišemo korisnika o grešci i dajemo opis greške
     MsgBox "Greska prilikom snimanja" & Chr$(13) & _
          Err.Description, vbCritical, "EDITOR"
     ' Dalje nastavljamo od labele IZLAZ
     Resume Izlaz
End Sub
```

Izmena fonta

```
Private Sub bFont_Click ()
     ' Procedura za izmenu atributa fonta
     ' U slučaju greške idi dalje
     On Error Resume Next

     ' Postavljamo svojstva Common Dialog kontrole
     ' pre otvaranja dijaloga za izbor fonta
     ' Želimo da se u Font dijalogu inicijalno postavimo na
     ' naziv fonta, veličinu i ostale atribute pri otvaranju
     With CommonDialog1
          ' Važno pre otvaranja Font dijaloga !!!
          ' cdlCFForceFontExist: naziv ukucanog fonta mora
          ' postojati u sistemu
          ' cdlCFBoth: Prikazuje instalirane i printerske i
          ' ekranske fontove
          ' cdlCFScalableOnly: Prikazuje samo skalabilarne,
          ' TrueType fontove instalirane u sistemu
          ' cdlCFWYSIWYG: Prikazuje samo fontove koji su
          ' zajednički i za ekran i za printer. Ovo je
          ' neophodno zbog kasnijeg štampanja dokumenta.
          ' U suprotnom može se dogoditi da korisnik izabere
          ' font koji vidi na ekranu, ali posle ne može da ga
          ' ispravno odštampa na štampaču.
          .Flags=cdlCFForceFontExist Or _
               cdlCFBoth Or cdlCFScalableOnly Or cdlCFWYSIWYG
          .FontName = EDITOR.ActiveForm.Text1.FontName
          .FontSize = EDITOR.ActiveForm.Text1.FontSize
          .FontItalic = EDITOR.ActiveForm.Text1.FontItalic
          .FontBold = EDITOR.ActiveForm.Text1.FontBold
```

```
          .ShowFont
     End With

     ' U slučaju da je korisnik pritisnuo Cancel dugme
     If Err.Number Then
          ' Brišemo grešku
          Err.Clear
          ' Izlazimo iz subrotine
          Exit Sub
     End If

     ' U suprotnom ...

     ' Postavljamo atribute fonta u dokumentu na one koje je
     ' korisnik selektovao u Font dijalogu
     With EDITOR.ActiveForm.Text1
          .FontName = EDITOR.CommonDialog1.FontName
          .FontSize = EDITOR.CommonDialog1.FontSize
          .FontItalic = EDITOR.CommonDialog1.FontItalic
          .FontBold = EDITOR.CommonDialog1.FontBold
     End With
End Sub
```

Boja teksta:

```
Private Sub bForeColor_Click()
     ' procedura za izmenu boje teksta dokumenta

     ' u slucaju greske idi dalje
     On Error Resume Next

     ' Pošto želimo da u otvorenom Color dijalogu aktuelna
     ' boja teksta dokumenta bude selektovana, to postižemo
     ' pomoću cdlCCRGBInit vrednosti Flags svojstva,
     CommonDialog1.Flags = cdlCCRGBInit
     ' i postavljanjem Color svojstva na aktuelnu vrednost.
     CommonDialog1.Color = EDITOR.ActiveForm.Text1.ForeColor

     ' Prikazujemo Color dijalog
     CommonDialog1.ShowColor
     ' Ako je korisnik pritisnuo Cancel taster
     If Err.Number Then
          ' Brišemo grešku
          Err.Clear
          ' Izlazimo iz subrotine
          Exit Sub
     End If

     ' Postavljanje boje teksta dokumenta na izabranu boju
     ' iz Color dijaloga
     EDITOR.ActiveForm.Text1.ForeColor = _
          EDITOR.CommonDialog1.Color
End Sub
```

Boja pozadine:

```
Private Sub bBackColor_Click()
     ' procedura za izmenu boje pozadine dokumenta
     On Error Resume Next
     CommonDialog1.Flags = cdlCCRGBInit
     CommonDialog1.Color = EDITOR.ActiveForm.Text1.BackColor
```

```
     CommonDialog1.ShowColor
     If Err.Number Then
          Err.Clear
          Exit Sub
     End If
     EDITOR.ActiveForm.Text1.BackColor = CommonDialog1.Color
End Sub
```

Štampanje dokumenta:

```
Private Sub bPrint_Click()
      ' Procedura za štampanje dokumenta

      ' U slučaju greške idi dalje
      On Error GoTo GRESKA

      ' Postavljamo kontrolno pitanje za potvrdu štampanja

      If MsgBox("Odstampati dokument", _
            vbQuestion + vbOKCancel, "EDITOR") = vbCancel _
            Then Exit Sub

      ' Postavljamo font atribute printer objekta
      With Printer
            .FontName = EDITOR.ActiveForm.Text1.FontName
            .FontSize = EDITOR.ActiveForm.Text1.FontSize
            .FontBold = EDITOR.ActiveForm.Text1.FontBold
            .FontItalic = EDITOR.ActiveForm.Text1.FontItalic
      End With

      ' Metodom Print printer objekta štampamo sadržaj
      ' aktivnog dokumenta
      Printer.Print EDITOR.ActiveForm.Text1.Text
      ' Obaveštavamo Windows da smo završili sa štampanjem
      Printer.EndDoc

Izlaz:
      Exit Sub

GRESKA:
      ' Poruka o grešsci
      MsgBox "Greska prilikom stampanja !" & Chr$(13) & _
            Err.Description, vbCritical, "EDITOR"
      ' Nastavljamo od labele IZLAZ
      Resume Izlaz
End Sub
```

Rad sa tekstom

```
Private Sub bCut_Click()
     SendKeys "^x"
End Sub

Private Sub bCopy_Click()
     SendKeys "^c"
End Sub

Private Sub bPaste_Click()
     SendKeys "^v"
End Sub

Private Sub bUndo_Click()
```

```
     SendKeys "^z"
End Sub
```

MDI CHILD FORMA

Na MDI Child formu koja u ovom slučaju predstavlja šablon forme potrebno jepostaviti samo text box. Ostavićemo inicijalno ime (name) **Text1**, i sledeća svojstva:
MultiLine - True
ScrollBars - Vertical

Pošto želimo da se prilikom izmena dimenzije forme, automatski vrši izmena dimanzija text box kontrole, odgovarajući kod pišemo za **Resize** događaj forme:

```
Private Sub Form_Resize()
     ' U slucaju greske nastavi dalje
     On Error Resume Next
     ' pozicija TextBox-a
     Text1.Top = 60
     Text1.Left = 60
     ' dimenzije TextBox-a
     Text1.Width = Me.Width - 270
     Text1.Height = Me.Height - 1000
End Sub
```

Naravno, svojstvo forme **MDIChild** treba postaviti na True.

MENI MDI Parent forme

```
Private Sub Izlaz_Click()
     ' Od korisnika trazi potvrdu zatvaranja programa.
     ' Ako korisnik izabere Yes dugme zavrsava se program
     If MsgBox("Izlaz iz programa", vbYesNo + vbQuestion, _
          "EDITOR") = vbYes Then End
End Sub

Private Sub A_Click (Index As Integer)
     ' Aranzira MDI Child prozore na način zavistan od
     ' vrednosti Index
     EDITOR.Arrange Index
End Sub
```

OLE automatizacija

OLE - Object Linking and Embeding (ugrađivanje i vezivanje objekata), je standard koji omogućava komunikaciju između aplikacija i međusobnu razmenu informacija. Ako na primer posmatramo Word dokument (doc), njega možemo, koristeći OLE, otvoriti i prikazati u VB formi koristeći OLE sposobnosti Word-a.

Postoje dve vrste OLE automatizacije:

1. **Ugrađivanje** (Embeding) - dokument je ugrađen u naš VB program. To znači da original ostaje nedirnut, sve što dalje radimo na dokumentu, se radi na kopiji koja je integrisana u VB program.
2. **Vezivanje** (Linking) - u naš VB program je ugrađen samo pointer (pokazivač) na izvorni dokument. Svaka izmena se vrši na originalu.

Ove funkcionalnosti u programu omogućava OLE kontrola koja je kontejner OLE objekata.

Metode OLE kontrole:

InsertObjDlg:
Otvara standardni dijalog za izbor postojećeg OLE izvornog dokumenta, ili kreiranje novog izborom OLE server aplikacije.

CreateEmbed: (*ImeFajla*):
Vrši ugrađivanje izvornog dokumenta u OLE kontrolu. Ime fajla određuje lokaciju , a njegova ekstenzija određuje koja server aplikacija obrađuje dati objekt.

CreateLink: (*ImeFajla*):
Isto kao CreateEmbed ali ne ugrađuje već vezuje izvorni dokument

Delete:
Briše ugrađeni ili vezani objekt u OLE kontroli. Original ostaje nedirnut, već se briše samo njegova kopija (embed) ili pokazivač na njega (link) u OLE kontroli

SaveToFile (*BrojFajla*):
Snima objekt iz OLE kontroli na BrojFajla pod kojim je isti otvoren pomoću *Open* naredbe. Snimanje se vrši u binarnom formatu i taj fajl ne može otvoriti server aplikacija.

ReadFromFile (*BrojFajla*):
Čita snimljeni binarni objekt i prikazuje ga u OLE kontroli

DoVerb (*Parametar*):
Šalje naredbu server aplikaciji. Najčešće su u upotrebi:

vbOleOpen
otvara server aplikaciju u odvojenom prozoru i učitava u nju OLE objekt koji je smešten u OLE kontrolu Visual Basic-a

vbOLEUIActivate
otvara server aplikaciju i integriše je u VB formu. Svi meniji server aplikacije se prenose na formu, a ako je na formi ranije kreiran meni, ostaju vidljive samo stavke kod kojih je svojstvo NegotiatePosition postavljano na različitu vrednost od (none)

Update:
ažurira izmene OLE izvornog dokumenta koje su urađene u OLE serveru i prikazuje ih u VB OLE kontroli

Close:
zatvara server aplikaciju

Svojstva OLE kontrole:

SizeMode:
određuje dimenzije OLE kontrole zavisno od njenog sadržaja

HostName:
Određuje ime OLE kontrole pod kojim će ona biti prijavljena server aplikaciji

Događaji:

Updated:
Izvršava se posle Update metode

Resize:
Izvršava se prilikom izmena dimenzija OLE kontrole

VISUAL BASIC 6.0

III KURS

Vladimir Tasić

Sadržaj:

KLASE

Razlozi za korišćenje klasa u aplikacijama su:
- Mogućnost višestrukog korišćenja

Jednom kreiranu komponentu mogu koristiti i drugi programeri. Pri tome alat Object Browser pomaže na taj način što daje informacije o svojstvima, metodama i događajima koje ta klasa ima.
- Jednostavnost

Jednostavnost se postiže skrivanjem složenosti programa od drugih korisnika. Ova osobina se zove enkapsulacija.
Class modul je vrsta Visual Basic modula. Moguće je imati više class modula u jednoj aplikaciji

Kreiranje Class modula

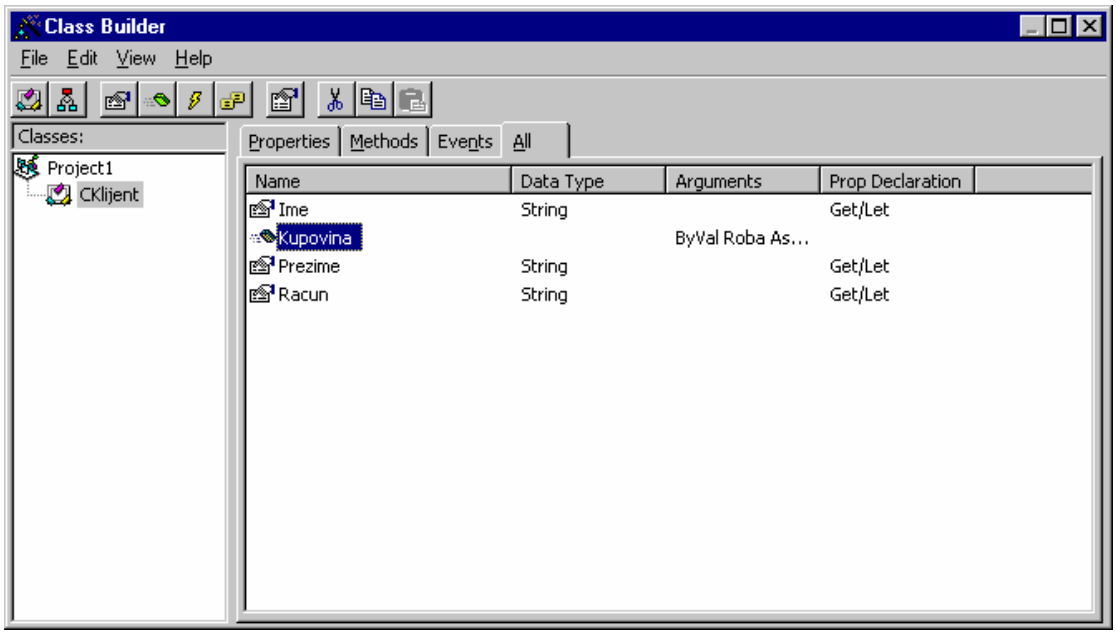
Moguće je izvesti ručno ili pomoću Class builder Add-In-a.
Class builder automatizuje dodavanje svojstava, metoda i događaja klasi.
Uključivanje Class Builder-a u projekat:

1. Na Add-Ins meniju izabrati Add-In Manager
2. Izabrati VB 6 Class Builder Utility
3. Podesiti Load Behavioar na Loaded

Za ručno dodavanje na Project meniju izabrati Class module, ali onda se ne mogu korisiti sve mogućnosti Class Builder-a.
Za dodavanje novog Class modula:

1. Na Add-Ins meniju izabrati Builder Utility

2. U Class Builder-u na File meniju izabrati New i kliknuti na Class
3. U Class Module Builder dialog box-u uneti ime klase
4. U Attributes tabu uneti opis klase i Help context ID ukoliko postoji Help file.



Kreiranje Class Interface-a

Pošto je klasa kreirana, potrebno je definisati svojstva, metode i događaje, koji se zajedno zovu interefejs klase.
Class modul ima dve ugrađena događaja: Initalize i Terminate.

Initialize događaj – dešava se onda kada se klasa kreira, ali pre nego što se svojstva postave i služi za inicijalizovanje bilo kog podatka koji klasa koristi ili za učitavanje formi koje klasa koristi.
Terminate događaj – dešava se kada objektna promenljiva izadje van opsega u kome je definisana ili kada se objektna promenljiva postavi na vrednost Nothing. Ovaj događaj se koristi za snimanje informacija, unload-ovanje formi ili za izvođenje zadataka koji se izvode kada se klasa gasi.

Svojstva

Svojstva definišu podatke ili atribute klase. Moguće je definisati svojstvo za klasu na dva načina:
- Preko promenljivih tipa public
- Preko Property procedura koje se izvršavaju kada se svojstva postavljaju ili iščitavaju; ovaj nacin omogućuje da se izvrši neki kod prilikom promene ili čitanja svojstva klase, a moguce je i izložiti svojstva koja su read-only. Property procedure je moguće kreirati u parovima: Let (ili Set) procedure za dodeljivanje vrednosti svojstvu i Get za vraćanje vrednosti iz procedure.

Metodi

Metodi klase predstavljaju funkcionalnost koju klasa obezbeđuje korisniku. Metod za objekat se kreira pomoću Public Sub ili Function procedura unutar Class modula.
Sledeći kod kreira metod koji povećava plate zaposlenih:

```
Public Function PovecanjePlate(Procenat As Double) As Integer
      mPlata = mPlata * (1 + Procenat)
End Function
```

Događaji

Događaji omogućuju da se klasi pruži obaveštenje o akciji koja se desila. Visual Basic ima dva ugrađena događaja – Initialize event i Terminate event. Moguće je deklarisati custom events za klasu.

Kreiranje instance klase

Class moduli služe kao šabloni za objekte. Da bi mogli da koristimo class moduli mora se prvo kreirati instanca class modula, a onda mogu pristupiti svojstvima, metodama i događajima te instance.
Class moduli i standardni moduli se razlikuju u dvema stvarima:
- Mora se eksplicitno napraviti instanca klase da bi se mogla koristiti
- Moguće je napraviti više instanci class modula.

U projektu koji sadrži class modul, instanca se može kreireti na dva načina:

Koristeći odvojene naredbe Dim i Set:

```
Dim zapCurrent As CZaposleni
Set zapCurrent = New CZaposleni
```

Ili koristeći kompaktniju sintaksu:

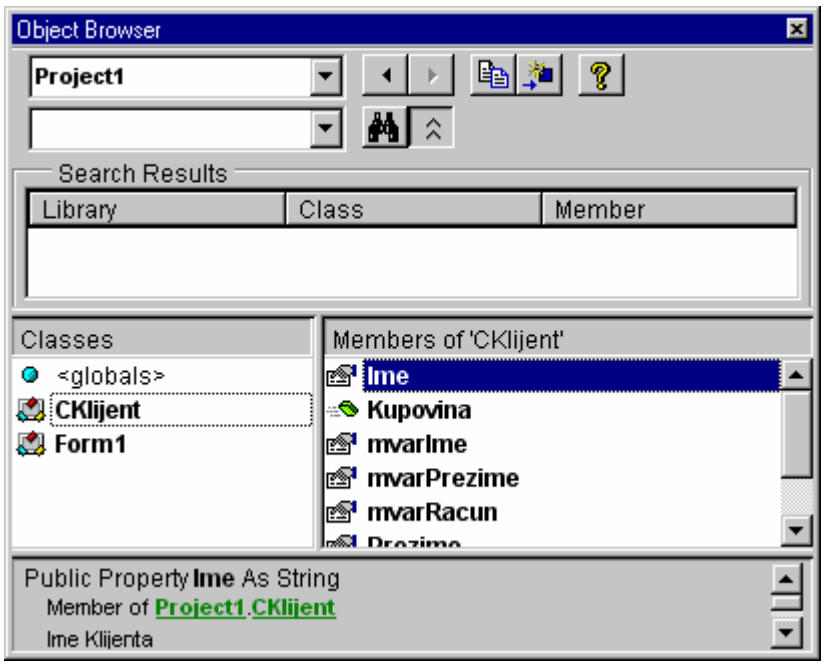
```
Dim zapCurrent As New CZaposleni
```

Prilikom rada na aplikacijama koje sadrže objekte preporučljivo je osloboditi memoriju kada korišćene instance završe svoju funkciju zbog kojih su kreirane. To se radi pomocu naredbi Set i Nothing.

```
Set zapCurrent = Nothing
```

Korišćenje Klasa

Pošto se napravi instanca class modula, moguće je testirati metode i svojstva klase.
Može se koristiti i Object Browser za pregled svojstava, metoda i događaja koji su definisani za klasu. Prozor Objekt Browser se otvara pritiskom na F2 funkcijski taster.



Sledeći kod pravi instancu klase CZaposleni i postavlja i ocitava Ime svojstvo

```
Dim zapCurrent As CZaposleni
Set zapCurrent = New CZaposleni
zapCurrent.Ime = "Jovan"    'Poziva Let proceduru.
MsgBox zapCurrent.Ime      'Poziva Get proceduru.
```

Primer: rešavanje kvadratna jednačine, u kome se koristi klasa KvadJednacina. Sledeći kod uneti u class modul Standard Exe projekta. Name svojstvo Class modula podesiti na KvadJednacina.

```
Option Explicit

Private PdblA As Double
Private PdblB As Double
Private PdblC As Double
Private PdblD1 As Double
Private PdblD2 As Double
Private PNemaR As Boolean
Private PTekst As String
Private D As Double

Private Sub Class_Initialize()
    MsgBox "Klasa inicijalizovana", vbInformation
    PdblA = 0
    PdblB = 0
    PdblC = 0
End Sub

Private Sub Class_Terminate()
    MsgBox "Klasa zavrsena", vbInformation
End Sub

Public Property Get A() As Double
    A = PdblA
End Property

Public Property Let A(locA As Double)
```

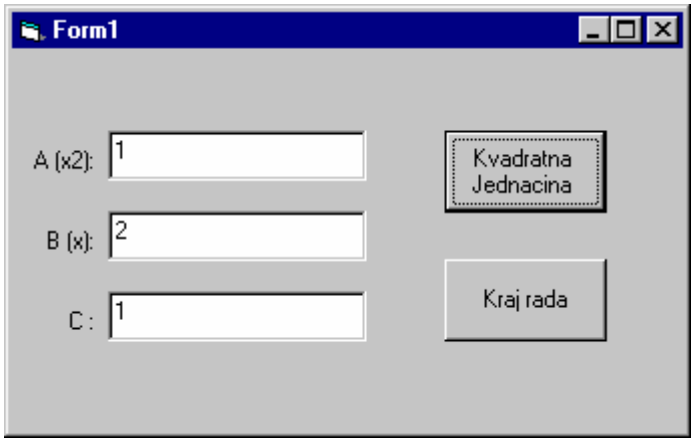
```
    PdblA = locA
End Property

Public Property Get B() As Double
    B = PdblB
End Property

Public Property Let B(locB As Double)
    PdblB = locB
End Property

Public Property Let C(locC As Double)
    PdblC = locC
End Property

Public Property Get C() As Double
    C = PdblC
End Property
```

```
Public Sub Izracunaj()
    If PdblA = 0 And PdblB = 0 And PdblC = 0 Then
        PdblD1 = 0
        PdblD2 = 0
        Exit Sub
    End If

    D = PdblB ^ 2 - 4 * PdblA * PdblC
    If D < 0 Then
        PdblD1 = 0
        PdblD2 = 0
        PNemaR = True
        PTekst = "Nema resenja"
        Exit Sub
    Else
        PNemaR = False
    End If

    D = Sqr(D)
    PdblD1 = (-PdblB + D) / (2 * PdblA)
    PdblD2 = (-PdblB - D) / (2 * PdblA)
    PTekst = PdblA & "x^2 + " & PdblB & "x +" & PdblC & "=0: " _
        & Chr$(13) & "x1:" & PdblD1 & "  x2:" & PdblD2
End Sub

Public Property Get NemaResenja()
    NemaResenja = PNemaR
End Property

Public Property Get Resenje1() As Double
    Resenje1 = PdblD1
End Property
```

```
Public Property Get Resenje2() As Double
    Resenje2 = PdblD2
End Property

Public Property Get TekstResenja()
    TekstResenja = PTekst
End Property
```

Formi koja se nalazi u aplikaciji dodati sledeći kod:

```
Private Sub cmdRacun_Click()
    Dim X As New KvadJednacina
    X.A = CDbl(txtA)
    X.B = CDbl(txtB)
    X.C = CDbl(txtC)
    X.Izracunaj
    If X.NemaResenja Then
        MsgBox "Jednacina nema resenje", vbExclamation
    Else
        MsgBox "x1=" & X.Resenje1 & Chr$(13) & _
            "x2=" & X.Resenje2, vbInformation, "Resenja"
    End If

    MsgBox X.TekstResenja, vbInformation, "Tekst resenja"
End Sub
```



COM KOMPONENTE

COM je standard koji omogućuje proširenje funkcionalnosti i međukooperativnosti komponenti koje ga podržavaju. COM nudi mehanizme koji omogućavaju da se komponente različitih proizvođaca softvera povezuju i komuniciraju među sobom na unapred definisani način.
Ideja je da treba uočiti i gde god je moguće koristiti vec postojeće COM komponente, čime se izbegava ponovno pisanje koda koji ne donosi nikakvu novu funkcionalnost.
Da bi koristili eksterne COM kompoinente u Visual Basic aplikaciji potrebno je učiniti da ta komponenta bude dostupna našoj aplikaciji, deklarisati objektnu promenljivu i u nju smestiti instancu klasu koju koristim, napraviti objekat i na kraju koristiti taj objekat za razvoj Visual Basic aplikacije.
Da bi komponenta bila dostupna klijent aplikaciji (aplikaciji koja koristi tu komponentu) ta komponenta mora biti dostupna i aplikacija mora biti upoznata da se koristi baš ta određena COM komponenta.
Prvi problem se rešava registrovanjem COM komponente koje se odvija pri postupku instalacije. Ukoliko postoje neki problemi, rešenje se može naći editovanjem registra, odnosno dodavanjem informacija potrebnih da bi se koristila ta komponenta u registry bazu podataka.
Drugi problem se rešava jednostavim izborom Project menija i References dialog box-a. Ukoliko je željena COM komponenta ispravno registrovana, naći ce se na spisku dostupnih stavki koje je moguće priključiti projektu klijent aplikacije u kojoj želimo da koristimo tu komponentu.

Korišćenje COM komponenti

Prvo se mora deklarisati objektna promenljiva sa referencom na objekat koji će se koristiti. Pošto je uspostavljena referenca na biblioteku objekata, Visual Basic detektuje objektnu promenljivu za vreme pisanja koda za tu aplikaciju. Visual Basic može da prikaže informacije o dostupnim metodama i svojstvima, kao i sintaksu za pozivanje metoda i svojstava. Moguće je

koristiti Object Browser za pregled informacija o metodama, svojstvima i događajima objekata.
Postoje slučajevi kad u vreme pisanja koda nije poznat određeni tip objektne promenljive koju aplikacija treba da koristi. Tada treba koristiti generičku objektnu promenljivu koja bi trebalo da predstavlja pokazivač svaku moguću vrstu objekata. To se postiže deklarisanjem oblika:

Dim objGeneric As **Object**

U tom slučaju objekat koji će se koristiti nije poznat sve do startovanja aplikacije, pa je logično da u tom slučaju nije moguće postaviti referencu na biblioteku objekata u Visual Basicu.
Pre nego što klijent aplikacija može da koristi metode, svojstva ili događaje, klijent aplikacija mora biti vezana (bound) za objekat. Vrsta promenljive koja je izabrana će odredtiti način povezivanja između klijent aplikacije i objekta.
Ako se radi sa unapred određenom (specific) objektnom promenljivom koristi se rano (early) binding, za koje je karakteristično povezivanje za vreme pisanja koda aplikacije (at design time) i u tom slučaju Visual Basic proverava sintaksu poziva objektnih promenljivih, a kompajler je u stanju da napravi efikasniji kod za pristup objektima u vreme izvršenja aplikacije.
Ako se radi sa objektnom promenljivom čiji se tip ne može unapred odrediti (generic) koristi se kasno (late) povezivanje koje se ostvaruje u vreme izvršenja aplikacije (at run time) i u tom slučaju nema nikakvih informacija o objektu u vreme pisanja aplikacije, a potrebno je dodatno angažovanje Visual Basica za pristup objektu u vreme izvršenja aplikacije, što ima loš uticaj na performanse klijent aplikacije.
Iz svega gore navedenog, treba koristiti generic promenljive samo kad je baš apsolutno neophodno.

Kreiranje objekata za komponente se u Visual Basicu može izvesti na tri nacina:

1. Kombinacijom New i Set naredbi - korisiti se pri ranom povezivanju.

2. Koristeći GetObject funkciju - priliko kreiranja instance objekta koji je snimljen u fajl. Kod koji sledi je primer koji pravi instancu Word dokumenta koji se zove Ugovor.doc i prikazuje je u Print Preview modu:

```
Sub PreviewUgovor()
     Static wdUgovor As Word.Document
     Set wdUgovor = GetObject("C:\Ugovor.doc", "Word.Document")
     wdUgovor.Parent.Visible = True
     wdUgovor.PrintPreview
End Sub
```

3. Koristeći CreateObject funkciju - je metod koji se primenjuje ako je neophodna generic promenljiva, odnosno kasno povezivanje. Sintaksa za CreateObject funkciju je:
CreateObject(*class*, [*servername*])
     class - ime aplikacije i klasa objekta koji treba napraviti (appname.objecttype)
       servername - ime mrežnog servera gde ce objekat biti napravljen.

Metode InternetExplorer objekta:

Navigate	- Koristeći URL
Quit	- Zatvara Internet Explorer aplikaciju
Refresh	- Ponovo učitava trenutnu stranicu
Stop	- Otkazuje sve navigacione ili download operacije

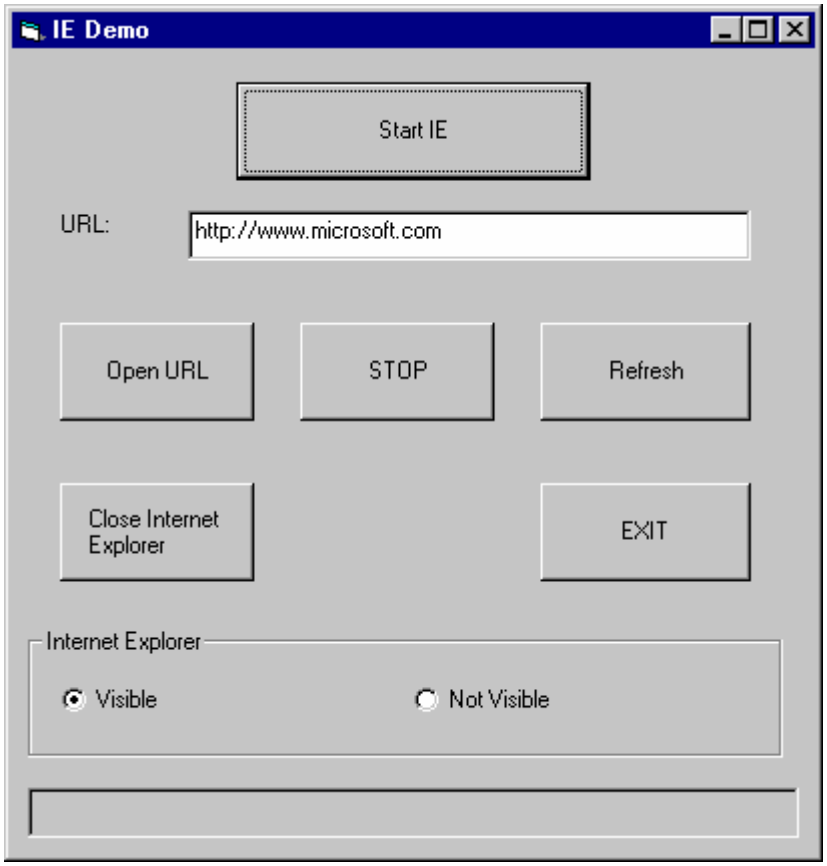
Događaji InternetExplorer objekta:

BeforeNavigate2
DownloadBegin
DownloadComplete
NavigateComplete2

Primer: Korišćenje MS Internet Explorera
Postaviti referencu na objekat InternetExplorer preko References opcije Project menija i izbora Microsoft Internet Explorer reference.

```
Option Explicit

Dim WithEvents ie As InternetExplorer

Private Sub cmdCloseIE_Click()
    ie.Quit
    Set ie = Nothing
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub

Private Sub cmdOpenPage_Click()
    ie.Navigate txtURL.Text
End Sub

Private Sub cmdRefresh_Click()
    ie.Refresh
End Sub

Private Sub cmdStartIE_Click()
    Set ie = New InternetExplorer
    ie.ToolBar = False
    ie.StatusBar = False
    ie.Width = 440
    ie.Height = 400
    ie.Top = 105
    ie.Left = 270
    ie.Visible = optVisible.Value
End Sub

Private Sub cmdStop_Click()
    ie.Stop
End Sub


Private Sub ie_BeforeNavigate2(ByVal pDisp As Object, URL As Variant, Flags
As Variant, TargetFrameName As Variant, PostData As Variant, Headers As
Variant, Cancel As Boolean)
    lblStatus.Caption = "Pocinje navigacija..."
End Sub

Private Sub ie_DownloadBegin()
    lblStatus.Caption = "Download pocinje..."
End Sub

Private Sub ie_DownloadComplete()
```

```
    lblStatus.Caption = "Download kompletan..."
End Sub

Private Sub ie_NavigateComplete2(ByVal pDisp As Object, URL As Variant)
    lblStatus.Caption = "Navigacija kompletirana..."
End Sub

Private Sub optNotVisible_Click()
    ie.Visible = False
End Sub


Private Sub optVisible_Click()
    ie.Visible = True
End Sub
```



ACTIVEX CONTROLE

Kontrole su objekti koji sadrže vizuelne elemente i kod i mogu se koristiti više puta. Visual Basic se isporučuje sa ugrađenim kontrolama koje se vide u Control Toolbox-u. Moguće je dodati ActiveX kontrole u Toolbox. Počevši od Visual Basic verzije 5.0 moguće je kreirati sopstvene ActiveX kontrole. Kontrole moraju biti stavljene u neki od kontejnera, kao što su forme ili aplikacije.
Kontrola kreirana u Visual Basicu se drukčije zove class control i predstavlja šablon za datu kontrolu. Control class se kompajlira u .ocx fajl. Za korišćenje kontrole u aplikaciji, treba jednostavno staviti kontrolu na formu i na taj način kreirati design-time instancu te kontrole. Kada korisnik startuje aplikaciju koja sadrži kontrolu, on dobija run-time instancu kontrole. Izvorni (source) kod i vrednosti svojstava za control class se smeštaju u tekst fajl sa .ctl ekstenzijom, dakle .ctl fajl je ekvivalent .frm fajlu koji se koristi za smestanje formi u Visual Basicu. Grafički elementi, koji se ne mogu snimiti kao tekst se nalaze u fajlu sa .ctx ekstenzijom, dakle .ctx fajl je ekvivalent .frx fajlu koji sadrži grafičke elementa u formama.

Za kreiranje ActiveX kontrola, treba uraditi sledeće:
1. Započeti ActiveX Control projekat.
2. Napraviti korisnički interfejs za kontrolu. ActiveX kontrole se prave od postojećih kontrola – onih koje se isporučuju uz Visual Basicu ili onih koji se mogu nabaviti odvojeno (od nezavisnih proizvođača).
3. Napraviti svojstva i metode za kontrolu.
4. Napraviti procedure događaja za postojeće kontrole članice.
5. Izložiti događaje za kontrolu
6. Napraviti property pages za kontrolu.
7. Debagovati i testrati kontrolu.

UserControl objekat

UserControl objekta je osnova za građenje kontrola. Svaka ActiveX kontrola kreirana u Visual Basicu sadrži UserControl objekat. UserControl objekti sadrže module za kod i visual designer window. Visual designer se koristi za smeštanje kontrola članica na UserControl objekat na isti način kao što se to radi sa formama u Visual Basicu.
Kontrola članica je instanca kontrole koja se smešta na UserControl objekat. Prilikom smeštanja ActiveX kontrole na formu, kreira se instanca UserControl objekta, zajedno sa instancama svake kontrole članice koja se stavi na UserControl objekat. Moguće je koristiti bilo koju standardnu kontrolu osim OLE kontejner kontrole.

ActiveX Control Interface Wizard

Ova mogućnost Visual Basic-a pojednostavljuje zadatak pravljenja kontrola. Wizard pomaže pri realizovanju svojstava, metoda i događaja. ActiveX Control Interface Wizard mapira funkcionalnost kontrole na funkcionalnost UserControl objekta ili kontrola članica. Postoje default mapiranja koja se mogu kasnije izmeniti tokom procesa.
Wizard takođe generiše odgovarajući kod za interfejs, uključujući:

- Kod Property procedure, ya implementiranje svojstava,
- Sub i Function procedure za implementiranje metoda,
- kod za pokretanje izabranih dogadjaja.

Primer: RGB ActiveX

```
Option Explicit
Event IzmenaBoje()

Private Sub sBlue_Change()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sBlue_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sGreen_Change()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sGreen_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sRed_Change()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sRed_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub UserControl_Resize()
      sRed.Width = UserControl.Width
      sGreen.Width = UserControl.Width
      sBlue.Width = UserControl.Width
      UserControl.Height = 255 * 3
End Sub
```

```
Public Property Get Red() As Integer
    Red = sRed.Value
End Property
```

```
Public Property Let Red(ByVal New_Red As Integer)
    If New_Red < 0 Or New_Red > 255 Then
        MsgBox "Invalid red", vbCritical, "Red"
        Exit Property
    End If
    sRed.Value() = New_Red
    PropertyChanged "Red"
End Property
```

```
Public Property Get Green() As Integer
      Green = sGreen.Value
End Property
```

```
Public Property Let Green(ByVal New_Green As Integer)
      If New_Green < 0 Or New_Green > 255 Then
            MsgBox "Invalid green", vbCritical, "Green"
            Exit Property
      End If
      sGreen.Value() = New_Green
      PropertyChanged "Green"
End Property
```

```
Public Property Get Blue() As Integer
      Blue = sBlue.Value
End Property
```

```
Public Property Let Blue(ByVal New_Blue As Integer)
      If New_Blue < 0 Or New_Blue > 255 Then
            MsgBox "Invalid blue", vbCritical, "Blue"
            Exit Property
      End If
      sBlue.Value() = New_Blue
      PropertyChanged "Blue"
End Property
```

```
' Read property values
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
      sRed.Value = PropBag.ReadProperty("Red", 0)
      sGreen.Value = PropBag.ReadProperty("Green", 0)
      sBlue.Value = PropBag.ReadProperty("Blue", 0)
End Sub
```

```
'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
      Call PropBag.WriteProperty("Red", sRed.Value, 0)
      Call PropBag.WriteProperty("Green", sGreen.Value, 0)
      Call PropBag.WriteProperty("Blue", sBlue.Value, 0)
End Sub
```

```
Public Sub PostaviBoju()
      UserControl.Parent.BackColor = RGB(Red, Green, Blue)
End Sub
```

COM DLL

Pomoću Visual Basica moguće je kreirati COM komponente u obliku izvršnih fajlova (EXE) ili DLL-ova. Međutim da bi komponente mogle koristiti prednosti MTS-a (Microsoft Transaction Server) COM komponente moraju biti napravljene kao DLL.
COM DLL je takozvana in-process COM komponenta, koja obezbeđuje brzi pristup objektima, ali je manje otporna na greške, odnosno ako DLL padne, pada ceo proces.
COM EXE je takozvana out-of-process COM komponenta, koja je tolerantnija na greške, odnosno ako EXE padne, ostali procesi u sistemu će nastaviti da se izvršavaju, ali je sporija od COM DLL-a.
Jedna COM komponente može da sadrži više class modula. Klasa je šablon koji definise metode i svojstva za objekat. Class moduli u Visual Basicu sadrže kod pomoću koga se implementiraju metode za tu klasu.
Pri kreiranju novog DLL projekta u Visual Basicu, Visual Basic pravi projekat sa jednim class modulom. Moguće je dodati novi class modul izborom Add Class Module opcije sa Project menija.

Obrada Grešaka

Prilikom pravljenja COM komponenti, koje će se koristiti sa MTS-om, veoma je bitno obratiti pažnju na obradu grešaka. U Visual Basicu procedura prenosi neobrađene greške proceduri koja ju je pozvala. Ako se greška prosledi skroz do pozivajuće procedure na najvišem hijerarhijskom nivou, izvršenje programa se prekida. Veoma je važno znati zašto se desila greška pri radu sa MTS-om. Komponenta mora izvestiti MTS da li je obavila posao uspešno. Praćenjem grešaka može se obavestiti MTS u kom se trenutno statusu izvršenja posla komponenta nalazi.
Visual Basic koristi interni Err objekat za smeštanje informacija o greškama koje su se desile. Da bi prosledio grešku nazad klijent aplikaciji (aplikaciji koja koristi COM komponentu) dovoljno je pozvati Raise metod.
Sintaksa Raise metoda:
Err.Raise (broj, izvor, opis, HelpFile, HelpContext).

Registrovanje COM DLL-a

Pre korišćenja COM DLL mora biti registrovan. Klijenti koriste sadržaj regisry-ja da bi locirali, napravili i koristili klase u COM DLL-u.

Postoji više načina za registrovanje COM DLL-a:

- Preko Setup programa, koji kad se startuje, registruje komponentu.

- Kompajliranjem DLL u Visual Basicu, čime se automatski registruje komponenta na računaru na kome je kompajlirana.

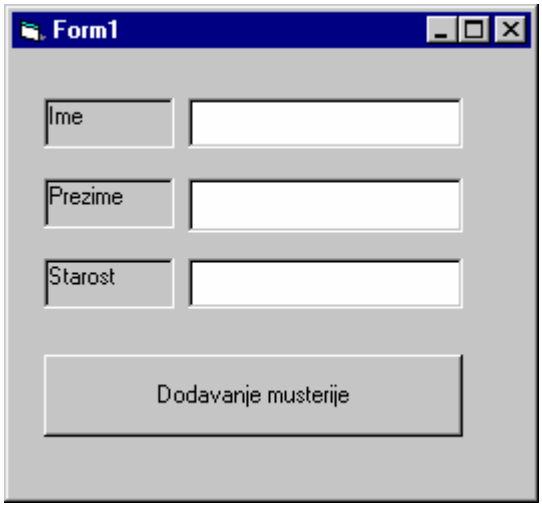
- Pomoću Regsvr32.exe alata.
Kada nema više potrebe za određenom komponentom treba je deregistrovati. Zavisno od Setup programa, neki DLL-ovi se mogu deregistrovati preko Control Panela koristeći Add/Remove Programs ikonu u Control Panelu.
Za ručno uklanjanje DLL-a iz registry-ja, treba startovati Regsvr32.exe sa /u opcijom.
Primer: Dll People, koji u sebi sadrži klasu Customer. I korišćenje napravljenog .dll-a u aplikaciji:

```
Option Explicit
Const mstrDataFilename = "customer.txt"
```

```
Public Sub AddCustomer(ByVal strFirst As String, ByVal strLast As String,
ByVal intAge As Integer)
    On Error GoTo ErrorHandler
    Open mstrDataFilename For Append Lock Write As #1
    Write #1, strFirst, strLast, intAge
    Close #1
    Exit Sub
ErrorHandler:
    'Uklanja guzvu koju je napravio dll
    Close
    'Izvestava klijenta o gresci
    Err.Raise Err.Number, "People Customer Module", Err.Description
End Sub
```

Sledeći kod se nalazi u aplikaciji koja koristi ovaj .dll.

```
Option Explicit

Private Sub cmdAddCustomer_Click()
Dim objCustomer As People.Customer
Set objCustomer = New People.Customer
objCustomer.AddCustomer txtFirstName, txtLastName, txtAge
End Sub
```



ACTIVEX DATA OBJEKTI (ADO)

Jednu od novih mogućnosti koju nudi Visual Basic 6.0 predtavljaju ADO Objekti i OLE DB standard.

OLE DB predstavlja najnoviju strategiju Microsofta pri dizajniranju database-oriented aplikacija. Ideja je bila de se prevaziđu ograničenja ODBC, koji omogućuje pristup samo relacionim bazama podataka. OLE DB je programski interfejs na nivou sistema koji omogućuje pristup svim vrstama podataka. OLE DB je sistem koji ima tri komponente: data providers, data consumers i service components.

Data providers su npr. Microsoft SQL Server, Exchange Server ili čak delovi operativnog sistema, kao što je na primer fajl sistemi koji nude podatke kojima druge aplikacije pristupaju. Interesantno je da postoje OLE DB provajedri za ODBC, koji omogućuje OLE DB data consumers pristup ODBC bazama podataka.

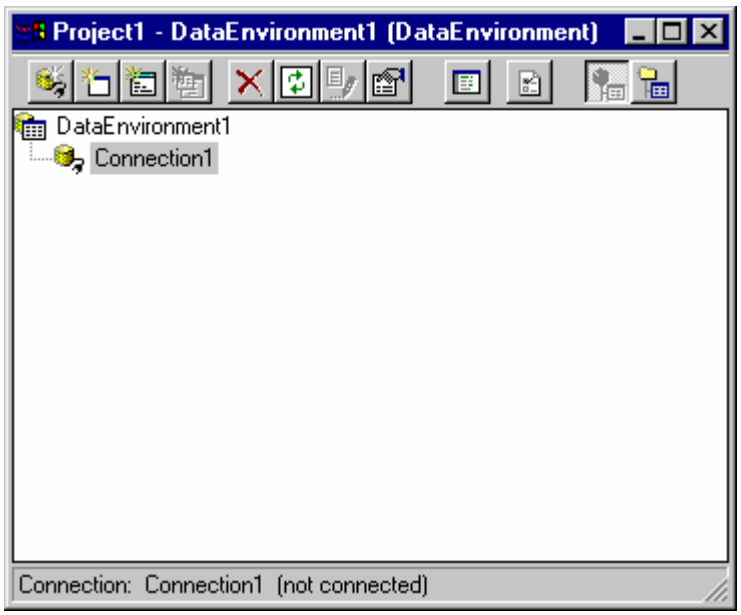
Data consumers su aplikacije koje koriste podatke koje izlažu data provajderi. ADO je zapravo programski interfejs za korišćenje OLE DB data. Svaka aplikacija koja koristi ADO je OLE DB data consumer.

Service components su delovi OLE DB koje obrađuju i transportuju podatke.

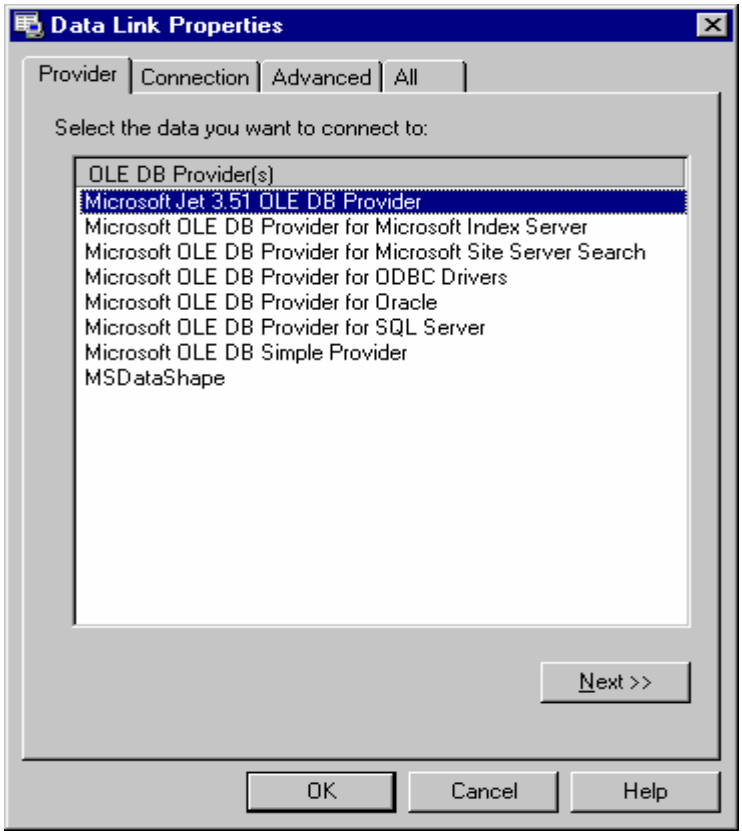
ADO je evolucija RDO i DAO arhitektura, kombinuje najbolje od obe arhitekture i zamenjuje ih sa interfejsom koji je veoma jednostavan za korišćenje. RDO i DAO ograničavaju programera na korišćenje ODBC i Jet data provajdera. ADO, s druge strane, obezbeđuje brz pristup svim tipovima podataka i informacija koje su dostupne preko OLE DB-a. Pri tome ADO predstavlja malo opterećenje za RAM i hard disk.
ADO se mogu razvijati pomoću svih alata koje nudi Visual Studio i svaka razvojna platforma daje međukooperativne ADO komponente.
Glavne komponente ADO objektnog modela su **Connection** objekat, **Command** objekat i **Recordset** objekat.
ADO podržava i tri kolekcije - **Errors** kolekciju, **Parameters** kolekciju i **Fields** kolekciju.

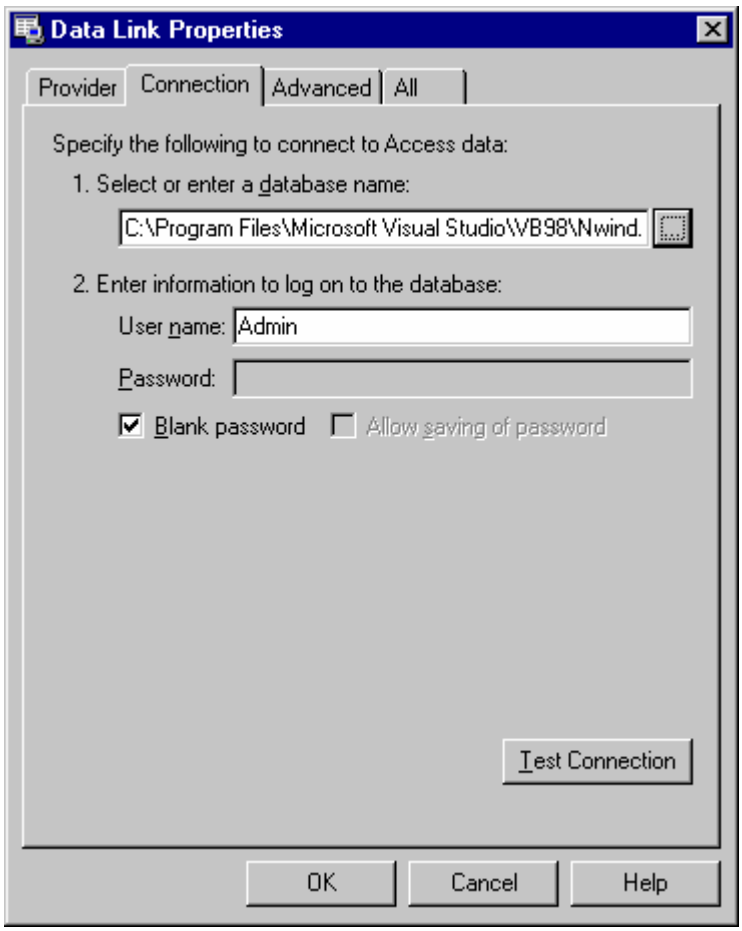
Connection objekat

Connection objekat omogućuje uspostavljanje veze sa izvorom podataka.
Connection objekat omogućuje aplikaciji da prenosi informacije o klijentu, kao što su korisničko ime i lozinka, ka bazi podataka gde se obavlja validacija.
Da bi se ADO uopšte mogli koristiti u Visual Basic aplikaciji mora se prvo postaviti referenca na Microsoft ActiveX Data Objects biblioteku. To se radi izborom Reference na Project meniju.
Konekcija sa izvorom podataka se izvodi kreiranjem instance Connection objekta i postavljanjem vrednosti argumenata.



Desni klik na Connection objekat i izbor Properties stavke daju Data Link Properties Dialog Box u kome se prvo vrši izbor OLE DB Provider-a. Izbor OLE DB Provider-a zavisi od tipa izvora podataka na koji želim da se konektujem. Zatim treba izabrati konekcione parametre za izabrani izvor podataka, koji zavise od tipa izvora podataka





Argumenti konekcije:

User ID	Validno korisničko ime
Password	Validna lozinka korisnika
Data Source	Ime udaljenog servera
Initial Catalog	Ime baze podataka koja se nalazi u spoljnem izvoru podataka

Ne preporučuje se korišćenje više konekcija iz iste aplikacije na istu bazu podataka, zato što svaka konekcija troši resurse i na klijent i na server strani, međutim ukoliko iz jedne aplikacije pristupam podacima iz različitih izvora podataka neizbežno je otvaranje više Connection objekata.

Command objekat

ADO Command objekti pristupaju upitima i prave upite koji se izvršavaju u okviru izvora podataka. Moguće je koristiti Command objekat za pristup

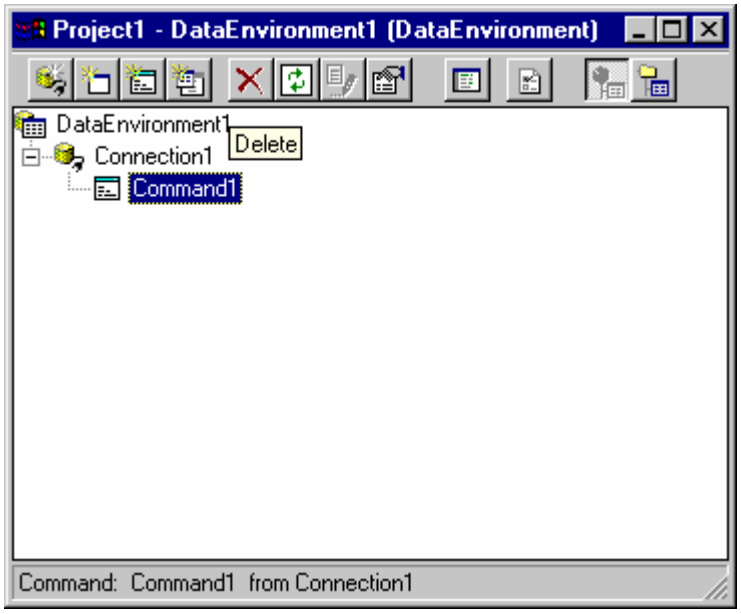
uskladištenim procedurama u eksternoj bazi podataka (od značaja za SQL Server).

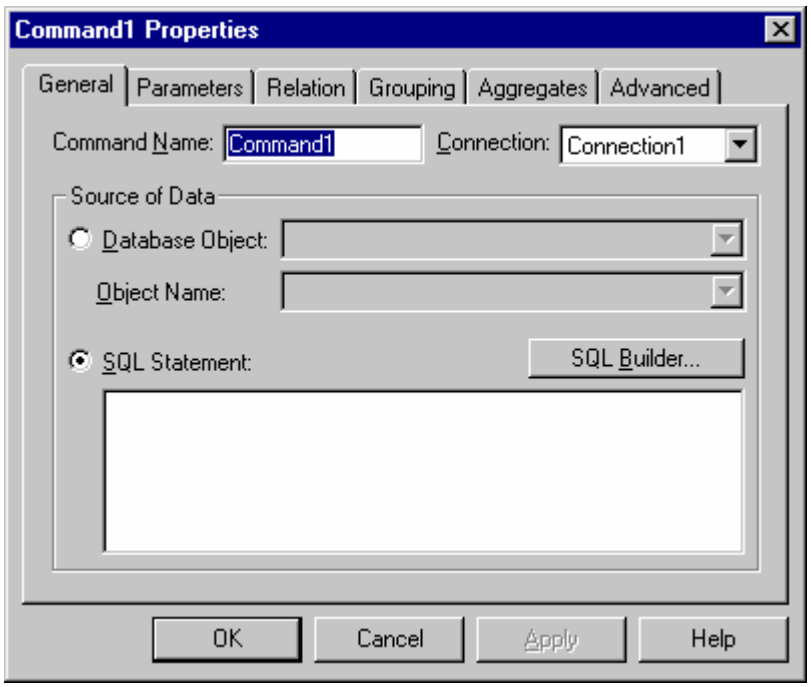
Svojstva:

ActiveConnection	Postavlja ili vraća aktivnu konekciju koju koristi objekat.
CommandText	SQL naredba, ime uskladištene procedure ili ime tabele koju će objekat koristiti.
CommandType	Kaže da li je CommandText svojstvo SQL naredba, uskladištena procedura ili ime tabele.
Prepared	Kaže da li SQL naredbu kreirati kao privremenu uskladištenu proceduru.
State	Kaže da li je naredba trenutno otvorena. zatvorena ili se izvršava.

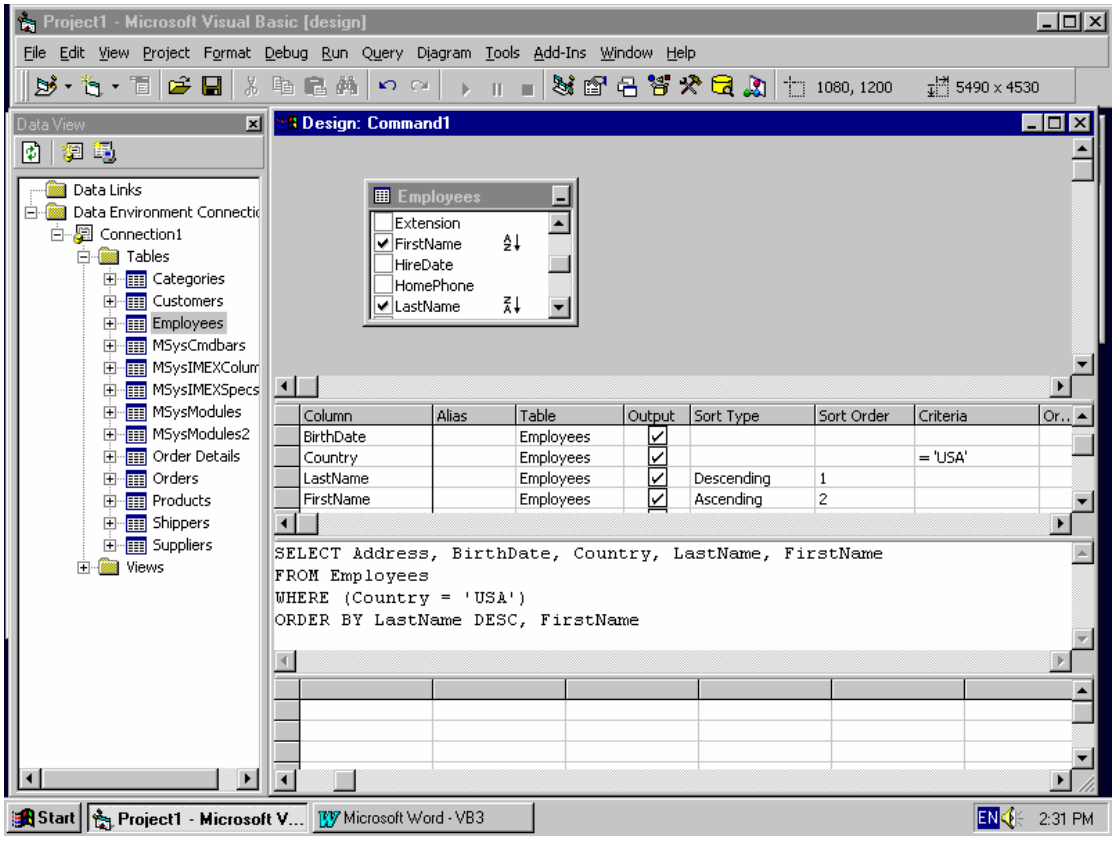
Metode:

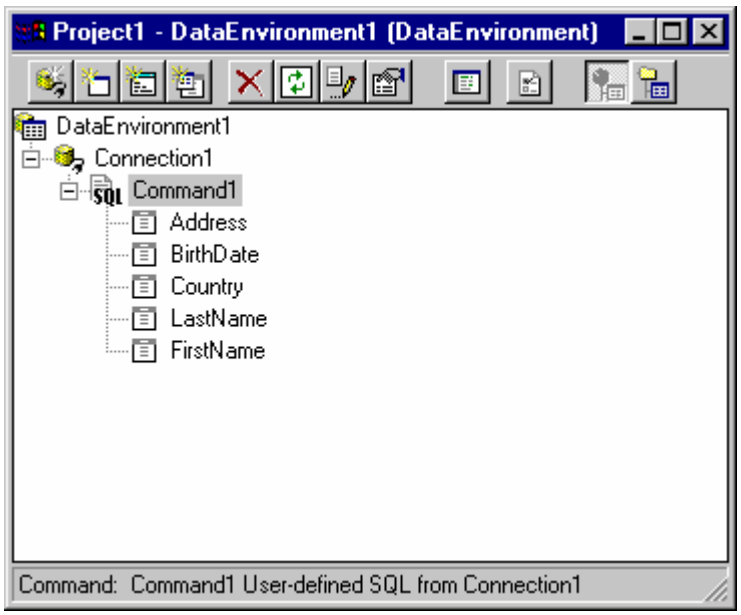
Cancel	Poništava naredbu koja se trenutno izvršava
CreateParameter	Pravi parameter object (koji se koristi sa uskladištenim procedurama).
Execute	Izvršava SQL naredbu.



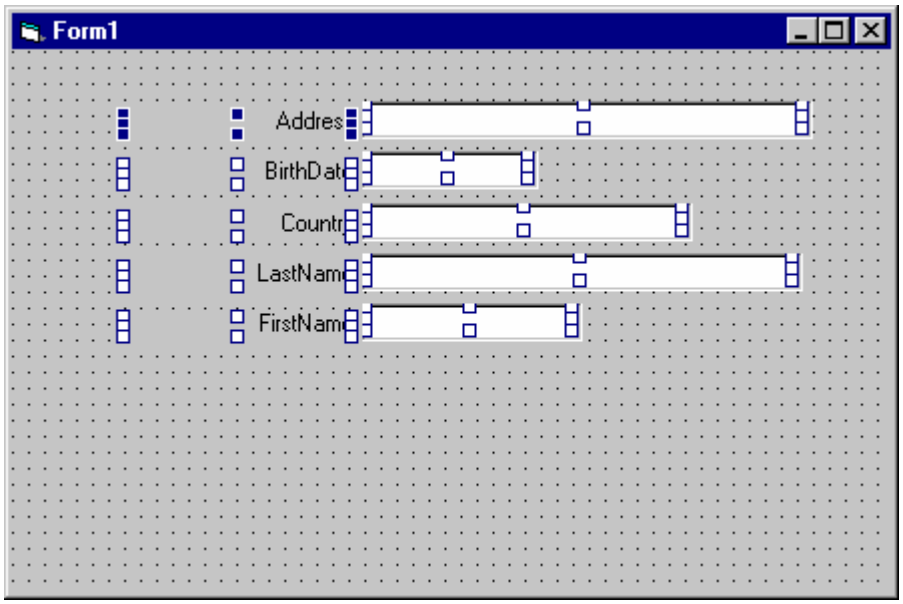


Dodavanje i podešavanje svojstava Command objekta se obavlje na sličan način kao i kod Connection objekta – otvaranjem Properties dialog box-a i unošenjem odgovarajućih parametara.

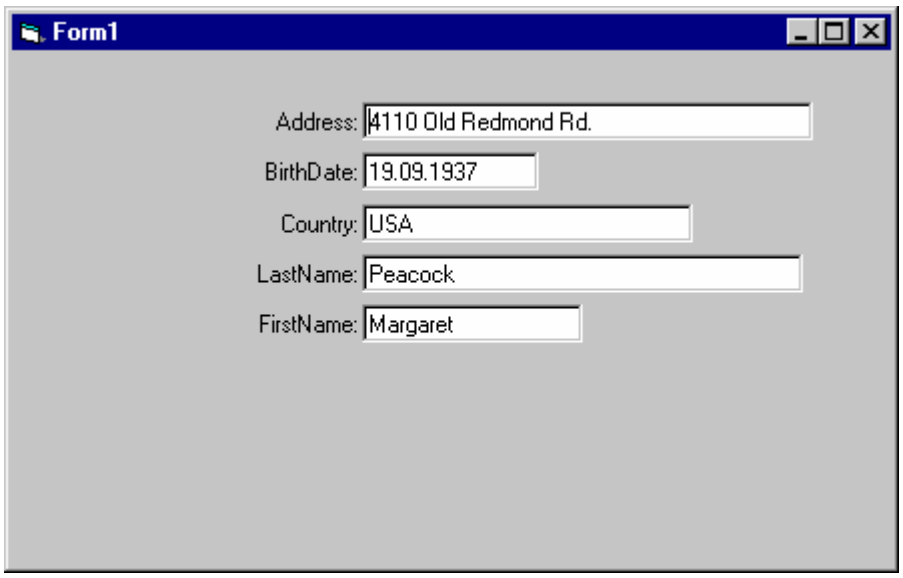




Napravljen Command objekat omogućuje da se na veoma efikasan način, korišćenjem miša i drag-and-drop metode, prave forme za prikazivanje i editovanje podataka u bazi podataka.



Kad se aplikacija startuje dobija se forma koja je sasvim funkcionalna i sa još par linija koda omogućuje punu fleksibilnost u radu sa konektovanom bazom podataka



Recordset objekat

Omogućuje aplikaciji da pristupa podacima vraćenim od strane SQL upita. Taj upit može biti kreiran od strane aplikacije, ili može biti smešten na server kao uskladištena procedura. Koristeći Recordset moguće je kretati se među vraćenim zapisima ili editovati njihove vrednosti.

Svojstva:

ActiveCommand	Vraća aktivnu naredbu za recordset.
ActiveConnection	Postavlja ili vraća aktivnu konekciju za recordset.
LockType	Postavlja ili vraća vrstu zaključavanja zapisa. Default je adLockReadOnly.
MaxRecords	Postavlja ili vraća maksimalni broj vraćenih zapisa.
PersistFormat	Određuje format u kome će biti snimljeni recordset data pri pozivu Save metoda.
RecordCount	Vraća broj zapisa u recordsetu
State	Vraća trenutno stanje recordseta.

Metode:

Open	Izvršava SQL naredbu.
Close	Zatvara recordset.
Requery	Ponovo izvršava SQL naredbu i ponovo kreira recordset.
Resync	Osvežava keširane zapise u recordsetu.
Save	Snima otvoreni recordset u fajl koji se može ponovo otvoriti kasnije.

Navigacija kroz recordset

Od svih ADO objekata, samo Recordset omogućuje korisnicima da se kreću kroz grupu vraćenih zapisa. Samo jedan zapis u okviru recordseta može biti vraćen u datom trenutku.

Svojstva recordseta za navigaciju:

AbsolutePage	Postavlja ili vraća stranu u kojoj se aktuelni zapis nalazi.
AbsolutePosition	Postavlja ili vraća apsolutnu poziciju zapisa.
BOF	Kaže da li je pokazivač usmeren na BOF
Bookmark	Vraća jedinstveni identifikator za aktuelni zapis.
EOF	Kaže da li je pokazivač usmeren na EOF.

Metode recordseta za navigaciju:

Move
MoveFirst
MoveLast
MoveNext
MovePrevious

Sortiranje

Sortiranje zapisa se vrši koristeći Sort svojstvo recordseta:

```
rsKlijenti.Sort = "Prezime ASC"
```

Da bi povratio originalni redosled treba onemogućiti sortiranje:

```
rsKlijenti.Sort = ""
```

Filtriranje

Filter svojstvo prikazuje samo one zapise recordseta koji ispunjavaju zahteve filtera:

```
rsKlijenti.Filter = "Uplate > 1000"
```

Da bi učinio sve originalne komponente dostupnima treba ukinuti filtriranje, što se može učiniti na sledeći način:

```
rsKlijenti.Filter = adFilterNone
```

Pretraga

Pretraga zapisa se vrši pomoću Find metode:

```
rsKlijenti.Find "Prezime = 'Jovanovic'"
```

Pri specificiranju string vrednosti, tekst uokviriti apostrofima, a za datume koristiti #.
Izraz za pretragu sadrži ime polja po kome se vrši pretraživanje, operator (=, <, >, "like") i vrednost za kojom se vrši pretraga koja može biti broj, string ili datum.

Izmena podataka

Izmena podataka se generalno može obavljatati na dva načina.

1) Koristeći **Execute** metode za izvršavanje SQL naredbi, koje mogu biti definisane na jedan od sledećih načina:

- pomoću SQL Insert naredbe koja dodaje jedan ili više novih zapisa izvoru podataka,
- pomoću SQL Update naredbe koja menja zapis ili grupu zapisa,
- pomocu SQL Delete naredbe koja briše jedan ili više zapisa.

2) Pomoću Recordseta, ali time smo ograničeni na jedno dodavanje, brisanje ili ažuriranje u datom trenutku:
- dodavanje novog zapisa korišćenjem AddNew i Update metoda recordseta,
- brisanje aktuelnog zapisa u recordsetu pomoću Delete metode recordseta,
- ažuriranje aktuelnog zapisa u recordsetu pomoću Update metode.

Diskonektovani Recordset objekti

ADO omogućuje korišćenje diskonektovanih recordset objekata, koji dozvoljavaju aplikaciji da napravi recordset, diskonektuje se od izvora podataka i omogući korisniku da pregleda i edituje taj diskonektovani recordset. Kada korisnik napravi željene izmene, aplikacija se može rekonektovati na izvor podataka i ažurirati bazu podataka.

Primer: konektovanje na nwind.mdb bazu podataka pomoću ADO objekata sa deklarisanjem Connection objekta u okviru koda.
U ovom primeru se ne koristi DataEnvironment, međutim, iako komplikovaniji ovaj način nudi veću fleksibilnost.


```
Option Explicit

Dim adoRS As ADODB.Recordset

Private Sub bFirst_Click()
    adoRS.MoveFirst
    If adoRS.BOF Then adoRS.MoveFirst
End Sub

Private Sub bLast_Click()
    adoRS.MoveLast
    If adoRS.EOF Then adoRS.MoveLast
End Sub

Private Sub bPrevious_Click()
    adoRS.MovePrevious
    If adoRS.BOF Then adoRS.MoveFirst
End Sub

Private Sub bNext_Click()
    adoRS.MoveNext
    If adoRS.EOF Then adoRS.MoveLast
End Sub



Private Sub bNew_Click()
    adoRS.AddNew
    Text1.SetFocus
End Sub

Private Sub bSave_Click()
    adoRS.Update
End Sub

Private Sub bDelete_Click()
```

```
    adoRS.Delete
End Sub

Private Sub bCancel_Click()
    adoRS.CancelUpdate
End Sub

Private Sub Form_Load()
  Dim adoCon As Connection
  Set adoCon = New Connection
  adoCon.CursorLocation = adUseClient
  adoCon.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data _ Source=C:\Program
Files\Microsoft Visual Studio\VB98\Nwind.mdb;"

Set adoRS = New Recordset
adoRS.Open "select CustomerID,CompanyName,City from Customers ORDER _ BY
CompanyName", adoCon, adOpenStatic, adLockOptimistic

Dim Ctl As Control
For Each Ctl In Me.Controls
    If TypeOf Ctl Is TextBox Then
        Set Ctl.DataSource = adoRS
    End If
Next

End Sub
```

Pravljenje izveštaja u Visual Basicu

Ranije verzije Visual Basica nisu imale na zadovoljavajući način rešen problem pravljenja izveštaja. Visual Basic 6.0 nudi izvanredan alat za pravljenje izveštaja – DataReport. DataReport se dodaje u projekat preko Project menija i u saradnji sa DataEnvironment-om omogućuje efikasno pravljenje izveštaja jednostavnim drag-and-drop postupkom. Potrebno je još samo postaviti svojstva DataSource i DataMember na odgovarajući Connection objekat (preko DataEnvironment-a) i na odgovarajući Command objekat. Napravljeni DataReport se startuje korišćenjem Show metode DataReporta.

```
DataReport1.Show 1
```

Primer: dodati DataEnvironment sa referencom na CustomerID, CompanyName i City polje iz Customers tabele u okviru nwind.mdb baze podataka. Dodati DataReport u projekat i drag-and-drop metodom prevući sva tri polja u Detail deo DataReporta.
Na glavnu formu aplikacije (iz prethodnog primera) dodati CommandButton, postaviti Name na bReport i napisati sledeći kod:

```
Private Sub bReport_Click()
    DataReport1.Show 1
End Sub
```

Pošto se aplikacija startuje i pritisne dugme REPORT dobija se željeni izveštaj, koji može da se štampa i eksportuje u druge aplikacije.

WEB BROWSER KONTROLA

WebBrowser kontrola omogućuje Visual Basic aplikaciji da obavlja pretraživanja (browsing), vrši pregled dokumenata i izvršava downloading. Ona omogućuje korisnicima da pretražuju sajtove na World Wide Web-u ili direktorijume na lokalnom fajl sistemu ili lokalnoj mreži.
WebBrowser je ActiveX kontrola i kao takva mora biti smeštena u formu. Korišćenje ove kontrole omogućuje pristup DHTML objektnom modelu (koji se koristi za dokumente).
Navigacija WebBrowser konrole se ostvaruje pomoću hipelinkova i URL-ova (Uniform Resource Locator). Kontrola omogućuje podršku za rad sa HTML i DHTML dokumentima, ali je takođe i kontejner za Active dokumente. Složeniji dokumenti, kao što je Microsoft Excell spreadsheet ili Microsoft Word dokument, takođe mogu biti otvoreni i editovani u okviru WebBrowser kontrole. WebBrowser kontrola ne spada u standardni set Visual Basic-ovih kontrola i potrebno ju je dodati preko Components stavke u Project meniju izborom Microsoft Internet Controls na Controls tabu.

Za rad sa WebBrowser kontrolom od interesa su sledeći elementi:

- **Navigate** metod koji odredjuje koji resurs, identifikovan pomoću URL,se pregleda.

- **LocationURL** i **LocationName** svojstva u **NavigateComplete** događaju prikazuju ime i URL resursa otvorenog u WebBrowser kontroli.

- **GoBack**, **GoForward**, **GoSearch** i **GoHome** metode obavljaju navigaciju izmedju sajtova.

- **Busy** svojstvo određuje da li je **WebBrowser** kontrola u stanju navigacije ka novoj lokaciji ili se vrši downloading fajla.

- **Stop** metoda poništava navigaciju ili download pre nego što se izvrši.

- **Refresh** metod ponovo učitava stranu koja je već prikazana.

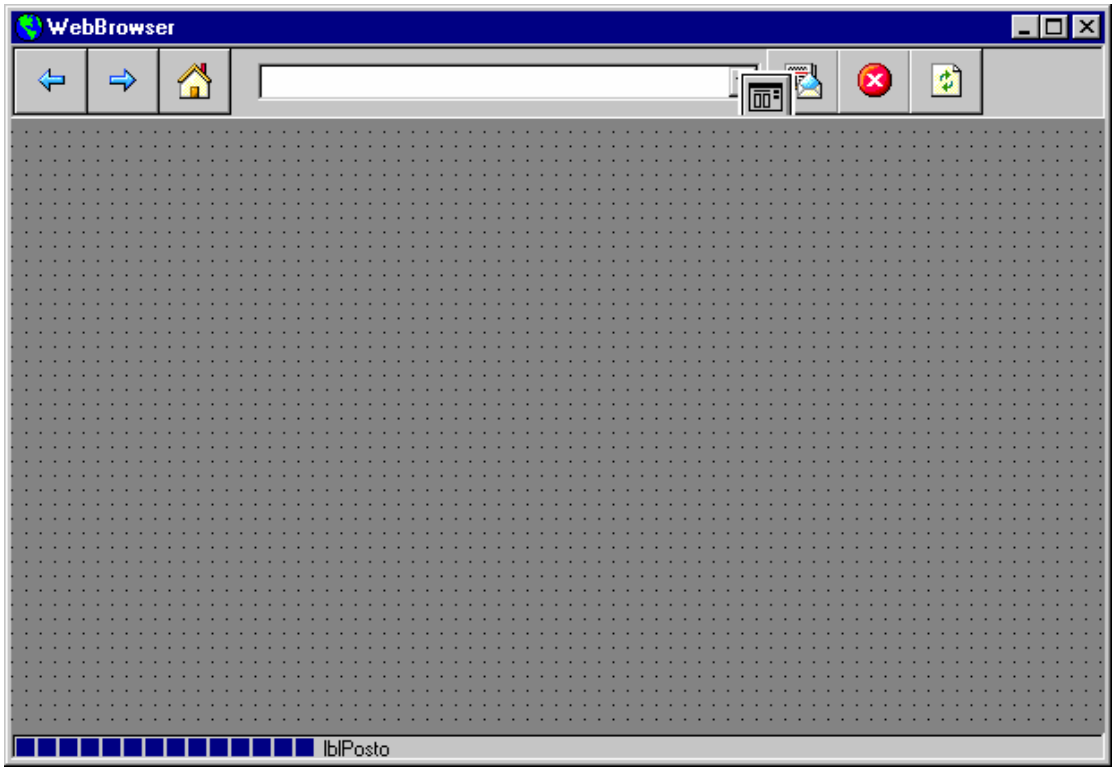
- **DownloadBegin**, **ProgressChange** i **DownloadComplete** događaji prikazuju informacije o statusu za vreme download-a strane.

Primer: korišćenje WebBrowser kontrole. Primer sadrži jednu MDI Formu, jednu child formu i class modul.

Kod u modulu:

```
Option Explicit

Public PageCount As Integer
Public CurPage As Integer

Public Sub SetNavButtons()
Exit Sub
  If PageCount = 0 Or PageCount = 1 Then
      frmMAIN.cmdBack.Enabled = False
      frmMAIN.cmdForward.Enabled = False
      Exit Sub
  End If
  If CurPage = 1 Then frmMAIN.cmdBack.Enabled = False
  If CurPage = PageCount Then frmMAIN.cmdForward.Enabled = False
  If CurPage > 1 And CurPage < PageCount Then
      frmMAIN.cmdBack.Enabled = True
      frmMAIN.cmdForward.Enabled = True
  End If
  If CurPage > 1 Then frmMAIN.cmdBack.Enabled = True
PageCount
End Sub
```

MDI Forma:



Kod MDI Forme:

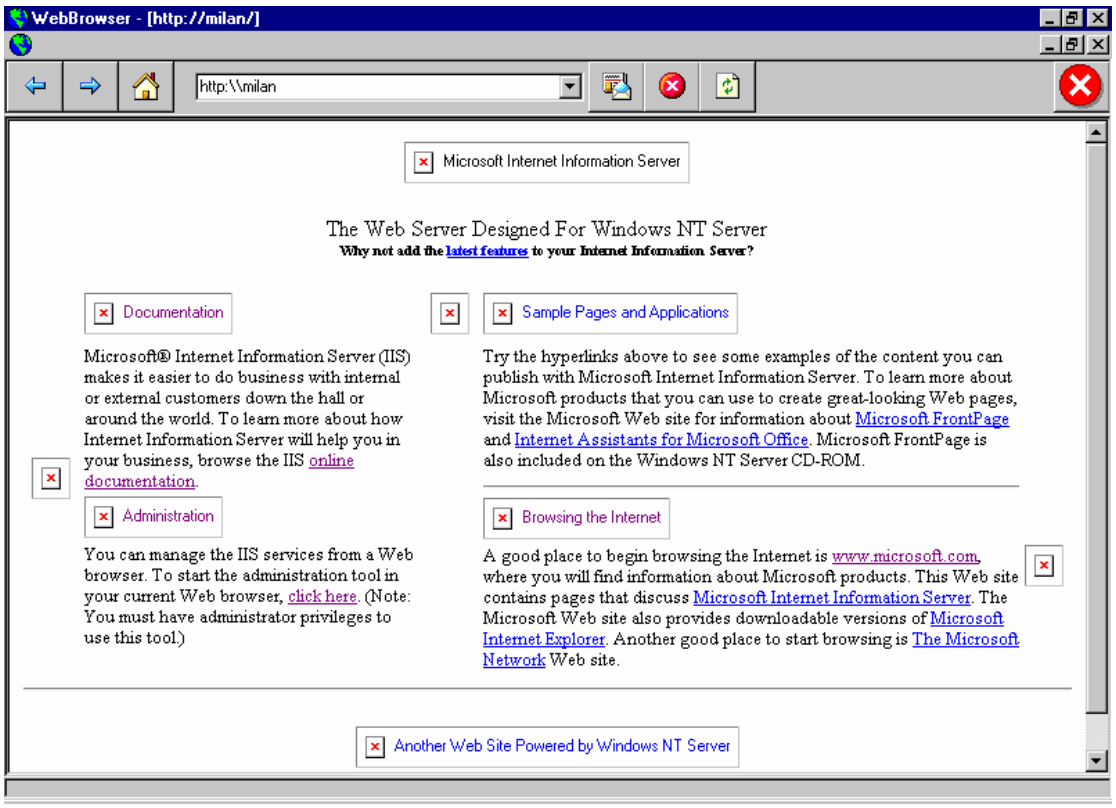
```
Option Explicit

Private Sub cmbLocation_KeyDown(KeyCode As Integer, Shift As Integer)
On Error Resume Next
```

```
  If KeyCode = 13 Then
    Me.ActiveForm.web.Navigate (cmbLocation.Text)
    If Err.Number Then
      MsgBox "Wrong URL", vbCritical, "Navigate"
      Err.Clear
    End If
  End If
End Sub

Private Sub cmdBack_Click()
  On Error Resume Next
Me.ActiveForm.web.GoBack
  CurPage = CurPage - 2
  PageCount = PageCount - 1
Call SetNavButtons
End Sub

Private Sub cmdBrowse_Click()
  On Error Resume Next
  cd.Flags = cdlOFNFileMustExist
  cd.ShowOpen
  If Err.Number Then
    Err.Clear
    Exit Sub
  End If
  Me.ActiveForm.Refresh
  Me.Picture1.Refresh
  Me.ActiveForm.web.Navigate (cd.FileName)
  cmbLocation.Text = cd.FileName

End Sub

Private Sub cmdExit_Click()
  End
End Sub

Private Sub cmdForward_Click()
  On Error Resume Next
Me.ActiveForm.web.GoForward
PageCount = PageCount - 1
Call SetNavButtons
End Sub

Private Sub cmdHome_Click()
  On Error Resume Next
  Me.ActiveForm.web.GoHome
End Sub

Private Sub cmdRefresh_Click()
  Me.ActiveForm.web.Refresh
End Sub



Private Sub cmdStop_Click()
With Me.ActiveForm
  If .web.Busy Then .web.Stop
End With
End Sub

Private Sub MDIForm_Load()
  Call SetNavButtons
```

```
  Dim x As Form
  Set x = New frmStart
  x.Show
  If Trim$(Command$) <> "" Then
    Me.ActiveForm.web.Navigate (Command$)
    cmbLocation.Text = Command$
  End If
End Sub
```

Kod child forme:

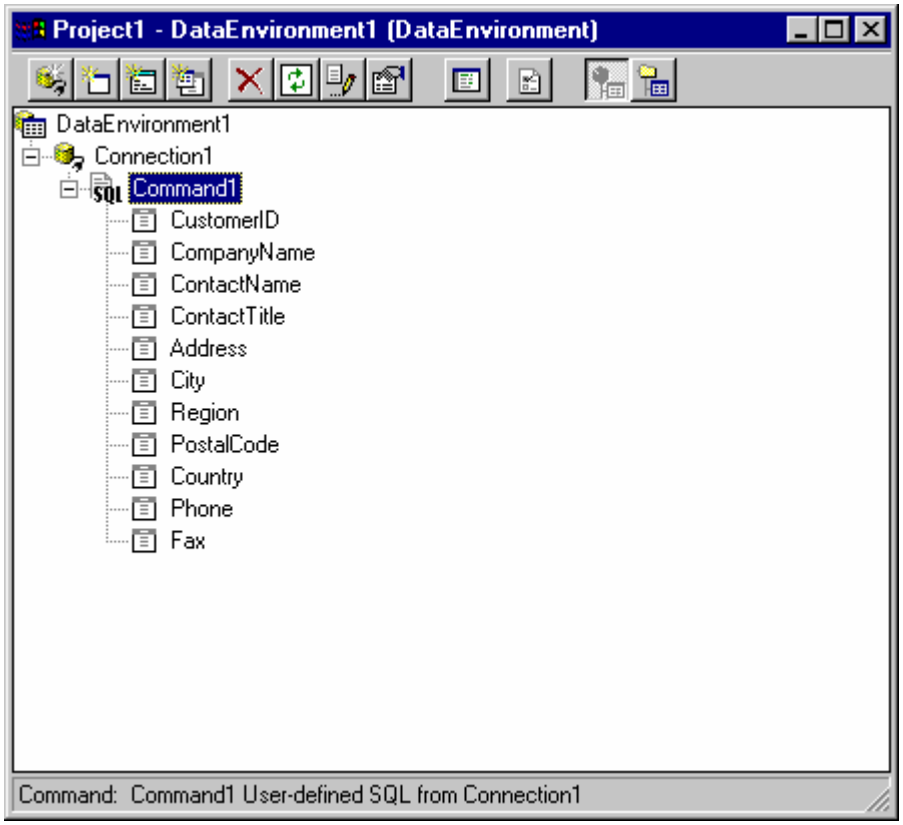
```
Private Sub Form_Resize()
  On Error Resume Next
  web.Width = Me.Width - 100
  web.Height = Me.Height - 400
End Sub

Private Sub Form_Unload(Cancel As Integer)
  End
End Sub

Private Sub web_DownloadBegin()
  On Error Resume Next
  Me.Caption = "Wait..."
  frmMAIN.pb.Visible = True
  frmMAIN.Label1.Caption = ""
  frmMAIN.Label1.Visible = True
End Sub

Private Sub web_DownloadComplete()
  On Error Resume Next
  Me.Caption = web.LocationURL
  frmMAIN.pb.Visible = False
  frmMAIN.Label1.Visible = False
  PageCount = PageCount + 1
  CurPage = CurPage + 1
  Call SetNavButtons
End Sub

Private Sub web_ProgressChange(ByVal Progress As Long, ByVal ProgressMax As
Long)
  On Error Resume Next
  frmMAIN.pb.Value = 0
  If Progress <> -1 And ProgressMax <> 0 Then
    frmMAIN.pb.Value = Progress * 100 / ProgressMax
    frmMAIN.pb.Refresh
    frmMAIN.Label1.Caption = Int(frmMAIN.pb.Value) & " %"
    frmMAIN.Label1.Refresh
  End If
End Sub
```

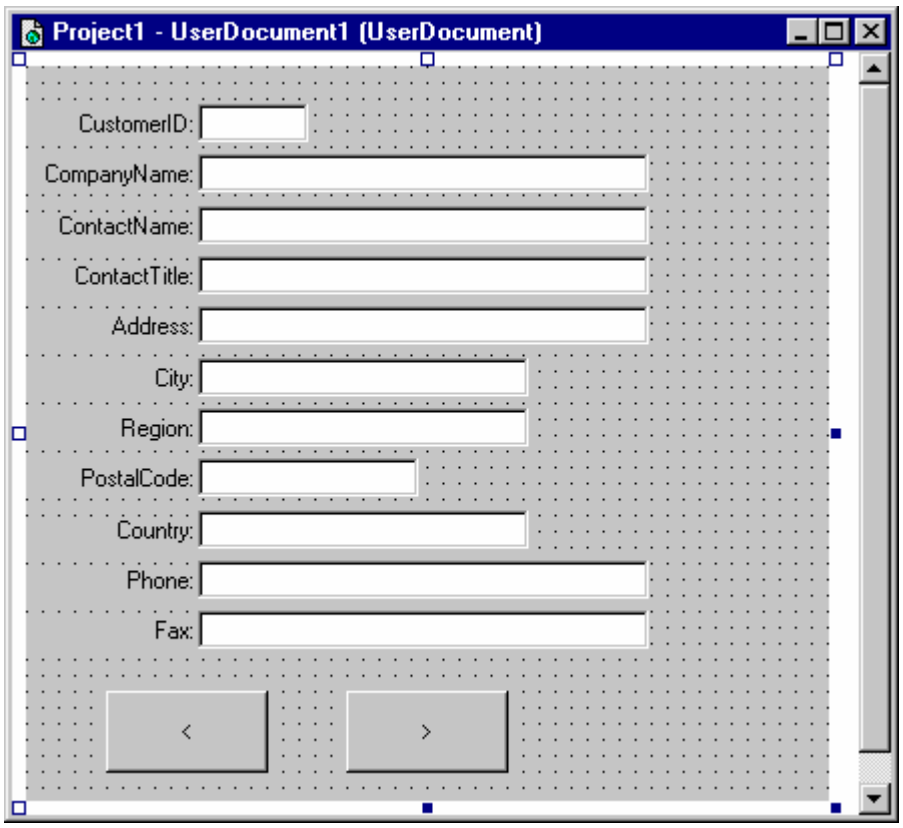


ACTIVE DOCUMENTS

Active documents su aplikacije koje su smeštene kao deo Internet browser prozora, ili bilo koje druge kontejner aplikacije koja podržava OLE dokument objekte.
Ukoliko programer poznaje Visual Basic, on već ima sposobnosti da razvija ovakve aplikacije, što znači da ne mora da poznaje HTML da bi kreirao aplikacije za Internet. Pri tome programer može da koristi razvojno okruženje Visual Basica, uključujući code window, debugger i compiler.
Active document pristup omogućuje da se obrada obavlja lokalno i smanjuje procesiranje na mreži.
Programer može da vidi izgled aplikacije u istom trenutku kad je i kreira, što nije moguće pri pisanju za HTML.
Active document se kreira dodavanjem kontrola i koda UserDocument objektu. UserDocument je osnova za sve Active documents i kod njega je implementiran interfejs koji omogućuje da UserDocument bude smešten unutar nekog kontejnera za dokumente.

Primer: Active dokument koji pristupa bazi podataka nwind.mdb i pristupa tabeli Customers. Projekat sadrži jedan UserDocument i jedan DataEnvironment i staruje otvaranjem Internet Explorer-a.

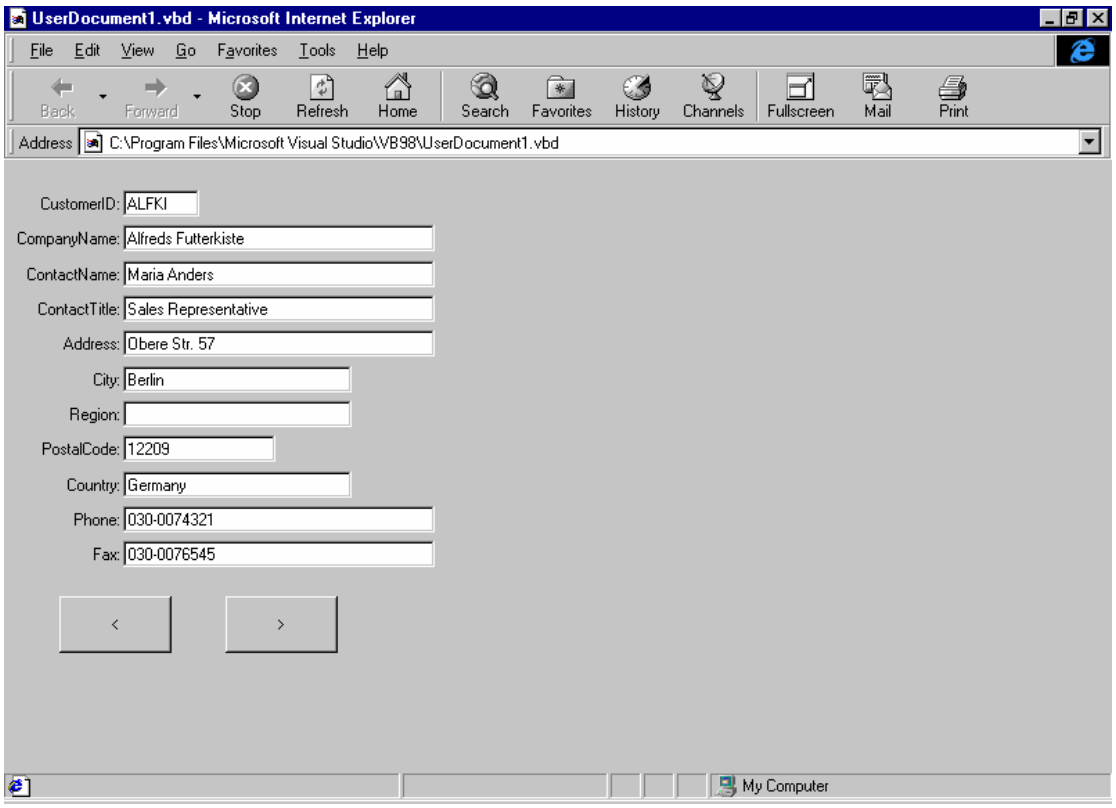




Kod za UserDocument:

```
Option Explicit

Private Sub Command1_Click()
      DataEnvironment1.rsCommand1.MovePrevious
End Sub

Private Sub Command2_Click()
      DataEnvironment1.rsCommand1.MoveNext
End Sub
```

Startovana aplikacija automatski startuje Internet Explorer i pristupa željenom sadržaju nwind.mdb baze podataka.



DHTML APLIKACIJE

Dynamic HTML (DHTML) omogućuje kreiranje Web orijentisanih aplkacija koje vrše obradu na Web klijentu i poseduju efikasnost i bezbednost COM komponente. DHTML je tehnologija ugrađena u Internet Explorer 4.0 koja definiše objektni model za HTML strane.
DHTML se može koristiti da odgovori na događaje i izmeni sadržaj na HTML stranici u svakom trenutku, ne samo pri download-ovanju ili osvežavanju strane.

Dakle, suština DHTML aplikacija je da se obrada obavlja na Web klijentu, mada je aplikacija u stanju da poziva i server.
DHTML aplikacija se sastoji od jedne ili više HTML strana i COM komponente. HTML strane predstavljaju korisnički interfejs za aplikaciju, dok COM komponenta sadrži funkcionalnost za tu aplikaciju.

DHTML Page designer omogućuje pravljenje Web orijentisanih aplikacija isto tako jednostavno kao i standardnih aplikacija u Visual Basicu. Iako je korisnički interfejs zapravo HTML strana, pri kreiranju DHTML aplikacija se koriste vizuelni alati i nema potrebe za korišćenjem HTML tagova.

DHTML može da radi u okruženju Internet Explorera 4.0 ili bilo koje druge aplikacije koja podržava DHTML, a to mogu biti i aplikacije napravljene korišćenjem WebBrowser kontrole.

Kreiranje DHTML aplikacije u Visual Basicu

- Otvoriti novi Visual Basic projekat koristeći DHTML Application projekat, koji sadrži DHTML Page designer i modul za pisanje koda.

- Napraviti novu HTML stranu, ili koristiti postojeću HTML stranu, kao korisnički interfejs za aplikaciju. Između DHTML Page designer-a i HTML strana posoji one-to-one relacija. Da bi se moglo koristiti više od jedne HTML strane u aplikaciji, potrebno je kreirati dodatne DHTML designer-e u projektu izborom Add DHTML Page na Project meniju.

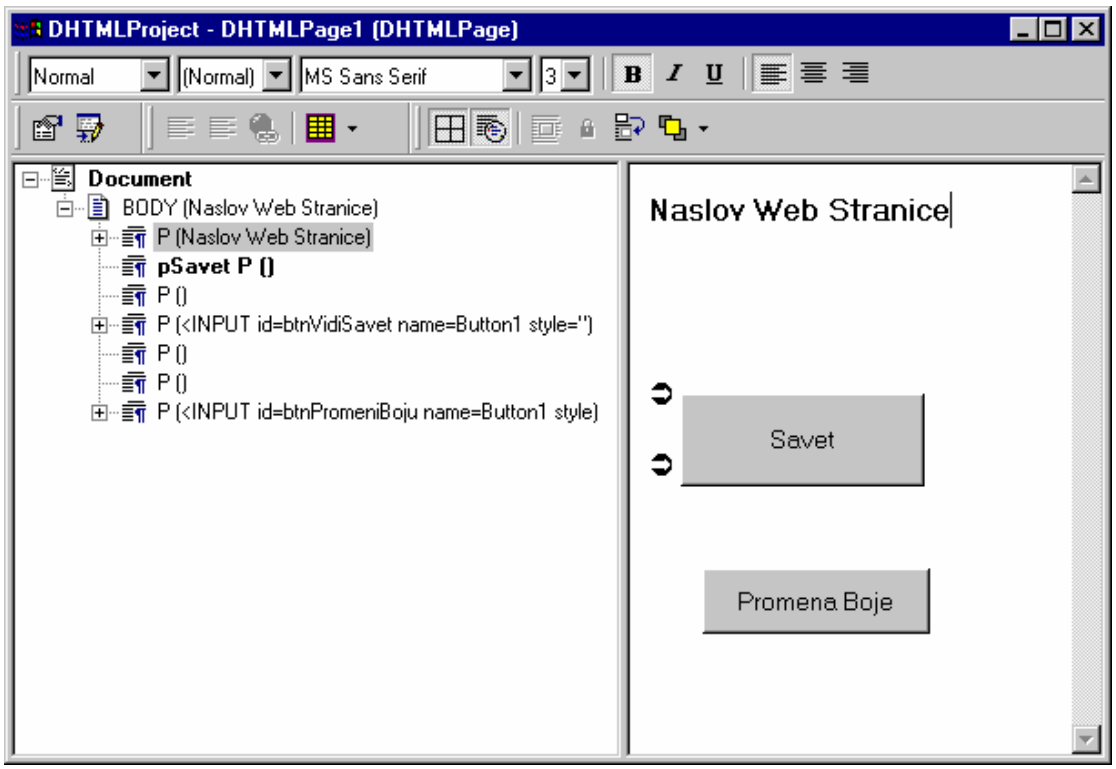
- Dodeliti jedinstveni identifikator svakom elementu HTML strane kome želim da pristupam u programu (preko koda).

- Napisati kod koji predstavlja funkcionalnost aplikacije.

- Testirati i debagovati aplikaciju, kao i svaku drugu rađenu u Visual Basicu.

- Kompajlirati aplikaciju.

Primer: DHTML aplikacija koja generiše HTML stranu čiji se sadržaj može dinamički menjati pomoću 2 Command Button-a. Projekat sadrži jedan DHTML Designer i jedan Modul.



Kod vezan za ova dva Command Button-a:

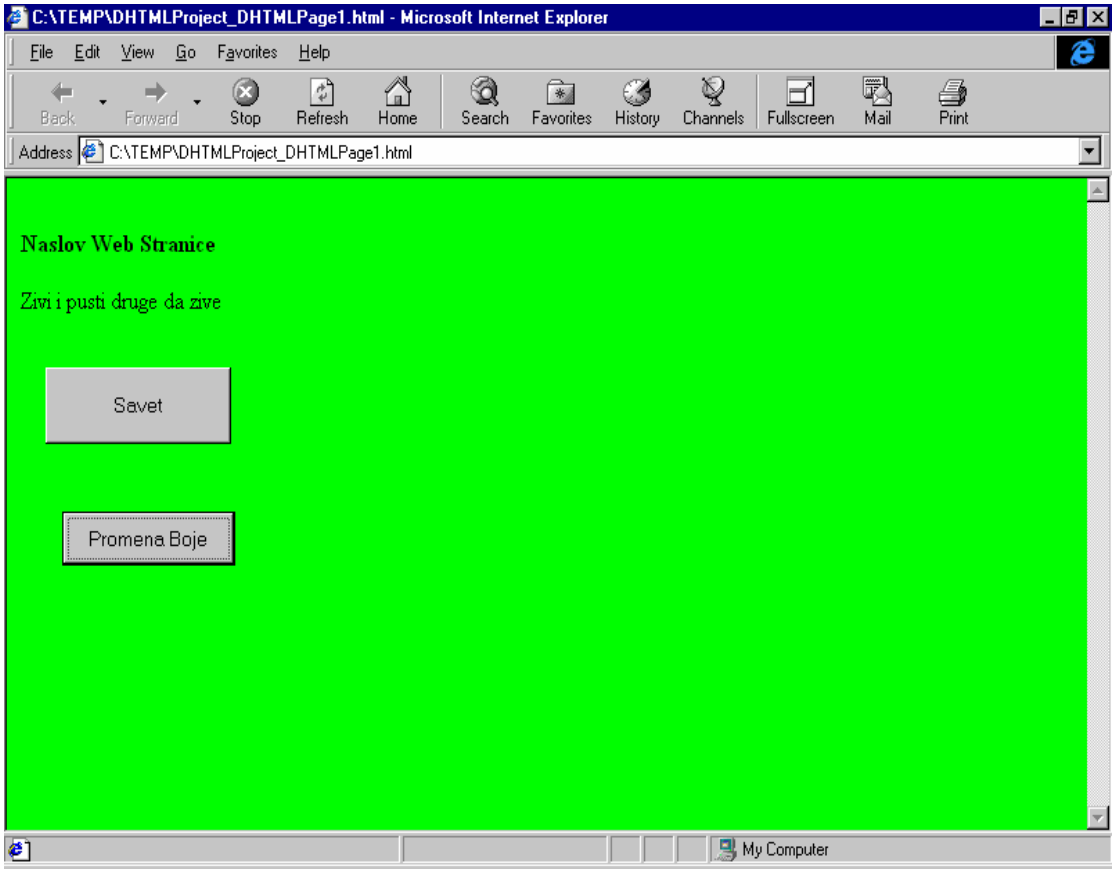
```
Private Function btnPromeniBoju_onclick() As Boolean
      Document.bgColor = RGB(0, 255, 0)
End Function

Private Function btnVidiSavet_onclick() As Boolean
      pSavet.innerText = "Zivi i pusti druge da zive"
End Function
```

Kod u modulu modDHTML, generisan od strane DHTML Designer-a, izgleda ovako:

```
'PutProperty: Store information in a cookie by calling this
'              function.
'              The required inputs are the named Property
'              and the value of the property you would like to store.
'
'              Optional inputs are:
`              expires : specifies a date that defines the valid
`              life time
'                           of the property.  Once the expiration date `
                     has been
'                           reached, the property will no longer be
`                           stored or given out.




Public Sub PutProperty(objDocument As HTMLDocument, strName As String,
vntValue As Variant, Optional Expires As Date)

     objDocument.cookie = strName & "=" & CStr(vntValue) & _
        IIf(CLng(Expires) = 0, ""; expires=" & _ Format(CStr(Expires),
"ddd, dd-mmm-yy hh:mm:ss") & " GMT") ' & _

End Sub
```

```
'GetProperty: Retrieve the value of a property by calling this
'             function.  The required input is the named Property,
'             and the return value of the function is the current
`             value
'             of the property.  If the proeprty cannot be found or
`             has expired,
'             then the return value will be an empty string.

Public Function GetProperty(objDocument As HTMLDocument, strName As String)
As Variant
    Dim aryCookies() As String
    Dim strCookie As Variant
    On Local Error GoTo NextCookie

    'Split the document cookie object into an array of cookies.
    aryCookies = Split(objDocument.cookie, ";")
    For Each strCookie In aryCookies
        If Trim(VBA.Left(strCookie, InStr(strCookie, "=") - 1)) = _
Trim(strName) Then
            GetProperty = Trim(Mid(strCookie, InStr(strCookie, "=") _ + 1))
            Exit Function
        End If
NextCookie:
        Err = 0
    Next strCookie
End Function
```

Aplikacija kad se startuje otvara Internet Explorer 4.0 i prikazuje generisanu Web stranu čiji se sadržaj i izgled mogu menjati.



TREE VIEW KONTROLA

Omogućava pregled stavki na više nivoa isto kao u Windows Explorer-u. Svaka stavka može biti na nekom od nivoa. Ako je na prvom nivou onda kažemo da je ta stavka roditelj, a ako ispod sebe ime pod-stavke, onda te stavke zovemo "deca" predhodne. Pomoću nje možemo prikazati neku hijerarhisku strukturu identičnu na primer, strukturi direktorijuma na disku. Svaka stavka može imati podstavke, podstavke mogu dalje imati svoje podstavke itd. Jednu stavku zovemo "ćvor" (node) i predstavljamo je objektnom promenjivom tipa Node.
Ova razgranata struktura mora biti u sprezi sa ImageList kontrolom. ImageList obezbeđuje slike (tipa Icon ili BitMap), koje omogućavaju grafičku prezentaciju granjanja.
Tipično, to su bitmape:

- zatvorena fascikla (c:\vbasic\bitmaps\outline\closed.bmp)
- otvorena fascikla (c:\vbasic\bitmaps\outline\open.bmp)
- dokument (c:\vbasic\bitmaps\outline\leaf.bmp)

Uobičajeno je da slika leaf.bmp predstavlja čvor koji ispod sebe nema "dece" i poslednji je u toj grani.

Properties dialog ListView kontrole:

- Style:
  više stilova grafičkog prikaza strukture,
  najčešće korišćeni je stil 7 (Lines,Plus/Minus,Image, Text)

- Line style:
  dve varijante prikaza linija koje povezuju čvorove
  Tree lines, Root lines

- Label edit
  Pošto svaki čvor sadrži tekst koji se vidi na ekranu, omogućeno je da se taj tekst edituje (potpuno isto kao kada bi ste hteli da u Windows Explorer-u izmenite naziv nekog fajla ili direktorijuma). Dakle, jednostruki klik mišem na predhodno selektovani tekst omogućava njegovo editovanje. Ako je Label edit postavljen na Automatic ovo se upravo ovako i događa, u suprotnom neophodno je programski obezbediti editovanje.

- Image List
  Unosi se (tj. bira iz liste) naziv ImageList kontrole čije slike će ListView koristiti za prikaz

- Sorted
  Ako je postavljeno na True. čvorovi u ListView kontroli se automatski sortiraju po tekstu koji je na njima.

Sintaksa za dodavanje novih čvorova u ListView kontroli je sledeca:

Predhodno dimenzionišemo Node objekat:
Dim A as Node

zatim,

Set A = TreeView1.Nodes.Add ([predak], [veza sa pretkom],
kljuc, tekst, slika, [Selektovana slika])

[predak]: ako dodajemo stavku koja treba da je "dete" neke ranije dodate stavke, ovde upisujemo Key te predhodne stavke "roditelja".Ako je stavka na prvom nivou ovaj parametar treba izostaviti.

[veza sa predkom]: ako smo definisali predhodnu opciju [predak] onda je obavezno definisati kakva je veza nove stavke sa svojim predkom. Postoji više mogucnosti, ali najčešće koristimo opciju tvwChild. Ona označava da je nova stavke dete stavke čiji je key [predak].

kljuc: jedinstveni identifikator svakog čvora u kontroli. Ne mozemo imati dva ista identifikatora na jednoj list View kontroli. Tip podataka je string i za njega važe isti kriterijumi kao i za naziv promenjive u Visual Basic-u.

Tekst: Tekst čvora.

Slika: Slika koja prezentuje ovaj čvor. Tip podataka je broj. Na primer 3, sto znaci da ovaj čvor prezentuje treća slika iz ImageList kontrole koju smo vezali za ListView kontrolu.

Selektovana slika: Opcioni broj slike iz ImageList kontrole koja će prezentovati selektovan čvor.

Dim A as Node

```
Set A = TreeView1.Nodes.Add ( , ,"N1","Prvi cvor",1)
```
Ovim smo dodali prvi čvor ciji je identifikator (key) N1, na kome piše "Prvi cvor" i koga će grafički predstavljati slika broj 1 iz dodeljene ImageList kontrole.

```
Set A = TreeView1.Nodes.Add ("N1", vwChild, "C1", "Dete prvog cvora",3)
```
Sada smo dodali novi čvor koji je dete (vwChild) čvora N1, na kome piše "Dete prvog cvora", i koga će grafički predstavljati slika broj 3 iz dodeljenje ImageList kontrole.

Primer: TreeView

```
Private Sub Command1_Click
' Add Node objects.
Dim nodX As Node
Dim i As Integer

For i = 0 To TreeView1.Nodes.Count - 1
TreeView1.Nodes.Clear
Next i



'Syntax: Add(relative, relationship, key, text, image, selectedimage)
'tvwLast     1    Last - Nod se dodaje na kraj posle svih ostalih ‘nodova na istom nivou na kojem je nod koji je naveden pod relative
'tvwNext     2    Next - Nod se dodaje posle noda koji je relative
'tvwPrevious 3    Previous - Nod se dodaje pre noda koji je relative
'tvwChild    4    Dete noda koji je relative

'Prvi nod “Koren”', slika 2, Key je “k”
Set nodX = TreeView1.Nodes.Add(, , "k", "Koren", 2)
nodX.ExpandedImage = 1        'Slika u otvorenom režimu je slika 1

' Sledeći nod  'Parent', slika 2
Set nodX = TreeView1.Nodes.Add(, , "r", "Roditelj", 2)
nodX.ExpandedImage = 1

Set nodX = TreeView1.Nodes.Add(, , "p", "Poslednji nod", 2)
nodX.ExpandedImage = 1

' Ovaj nod je dete Noda 1 ("Koren"),koristi sliku 3
Set nodX = TreeView1.Nodes.Add(1, tvwChild, "d", "Dete", 3)

'Sledeći nod je dete noda pod imenom “Roditelj”
'umesto da koristimo indeks koristimo Key noda “Roditelj” ("r.")
Set nodX = TreeView1.Nodes.Add("r", tvwChild, "nes", "Ne sortirano", 2)
nodX.ExpandedImage = 1

' Dodajemo tri noda sve deca noda “Ne sortirano”
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "xz", "Xyz", 3)
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "datum", "1967", 3)
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "srt", "Sortirano",2)
nodX.ExpandedImage = 1
' Sledeci kreirani nodovi će biti sortirani
nodX.Sorted = True

' Na kraju dodajemo tri noda, decu noda "Sorted,"
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "x", "Milan", 3)
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "j", "Petar", 3)
```

```
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "a", "Aleksandar", 3)
nodX.EnsureVisible
End Sub
```

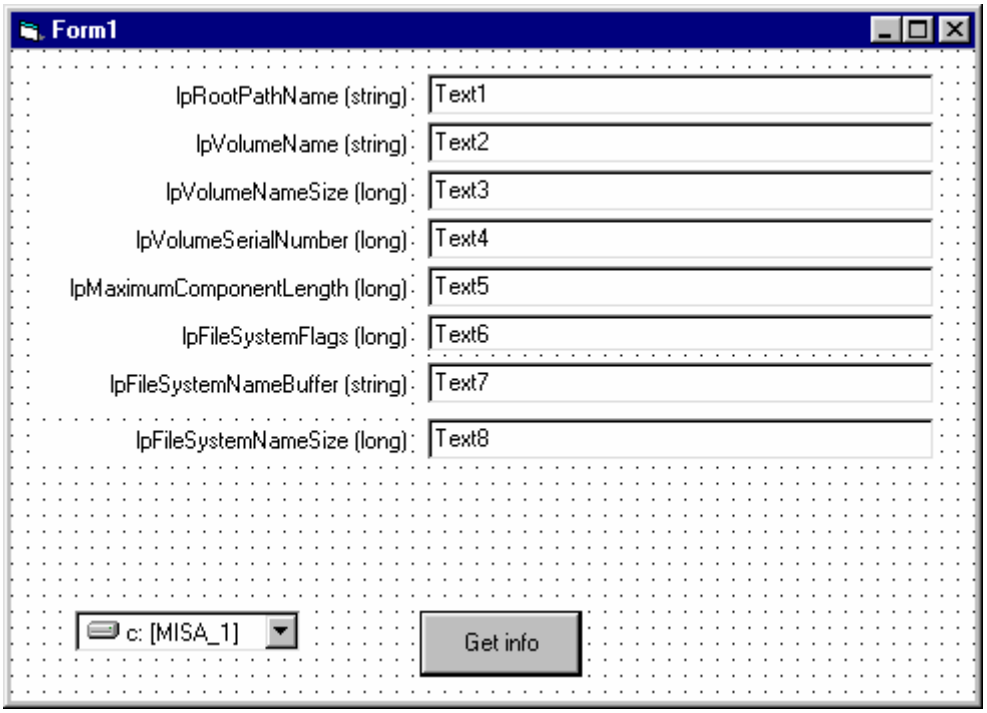
```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
Label1.Caption = Node.Text & "  " & Node.Index
End Sub

Private Sub Command2_Click()
Dim I As Integer
For I = 1 To TreeView1.Nodes.Count
TreeView1.Nodes(I).Expanded = True 'ili False
Next I
End Sub
```

VEZA SA EKSTERNIM DLL BIBLIOTEKAMA

Primer: Windows sistemski direktorijum

```
Declare Function GetWindowsDirectoryA Lib "kernel32" _
(ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Sub Command1_Click()
      Dim a As String * 255
      Dim x As Long
      x = GetWindowsDirectoryA(a, Len(a))
      MsgBox "Windows direktorijum: " & a, 64
End Sub
```

API Demo (Disk info, alwaus on top, wavplay)



```
Declare Function GetVolumeInformation Lib "kernel32" Alias _
"GetVolumeInformationA" _
(ByVal lpRootPathName As String, ByVal lpVolumeNameBuffer As String,_ ByVal
nVolumeNameSize As Long, lpVolumeSerialNumber As Long, _
lpMaximumComponentLength As Long, lpFileSystemFlags As Long, _
ByVal lpFileSystemNameBuffer As String, ByVal nFileSystemNameSize _
As Long) As Long

Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _ ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _ ByVal cx As
Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" _ (ByVal
lpszSoundName As String, ByVal uFlags As Long) As Long
'1 asinhrono, 2 sinhrono


Private Sub cmdGetInfo_Click()
      Dim lpRootPathName As String
      Dim lpVolumeName As String * 255
      Dim lpVolumeNameSize As Long
      Dim lpVolumeSerialNumber As Long
      Dim lpMaximumComponentLength As Long
      Dim FileSystemFlags As Long
      Dim FileSystemNameBuffer As String * 255
      Dim nFileSystemNameSize As Long
      Dim nIzlaz As Long

      lpRootPathName = Left(Drive1.Drive, 2) & "\"
      lpVolumeNameSize = Len(lpVolumeName)
      nFileSystemNameSize = Len(FileSystemNameBuffer)
      nIzlaz = GetVolumeInformation(lpRootPathName, lpVolumeName, _
      lpVolumeNameSize, lpVolumeSerialNumber, _ lpMaximumComponentLength,
      FileSystemFlags, _
      FileSystemNameBuffer, nFileSystemNameSize)

      Text1.Text = lpRootPathName
      Text2.Text = lpVolumeName
      Text3.Text = lpVolumeNameSize
      Text4.Text = lpVolumeSerialNumber
      Text5.Text = lpMaximumComponentLength
      Text6.Text = FileSystemFlags
      Text7.Text = FileSystemNameBuffer
      Text8.Text = nFileSystemNameSize
End Sub


Private Sub Form_Load()
      SetWindowPos Me.hwnd, -1, 0, 0, 0, 0, 3 'ili 1 ili 2 ili 3
      'ako je 1; pozicija in pix
End Sub
```

Visual Basic - napredni kurs

S K R I P T A

Vladimir Tasić

BAZE PODATAKA - ACCESS Terminologija

Visual basic je od samog početka imao jaku podršku za rad sa bazama podataka različitih formata. Činjenica je da ćete u praktičnom radu gotovo uvek kreirati aplikacije koje rade sa bazama podataka, tako da je neophodno teoretski i tehnički potpuno savladati ovu oblast.

Sama baza podataka je jedna datoteka na disku u kojoj su po strogo određenim pravilima smešteni podaci. Upravo ovi različiti načini i pravila smeštanja podataka definišu ono što u praksi zovemo format baze podataka. Imajući u vidu dinamiku razvoja računske industrije, jasno je da postoji puno različitih formata, zapravo, svaki veći (bivši ili sadašnji) proizvođač softvera je ustanovio svoj format baze. Mnogi od njih nisu zaživeli u praksi, a oni koji su opšte prihvaćeni su postali de facto standard za različite oblasti korišćenja. Da bi komplikacija bila veća, isti proizvođač, u toku usavršavanja, ima više verzija svog formata baze, koji su najčešće kompatibilni na dole. Ovo znači da novija verzija programa donosi i novu verziju formata baze, ali može čitati ili/i konvertovati stari format u novi.

Ovde možete postaviti pitanje, zbog čega postoji toliko različitih formata baze podataka ? Za ovo pitanje postoji puno odgovora, ali mislim da su naredna dva sveobuhvatna:

Zavisno od tipa informacija, njihovog kvantiteta, organizacije računara (da li je u pitanju jednokorisnički računar, manja mreža ili puno servera raspoređenim na velikom prostoru), zahtevima za bezbednost informacija, postoje različiti tipovi baza podataka. Jasno da ćete za realizaciju jednostavne apllikacije za fakturisanje koja radi na samostalnom računaru male firme koristiti jedan tip, a na primer, za vođenje platnog prometa velikog preduzeća ili banke koristiti sasvim drugi tip i format baze podataka. Ključna odluka koju treba doneti na samom početku planiranja sistema je upravo izbor odgovarajuće baze podataka.

Na žestokoj tržišnoj utakmici, različiti proizvođači nude svoja rešenja, u svakoj narednoj verziji sa naprednijim mogućnostima i poboljšanim perfomansama. Kada odredite tip baze koji vam je potreban, naredni korak je opredeljenje za određenog proizvođača. Ovde ne postoje neka određena pravila za donošenje odluka, jednostavno zato što je puno faktora u igri. Pomaže jedino iskustvo i konsultacije sa kolegama.

Ovde možete zapaziti da bazu podataka u širem smislu reči može predstavljati i MS Word dokument, AutoCad crtež pa čak i obična fotografija. Da bi suzili definiciju uvešćemo nov termin – **RDBMS** (Relational Database Managment System) ili sistem za upravljanje relacionih baza podataka. RDBMS je softverski proizvod koji struktuira podatke u skladu sa relacionim modelom i omogućava manipulaciju podataka na bazi relacione algebre. Korisnik je pošteđen tehničkih detalja oko načina smeštanja i čitanja podataka jer fizički aspekt njihovog manipulisanja izvodi RDBMS.

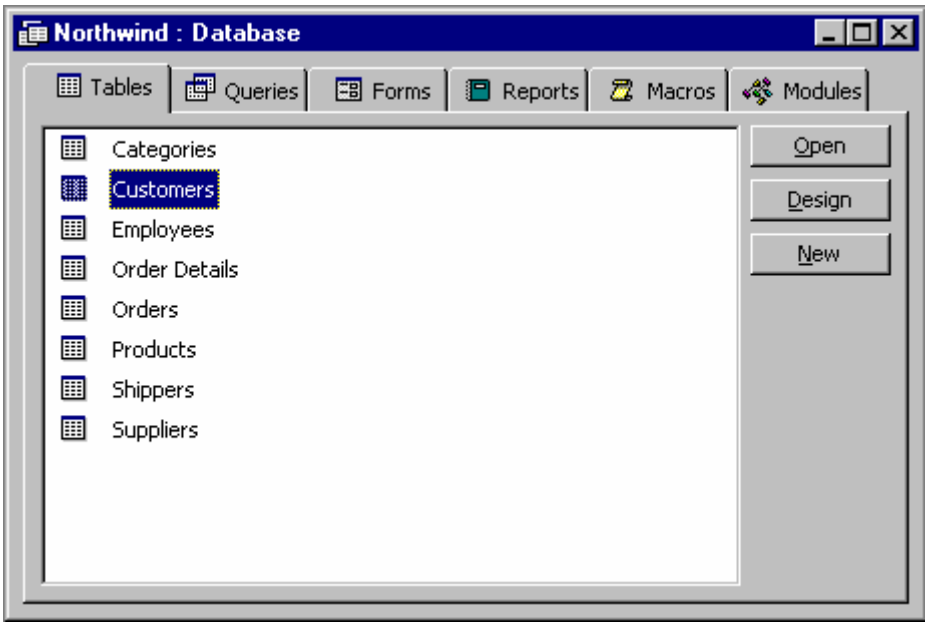
Relacioni model predstavlja podatke u vidu tabele, dakle redovi i kolone, i sastoji se od kolekcije tabela, indeksa, upita i drugih objekata. Tabele se vezuju pomoću ključnih polja različitim tipovima relacija.

Pregled podataka u redovima i kolonama:

Na slici se vide kupci iz baze podataka koji su organizovani u tabelu. Terminiloški, svaki red predstavlja podatke o jednom kupcu i takav red podataka se naziva zapis (record) ili slog. Jedan slog se sastoji od najmanje jednog polja sa podatkom. Ovo predstavlja najmanju jedinicu mere baze podataka i naziva se polje (field).
Vidimo da jednog kupca definišu četiri polja:
Sifra kupca
Kompanija
Adresa
Grad

Kažemo da se slog sastoji od četiri polja. Svi slogovi (svi kupci), čine tabelu Customers. Jedna baza podataka može imati više tabela, kao što je prikazano na slici:

Pregled svih osam tabela u bazi podataka



Na slici možete uočiti da osim tabela postoje još neki objekti u bazi podataka:

- *Queries*

- *Forms*
- *Reports*
- *Macros*
- *Modules*

Nas osim Tables, interesuju još samo Queries objekti koji će biti objašnjeni u narednim poglavljima. Ostali su specifičnost Access baze podataka.

Baza podataka je hijerarhijski organizovana:
Sama baza podataka koja najčešće predstavlja jednu datoteku na disku
Tabele u kojima su smešetene informacije. Složenija baza podataka može u sebi imati više desetina tabela. Tabela nam služi da logički grupišemo informacije. Na primer: kupci, dobavljači, fakture i slično zavisno od organizacije.
Jedan red u tabeli se naziva zapis ili slog (record) i predstavlja kompletnu informaciju date logičke celine. Jedan zapis se sastoji od bar jednog polja koje čuva odgovarajuću informaciju.

Ovakva struktura je najčešća u savremenom RDBMS –u, međutim neki stariji sistemi kao što je Dbase unekoliko odstupaju od nje, ali suština je uvek ista.

Karakteristike Access baze podataka:

- Maksimalna veličina mdb datoteke je 1 gigabajt. Međutim, imajući u vidu da Access podržava metodu vezanih (linked) tabela, koja je će biti objašnjena kasnije, ovo ograničenje se može prevazići.
- Maksimalni broj objekata (tabela i upita u našem slučaju) je 32768, što u svakom slučaju daleko prevazilazi realne potrebe.
- Svaki objekt može imati ime maksimalne dužine do 64 karaktera, uključujući i razmake.
- Maksimalna veličina jedne tabele je identična maksimalnoj veličini baze podataka, 1 gigabajt. Ovo ograničenje izgleda na prvi pogled nedostižno, međutim, pošto Access u tabeli može da čuva i objekte kao što su Word dokument, Excel tabela, AutoCad crtež ili skeniranu sliku, lako se može dogoditi da dostignete ovaj limit. Imajte u vidu da jedna kvalitetna slika može imati veličinu od jednog megabajta, što znači ograničenje od oko 1000 slika u jednoj tabeli.
- Maksimalni broj konkurentnih korisnika (korisnici koji istovremeno rade sa istom bazom podataka) je ograničen na 256.

Iz ovoga proizilazi, da je Access baza namenjena za korišćenje u manjim i srednjim firmama, sa malim do srednjim protokom i količinom podataka. Tipična upotreba su razni sistemi za fakturisanje, vođenje materijalnog i finansijskog knjigovodstva, aplikacije za obračun plata, platni promet i slično. Međutim, u Access-u možete na sličan način kao i u Visual Basic-u kreirati aplikacije sa formama, kodom, modulima i izveštajima, ali sa bitno manje programskih mogućnosti od VB-a. Da bi se ova aplikacija izvršavala, krajnji korisnik mora posedovati ili Access ili Access run-time modul. U ovom kontekstu Access se može izvanredno iskoristiti za pisanje kako samostalnih aplikacija, tako i front-end aplikacija u klijent-server okruženju kada se vrši konekcija ka eksternoj bazi podataka kao što SQL Server ili Oracle.

U svakom slučaju, programeri koji ovladaju Visual Basic-om, veoma brzo mogu savladati Access i obrnuto, imajući u vidu da je programski jezik gotovo identičan, ali je potreban donekle različit način razmišljanja pri projektovanju aplilkacija. Kod nas i u svetu je veoma česta pojava da programer paralelno koristi i kombinuje Access i Visual Basic, zavisno od konkretnih zahteva.

U svim narednim primerima u vezi Access baze podataka, radi pogodnosti će biti korišćena baza Nwind.mdb koja se automatski instalira prilikom instalacije Visual Basic-a.

Indeksi

Podaci koji se smeštaju u tabelu su što se redosleda tiče, složeni onim redom kojim su i uneti (postoji i druga mogućnost, pogledajte metodu *compactdatabase*). U realnoj situaciji, podatke prikazujemo uvek po nekom redosledu (datumu, imenu, količini i sl.). To možemo uraditi i bez korišćenja indeksa, ali uz katastrofalan uticaj na perfomanse.

Situacija je slična kao sa knjigom. Ako želite da pronađete neki pojam možete pregledati knjigu stranu po stranu ili jednostavno pogledati indeks na kraju knjige, utvrditi na kojim stranicama se nalazi potrebni pojam i direktno je otvoriti. Jasno je da je drugi način daleko brži. Za prvi način pretraživanja

kažemo da je *sekvencijalan* (negde i *table scan*), da bi pronašli određeni podatak moramo proći kroz sve podatke, a za drugi kažemo da je indeksirani.

Indeks predstavlja niz pokazivača koji govore na kojoj poziciji se nalazi stvarni podatak, odnosno slog. U praksi korišćenje indeksa znači razliku između dugog čekanja i gotovo trenutnog dobijanja traženog podatka ili redosleda sortiranja. Dakle indeksi drastično unapređuju perfomanse u slučajevima pretraživanja i/ili prikaza podataka po zadatom kriterijumu i sortiranjima.

Indeksi imaju još dve značajne uloge:
- kreiranje primarnog ključa
- uspostavljanje relacija

Poželjno je da svaka tabela poseduje primarni ključ (*primary key*). To je polje koje jedinstveno identifikuje jedan slog, ne postoje dva sloga koja imaju istu vrednost u polju koje predstavlja primarni ključ. Dobri kandidati za primarni ključ su polja koja sadrže šifru kupca ili proizvoda, broj lične karte i socijalnog osiguranja, broj fakture i slično.

Generalno postoje dva tipa indeksa:
- indeks, koji dopušta da podaci u poljima od kojih se sastoji mogu biti duplirani u tabeli,
- jedinstveni indeks koji ne dozvoljava dupliranja. Očigledno, primarni ključ poseduje jedinstveni indeks. Druga polja koja nisu primarni ključ, takođe mogu posedovati jedinstveni indeks. što je najbolji način za proveru jedinstvene vrednosti.

Oba ova tipa indeksa mogu dalje biti:
- jednostruki, indeks se sastoji od samo jednog polja
- višestruki, indeks se sastoji od dva do najviše deset polja. U slučaju da često pretražujemo ili sortiramo informacije po više kriterijuma, dobra ideja je kreirati višestruki indeks.

Pažljivi čitalac će postaviti pitanje, zašto ne postaviti indekse za sva polja u tabeli pa će svaka kasnija operacija pretraživanja i sortiranja biti ubrzana. Stvar je u tome što indeksi iako sa jedne strane ubrzavaju posao, sa druge strane ga usporavaju. Svaki put kada dodajemo, menjamo ili brišemo podatke, Access mora ažurirati i sve indekse koji su definisani u tabeli. Ovo se radi automatski, ali što ima više indeksa, potrebno je više vremena. Dakle previše indeksa usporava gore pomenute operacije.

Istina je kao i uvek na sredini. Analizom potrebnih operacija u konkretnom slučaju, potrebno je postaviti indekse za samo potrebna polja. Kasnije, uvek možemo dodati indeks ili obrisati onaj koji se ne koristi. Ovo je jednostavna operacija koja ne utiče na postojeće podatke i sigurno ćete je puno puta primenjivati u praksi.

Drugi način, koji može dobro doći je kreiranje privremenih indeksa iz same Visual Basic aplikacije. U slučaju da je zbirno vreme: kreiranje indeksa + prikaz podataka po tom indeksu + brisanje indeksa manje (a gotovo uvek jeste) od vremena potrebnog za sekvencijalno pretraživanje, ovo predstavlja odlično rešenje.

Automatizovana optimizacija

Prilikom zadavanja upita, Access automatski određuje koje indekse će koristiti od onih koji su mu na raspolaganju. Ova tehnika pod nazivom ***Rush-More***, egzistira još od Access-a 2.0 i Microsoft ju je odkupio od kompanije Fox tj, njihovog RDBMS proizvoda FoxPro, koji je u vreme kada se pojavio bio apsolutni šampion po pitanju perfomansi.

Ovaj automatizam je sa jedne strane dobro došao, zadate upit i Access ga automatski izvrši na najbolji mogući način, ali ponekad programerima nedostaje puna kontrola i nikad niste sigurni da ste najoptimalnije kreirali i iskoristili postojeće indekse. Kao i uvek u praksi, eksperimantisanje i pregled perfomansi je dobro rešenje.

Kreiranje baze podataka i njenih objekata

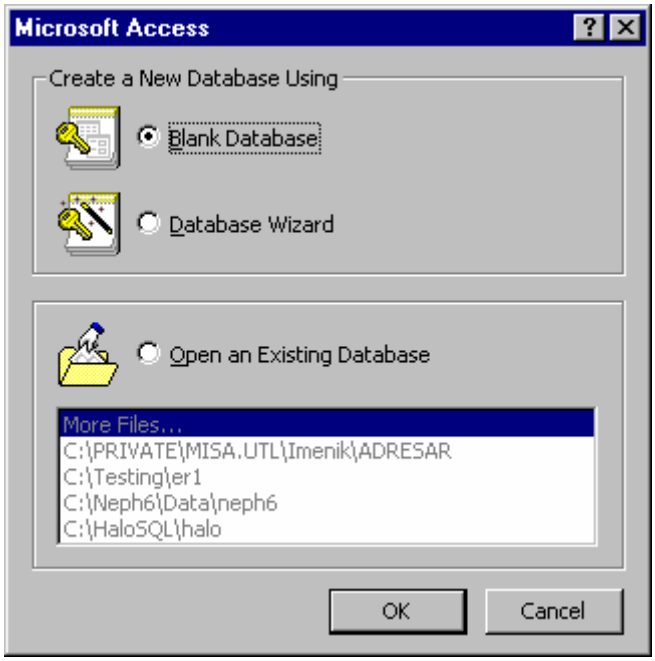
Postoji nekoliko načina kreiranja Access baze i objekata. Zavisno od ličnih preferenci, možete početi na sledeće načine:

- Koristeći Access aplikaciju. Po našem mišljenju najjednostavniji način. Nudi fleksibilnost i kasnije jednostavne intervencije na izmeni strukture podataka. Svakom Visual Basic programeru, Access je neophodna alatka i treba je imati na računaru.

- Direktno iz VB koda, koristeći ili DAO objekte ili SQL jezik. Korisno, u slučaju da iz aplikacije treba kreirati celu bazu ili neku privremenu tabelu.
- Pomoću Visual Data Manager-a, alatke koja se isporučuje uz VB i koja je uzgred i napisana u Visual Basic-u.
- Možete sami napisati svoju *custom* aplikaciju koju tačno prilagodite svojim potrebama za ove zadatke.
- Koristeći programe drugih proizvođača, na primer ErWin (Logic works), za formiranje logičke strukture baze i njeno kreiranje.

Kreiranje baze podataka u Access-u

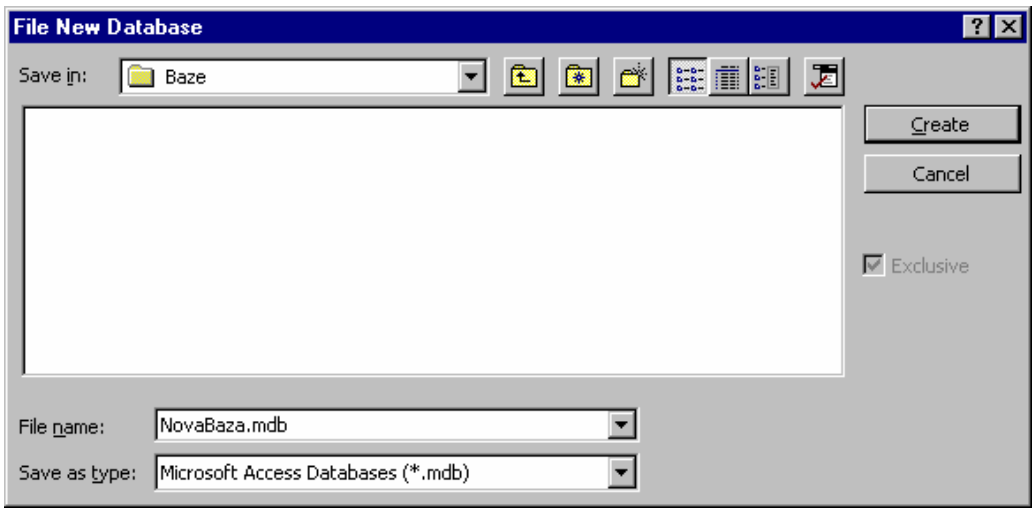
Pokrenite Access i inicijalno dobijate sledeći dijalog:



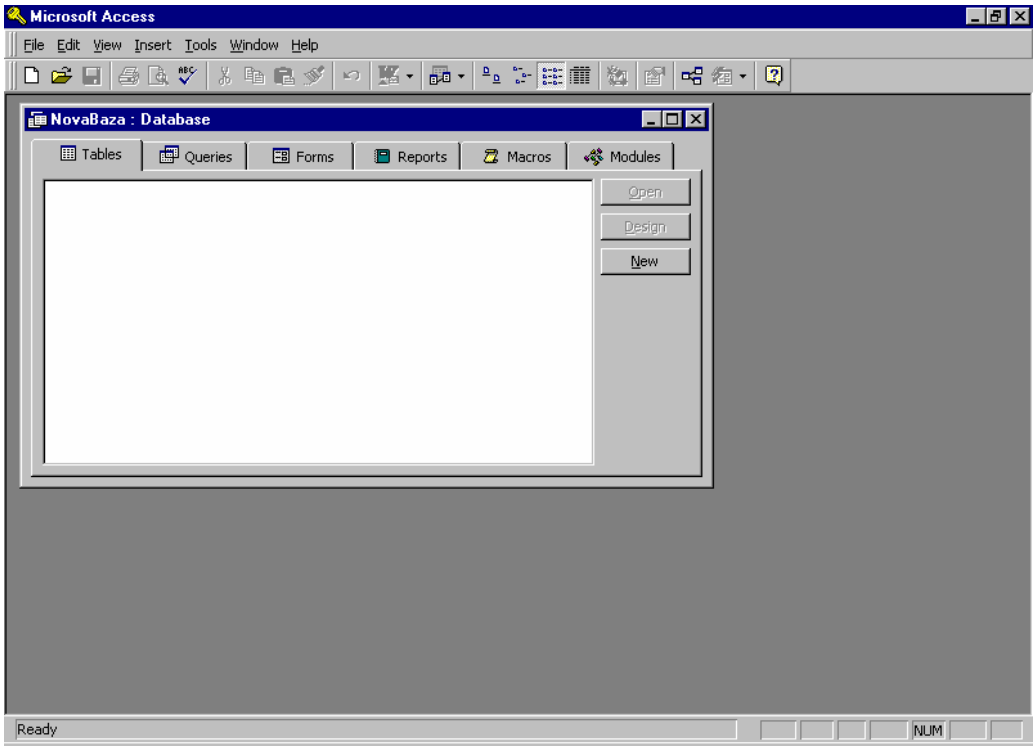
Ako želite novu (praznu) bazu podataka, potrebno je selektovati opciju **Blank Database** (kao na slici) i pritisnuti na dugme **OK**. U slučaju da želite da otvorite ranije kreiranu bazu podataka radi pregleda ili modifikacija, selektovaćete opciju **Open an Existing Database**, pa ili iz liste izabrati potrebnu ili selektovati **More Files...** koji otvara standardni dijalog za izbor datoteke.

Poslednja opcija, Database Wizard, pokreće niz dijaloga koji Vam nude šablone više predefinisanih tipova baza podataka, grupisanih po različitim oblastima delatnosti. Na osnovu vašeg izbora Access automatski kreira bazu, tabele, pa ih čak i po želji puni podacima radi testiranja. Iako na prvi pogled ova mogućnost deluje veoma privlačno, kod nas u praksi je gotovo neupotrebljiva. Svi nazivi tabela i polja su na engleskom jeziku, pa kasnija njihova modifikacija zahteva više vremena, nego da smo bazu kreirali ručno.

Dakle, selektovali smo **Blank Database**, pritisnuli taster **Ok** i time otvaramo **Save As...** dijalog u kome definišemo ime (file name) i lokaciju gde će biti kreirana nova baza:



U delu **File name** unesite naziv baze podataka, (možete izostaviti ekstenziju ***mdb***, biće automatski dodana), izaberite lokaciju i kliknite na taster **Create**. Access kreira novu bazu i posle nekoliko trenutaka je otvara:

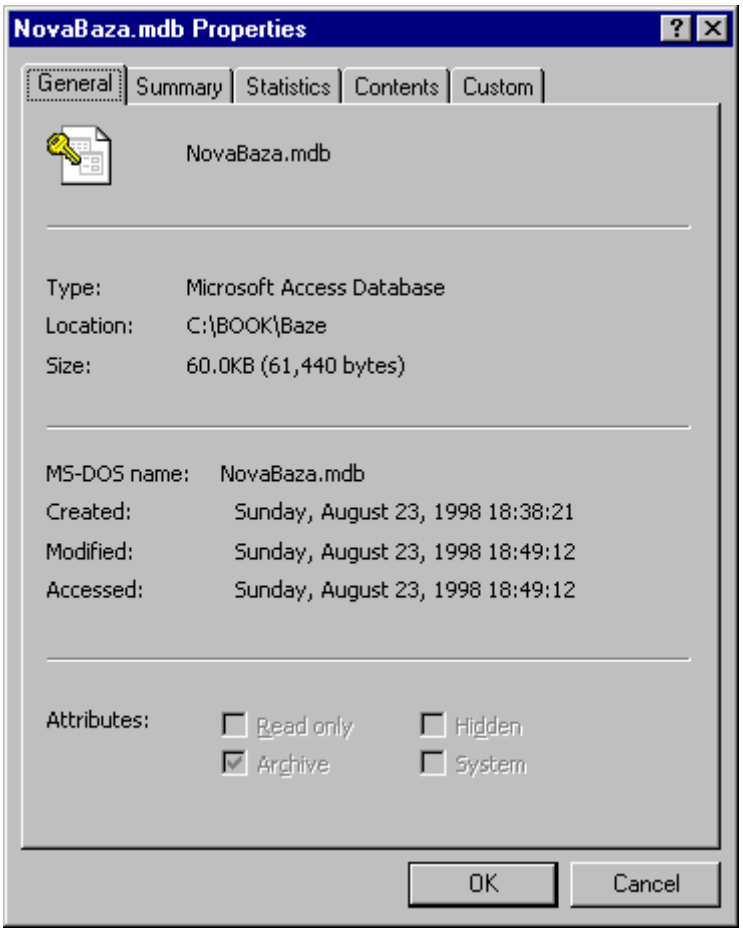


Na slici vidimo u tabbed dijalogu sve objekte baze podataka, a za nas od interesa su objekti **Tables** i **Queries**. U Access terminologiji ovaj prozor se zove *Database Window*. Pošto je baza tek kreirana, u njoj ne postoji ni jedan objekat. Omogućeno je samo dugme **New**, koje zavisno od toga koji objekat je izabran, otvara odgovarajući prozor za njegovo kreiranje.

NAPOMENA

Ako u ovom momentu u Windows Explorer-u pogledate upravo kreiranu bazu, primetićete da iako je prazna, baza ima veličinu od 60Kb. Ovo je zbog toga što Access u svakoj bazi podataka čuva i tabele sa raznim sistemskim informacijama. Ove tabele inicijalno nisu vidljive, i ne treba ih ručno menjati. Svaka tabela ima u nazivu prefiks MSys... i atribut koji govori da su u pitanju sistemske tabele. Ovo je značajno zato što analizirajući strukturu Access baze iz Visual Basica (što ćemo raditi u narednim poglavljima), ove tabele jesu vidljive i mogu prouzrokovati probleme.

Slično VB projektu, Access baza ima svojstva koja se mogu zgodno upotrebiti u praksi. Ova svojstva između ostalog daju informaciju kada je baza kreirana, kada je poslednji put modifikovana, kada joj je poslednji put pristupano i još dosta informacija koje možete sami setovati. Ova svojstva se dobijaju iz menija **File** – **Database Properties** i odnose se na otvorenu bazu. Takođe su dostupna iz Visual Basic-a:



Tabele u Access-u

Tabela je objekat koji fizički sadrži podatke, po relacionom modelu u obliku redova i kolona. Tabela se sastoji od polja, koja isto tako predstavljaju objekte, pa samim tim imaju i svojstva. Dva najbitnija svojstva su **ime** i **tip** polja.
Pomoću imena se referišemo najčešće na vrednost sadržanu u tom polju, ali i na svojstva polja. Ime polja može biti dužine do 64 karaktera, uključujući razmake i neke specijalne karaktere. U istoj tabeli svako polje mora imati različit (jedinstven) naziv. Naravno, u dve ili više različitih tabela možemo imati polje sa istim nazivom. Na primer, imena tabela mogu biti: Lista kupaca, Lager lista, Statistika prodaje i slično. Imena polja mogu biti Sifra robe, Adresa kupca ...

NAPOMENA:
Access dozvoljava razmake u nazivu polja i tabela, međutim ako kasnije želite da bazu prebacite u neki jači RDBMS, vodite računa da li on podržava razmake u nazivima. Na primer MS SQL Server ih ne podržava, pa u tom slučaju morate konvertovati imena i aplikacije koje rade sa njima.

Tip polja **(field type)** određuje koji tip podataka polje može sadržati. Zavisno od tipa, polje u tabeli zauzima određeni prostor **(field size)** na disku. U narednoj tabeli je data lista tipova, tip podataka i opseg koje može da sadrži, broj bajtova koje zauzima i paralelni tip Visual Basic varijable.

Tipovi podataka u tabeli

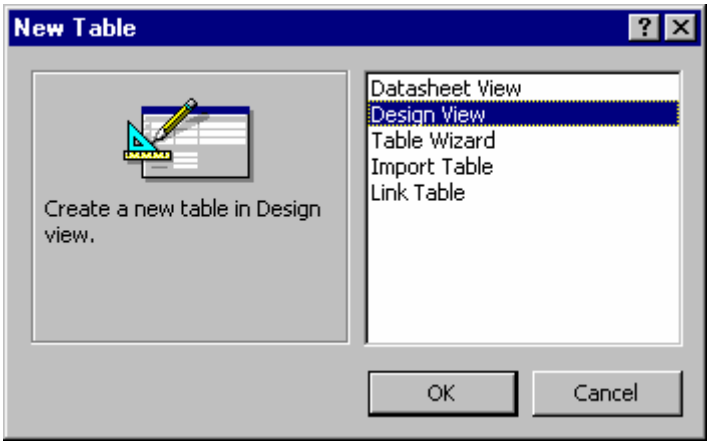
Tip	Opseg	Zauzeće (bajtova)	VB varijabla
Text	Tekst do 255 karaktera	1-255	String String * x
Memo	Tekst do 65535 karaktera (data kontrola) Tekst do 1 gigabajt (DAO pristup)	1-1 GB	String String * x
Number Byte	Celi pozitivni brojevi 0-255	1	Byte
Number Integer	Celi brojevi u intervalu -32768 do 32767	2	Integer
Number Long Integer	Celi brojevi u intervalu –2,147,483,648 do 2,147,483,647	4	Long
Number Single	Realni broj u intervalu –3.402823E38 do –1.401298E–45 za negativne i□1.401298E–45 do 3.402823E38 za pozitivne vrednosti	4	Single
Number Double	Realni broj u intervalu –1.79769313486231E308 do –4.94065645841247E–324 za negativne i 1.79769313486231E308 do 4.94065645841247E–324 za pozitivne vrednosti	8	Double
Date/Time	Datum i vreme u opsegu od godine 100 do 9999	8	Date
Currency	Valuta i realni brojevi preciznosti do 15 cifara sa leve strane decimalne tačke i 4 cifre sa desne strane. Korisno u monetarnim kalkulacijima jer ne dolazi do zaokruživanja velikih brojeva i prikaza u eksponencijalnom obliku.	8	Currency
Yes/No	Sadrži samo vrednosti True ili False.	1 bit	Boolean
OLE Object	Objekt koje se može insertovati iz bilo koje OLE Object Server aplikacija (slika, Word doc, zvuk animacija ...)	max do 1 gigabajt	Object
Autonumber	Specijalno auto inkrementirajuće polje koje automatski održava Access i koje se ne može direktno menjati. Često se koristi kao primarni ključ jer se garantuje njegova jedinstvena vrednost u okviru jedne tabele.	4	Long
Hyperlink	Tekst koji služi kao hiper link adresa. Sastoji se od tri dela: displaytext URL subaddress	max 3 x 2048 bajtova	String String * x

Neophodno je, zavisno od zahteva precizno definisati tip i veličinu svakog polja u tabeli. Kasnije se izmene mogu lako izvršiti, ali vodite računa da može doći do gubitka dela podataka ako tip polja sa većim menjate u tip sa manjim opsegom.

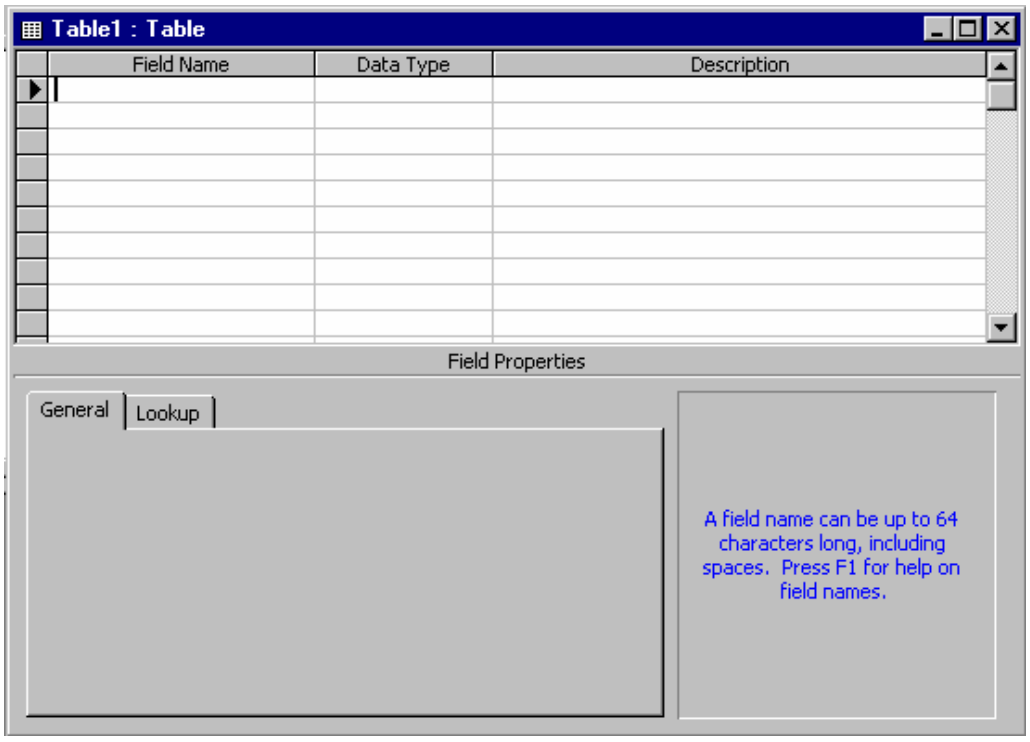
Primećujete da svaki tip polja u tabeli ima odgovarajući tip varijable u Visual Basic-u.

Kreiranje tabele:

Pokrenite Access i otvorite predhodno kreiranu bazu podataka NovaBaza.mdb, selektujte tab **Tables** i kliknite na dugme **New**, što otvara sledeći dijalog:



Izaberite **Design View** i kliknite dugme **Ok**. Otvara se glavni prozor u kome kreiramo polja jedne tabele, definišemo tipove polja i podešavamo njihova svojstva. Inicijalno, nova tabela se zove **Table*x***, gde je ***x*** redni broj tabele u bazi. U našem slučaju **Table1**, što se kasnije može izmeniti.



Opis kolona:

- Field Name: unosimo naziv polja (obavezno za unos)
- Data Type: iz combo box-a biramo tip polja (obavezno za unos)
- Description: unosimo svoj opis polja (opciono), koji nije od interesa za Visual Basic.

U donjem delu prozora sa leve strane, nalaze se dva tab-a. U **General** tab-u postavljamo svojstva polja, dok **Lookup** tab nije bitan za Visual Basic. Kao što je uobičajeno, počinjemo od kreiranja jednostavne tabele. U bazi želimo tabelu pod nazivom **Klijenti** u kojoj ćemo definisati potrebna polja i u kojoj ćemo čuvati informacije o klijentima svoje firme. Demonstracije radi, ovim poljima ćemo pokriti sve tipove podataka. Pre svega da postavimo sebi zadatak, koje informacije želimo da imamo i kog tipa su te informacije:

- Jedinstvena identifikacija svakog sloga. Nije obavezno, ali u praksi veoma korisno, jer omogućava jednoznačno pozicioniranje na potrebni slog i manipulisanje sa njim. Ovo polje bi u našem slučaju trebalo da bude transparentno za korisnika. Nema potrebe da ga opterećujemo time da mora da izmišlja jedinstvene brojeve za svaki slog u tabeli. Ovo bi trebalo da bude automatski, takođe ovo

polje obično predstavlja i primarni ključ tabele. Najbolji izbor je polje tipa **Autonumber** i nazvaćemo ga **ID Klijenta**.

- Polje koje čuva ime klijenta. Nazvaćemo ga **Ime klijenta** i imajući u vidu da može sadržati kombinovano tekst i brojeve, to će biti polje tipa **Text**. Kod tekstualnih polja potrebno je odrediti njihovu dužinu (od 1 do maksimalno 255 karaktera uključujući i razmake). Recimo da nam je dovoljna dužina **80 karaktera**. U ovakvim slučajevima potrebno je uspostaviti balans između zaista potrebne veličine i minimalizacije veličine baze. Ne treba preterivati, ako je potrebno lako ćete povećati veličinu tekstualnog polja. Takođe imajte u vidu i perfomanse. Dugačka polja znače i duža pretraživanja i sortiranja.
- Potrebno je imati informaciju od kog datuma sarađujemo sa klijentom. Ovo polje ćemo nazvati **Saradnja** i biće tipa **Date/Time**. Jasno, u Visual Basic-u koristićemo godinu sa četiri cifre, jer ne želimo problem 2000 - te.
- U tabeli ćemo voditi stalne i sporadične klijente. Potrebno je polje koje će difinisati status klijenta u tom smislu. S obzirom da status može imati samo dve vrednosti (Da ili Ne) koristićemo **Yes/No** tip podatka i to polje možemo nazvati **Stalni klijent**.
- Finansijski je od značaja imati informaciju o trenutnom saldu klijenta. Pošto je u pitanju novac upotrebićemo **Currency** tip, a polje ćemo nazvati **Saldo**.
- Vodićemo broj realizovanih ugovora sa klijentom. To polje treba biti numeričkog tipa, celobrojno. Dovoljno je uzeti **Number - Byte** (opseg od 0-255), a ako nije dovoljno možemo uzeti sledeći tip po opsegu Number – Integer (do 32767). Ovo polje ćemo nazvati **Ugovori**.
- Sve ostale informacije koje nismo predvideli i koje mogu biti različitih tipova, standardno se smeštaju u polje pod nazivom **Komentar** koje je tipa **Memo**.
- Na kraju, demonstracije radi, u tabelu možemo smestiti i opštu informaciju o klijentu. To može biti njegova slika, logotip, neki dokument ili bilo koji drugi OLE objekt. Ovo polje ćemo nazvati **Informacije** i biće tipa **OLE Object**.

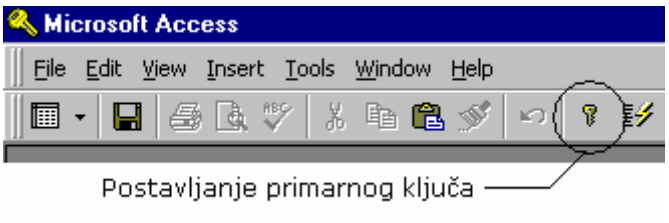
Svojstva polja

Pre nego što pređemo na kreiranje tabele, moramo upoznati svojstva polja. Sledi pregled svojstava koja su od značaja prilikom pristupa bazi iz Visual Basic-a.

- **Field Size** (veličina polja)
  dvojako svojstvo:
  U slučaju polja tipa Text, određuje maksimalni broj karaktera koje može sadržati (1-255);
  U slučaju polja tipa Number, određuje tip broja (Byte, Integer, Long Integer, Single ili Double). Ovim je indirektno određena i veličina polja.
- **Default value** (inicijalna vrednost)
  svojstvo koje određuje inicijalnu vrednost polja prilikom dodavanja novog sloga u tabelu. Može biti konstantna vrednost ili funkcija koja vraća vrednost. Ako na primer za polje **Saradnja** u našoj tabeli, postavimo ovo svojstvo na *Date()*, prilikom dodavanja novog sloga, polje će automatski dobiti vrednost današnjeg datuma. Neophodno je voditi računa da tip podataka polja i konstantna vrednost ili vrednost koju vraća funkcija budu isti. Ovo svojstvo nije restriktivno. U pitanju je samo ponuđena vrednost, koju korisnik može izmeniti.
- **Validation rule** (pravilo validacije)
  Restriktivno svojstvo. Zadaje se uslov koji vrednost uneta u polje mora zadovoljavati, u suprotnom slog neće biti ažuriran (snimljen ili izmenjen). Na primer za isto polje Saradnja možemo odrediti da uneti datum početka saradnje bude jednak ili raniji današnjem datumu. U tom sllučaju Validation rule bi postavili na *<= Date()*. Sada svaki uneti datum mora biti manji ili jednak današnjem, tj. datumi noviji od današnjeg neće biti prihvaćeni.
- **Validation text** (tekst validacije)
  Svojstvo koje ima smisla samo ako je postavljeno **Validation rule** svojstvo. U slučaju da pravilo validacije nije zadovoljeno, u dijalogu za poruke će biti prikazan tekst koji je unet u **validation text** svojstvu. Kod našeg datuma možemo postaviti ovo svojstvo na *Nisu dozvoljeni datumi veći od današnjeg*. U Visual Basicu, neodgovarajuća vrednost generiše grešku u vreme izvršavanja, koja se može uhvatiti standardnom *On Error* konstrukcijom. Tada *Err.Description* sadrži tekst definisan u Validation text svojstvu.
- **Required** (obavezno za unos)
  Ovo svojstvo može imati vrednosti Yes ili No, (tj. True ili False). Inicijalno je postavljeno na No, i

postoji kod svih tipova polja osim kod Autonumber i OLE Object. Ako je postavljeno na Yes, dato polje je obavezno za unos, slog se ne može ažurirati ako u polju ne postoji vrednost. Ne postojećom vrednošću se smatra NULL vrednost u polju, osim u polju tipa text u slučaju da je svojstvo Allow Zero Length postavljeno na Yes. Dakle ako u numeričkom polju unesete vrednost 0, smatra se da polje ime vrednost i da je uslov Required zadovoljen.

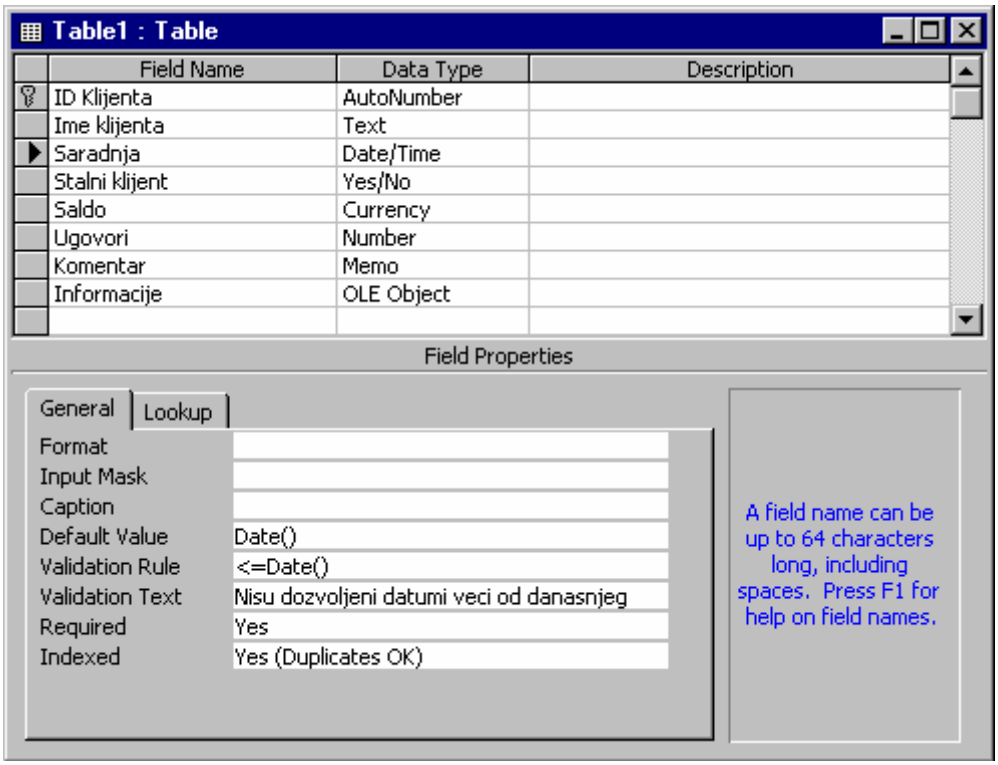
- **Allow Zero Length** (dozvoljena NULL dužina)
  Svojstvo koje je dostupno samo kod polja tipa Text i Memo. Određuje da li je dozvoljena NULL vrednost (Yes) ili ne (No). Koristi se u saradnji sa **Required** svojstvom.
  Ako je Required = Yes, Allow Zero Length = No, polje ne sme imati NULL vrednost niti sme imati samo razmake u sebi.
  Ako je Required = Yes, Allow Zero Length = Yes, polje ne sme imati NULL vrednost ali sme imati razmake u sebi.
  Prilikom ažuriranja Text ili Memo polja, vrši se automatski RTrim, tj. uklanjaju se krajnji razmaci.
- **Indexed** (Indeksirano)
  Određuje da li će biti kreiran indeks nad poljem. Indeks ima isti naziv kao i polje. Moguće su tri opcije:
  No: nema indeksa, inicijalna vrednost
  Yes (Duplicates OK): kreiran je indeks koji dozvoljava duplirane vrednosti
  Yes (No Duplicates): kreiran je jedinstveni indeks koji ne dozvoljava duplirane vrednosti.
  Indekse ne možete kreirati nad poljima tipa Memo i OLE Object. Kreirani indeksi su uvek jednostruki, tj. sastoje se od jednog polja. Access naravno dozvoljava i višestruke indekse, ali o tome kasnije.
- **Primary key** (primarni ključ)
  U jednoj tabeli može postojati samo jedan primarni kjuč, mada se on može sastojati od više polja. Nad primarnim kjučem je uvek kreiran jedinstveni indeks (Indexed: Yes (No Duplicates)), pod iniciajlnim nazivom PrimaryKey, koji se može kasnije promeniti. Ovo svojstvo se ne postavlja na istom mestu kao druga svojstva. Ovo radimo tako što prvo selektujemo odgovarajuće polje i potom u toolbar-u kliknemo na dugme Primary Key (ikona žutog ključa) na slici:



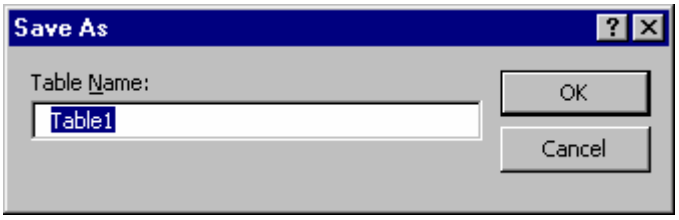
Sada možemo kreirati plan definicije tabele Klijenti. Ovo neka bude bezrezervna praksa. Struktura baze podataka je temelj aplikacije. Ako se pogrešno realizuje, a Visual Basic aplikacija je dostigla visok stepen realizacije, gotovo svaka izmena u strukturi baze podataka za sobom povlači velike izmene u VB aplikaciji.

Naziv polja	**Tip - Veličina**	**Svojstva**
ID Klijenta	AutoNumber	Primary Key
Ime klijenta	Text – 80	Required: **Yes** Allow Zero Length: **No** Indexed: **Yes (Duplicates OK)**
Saradnja	Date/Time	Default value: **Date()** Validation rule: **<= Date()** Validation text: **Nisu dozvoljeni datumi veci od danasnjeg** Required: **Yes** Indexed: **Yes (Duplicates OK)**
Stalni klijent	Yes/No	Default value: **True**
Saldo	Currency	Default value: **0**
Ugovori	Number – Byte	Default value: **0**
Komentar	Memo	
Informacije	OLE Object	

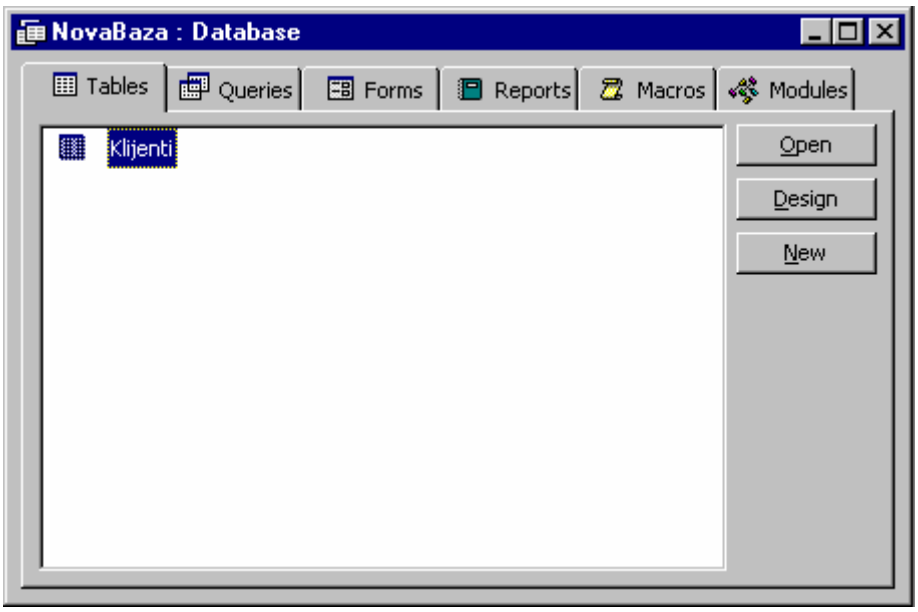
Sada možete redom kreirati potrebna polja. Na kraju tabela bi trebala da izgleda kao na slici (fokus na polju Saradnja):



Na kraju potrebno je snimiti definiciju tabele. Meni **File** – **Save** ili *Ctrl-S*. Pošto tabela još uvek nema svoje ime, dobijamo prompt za unos imena, sa inicijalnim imenom Table1.



Otkucajte ime **Klijenti** i kliknite na dugme **OK**. Ovim je kreiranje tabele završeno. Naziv tabele se sada vidi u DataBase prozoru Access-a. Za sada u njoj nema podataka, ali ćemo u narednom poglavlju, iz Visual Basic-a kreirati projekt koji će koristiti upravo ovu bazu podataka i tabelu Klijenti.



Pristup Access bazi iz Visual Basic-a

Data kontrola

Data kontrola je jedna od složenijih kontrola Visual Basic-a, služi za pristup različitim bazama podataka. Postoji od VB-a 3, i bar po Microsoft reklamnom sloganu omogućava manipulaciju bazom bez imalo kodiranja, što je delimično tačno. Automatizam koja ona nudi omogućava zaista najbrži i najlakši način za kreiranje formi za pregled podataka, međutim taj automatizam može ponekad i smetati. Takođe, postavlja se pitanje perfomansi, jer je pristup pomoću Data kontrole najsporiji. Može biti primerena za desk top baze podataka kao što je Access, koje ne podržavaju klijent – server arhitekturu. Nalazi se u osnovnom setu kontrola i ne može se ukloniti iz programa.

Ključna svojstva data kontrole

Data kontrola poseduje tri ključna svojstva za konekciju prema bazi podataka.

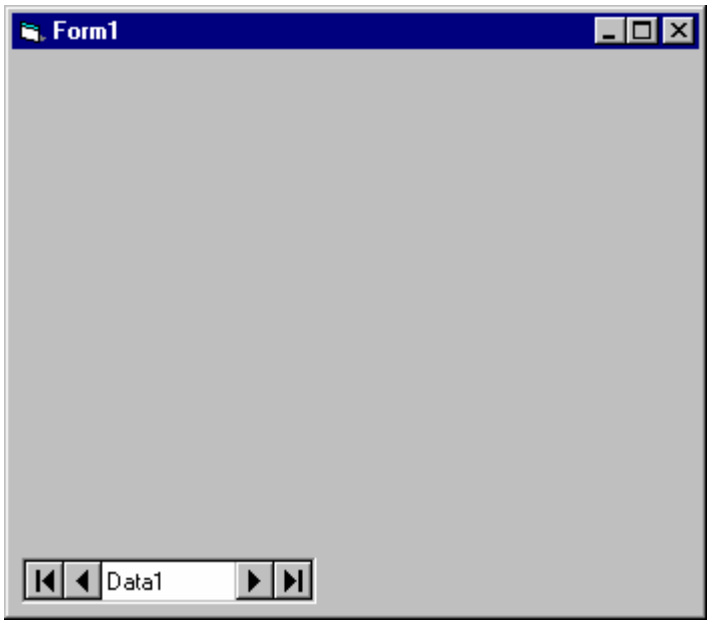
Connect (Konekcija)
Svojstvo tekstualnog tipa koje određuje tip baze podataka. Na raspolaganju su sledeće mogućnosti koje su izlistane u prozoru svojstava Data kontrole:
Access (default)
DBASE III, IV, 5.0
Excel 3.0, 4.0, 5.0, 8.0
FoxPro 2.0, 2.5, 2.6, 3.0
Lotus WK1, WK3, WK4
Paradox 3.x, 4.x, 5.x
Text

DataBaseName (Ime baze podataka)
Svojstvo tekstualnog tipa koje daje punu naziv baze podataka na koju se vezujemo. Zavisno od tipa baze podataka može sadržati samo direktorijum gde se nalaze podaci (za na primer DBASE format). Sa desne strane ovog svojstva postoji komandno dugme koje otvara standardni Open dijalog za izbor datoteke.
RecordSource (Izvor podataka)
Svojstvo, takođe tekstualnog tipa, koje definiše (u slučaju Access-a) nešto od sledećih mogućnosti:
Ime tabele iz baze koja je postavljena u DataBaseName svojstvu;
Ime snimljenog upita iz baze
Tekst SQL upita koji se zadaje direktno iz Visual Basic-a

Sva ova tri svojstva je moguće izmeniti za vreme izvršavanja programa, što nam otvara mogućnosti za laku izmenu parametara i run time izmene tipa baze, same baze podataka i izvora podataka iz te baze.

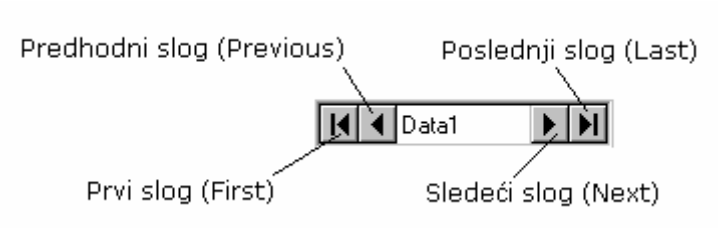
Sada ćemo korak po korak postaviti ova svojstva kako bi se povezali sa našom bazom podataka.

- Pokrenite Visual Basic i izaberite tip projekta Standard Exe.
- Postavite Data kontrolu na inicijalnu formu, kao što je prikazano na slici:



- Selektujte data kontrolu i otvorite njen prozor svojstava
- Za svojstvo Connect ostavite inicijalnu vrednost *Access*
- Za svojstvo DataBaseName ukucajte ili izaberite iz Open dijaloga lokaciju i ime baze podataka NovaBaza.mdb. Zavisno od toga gde ste smetili bazu na vašem disku vrednost ovog svojstva može na primer biti: *C:|BOOK|Klijenti|NovaBaza.mdb*
- U svojstvu RecordSource otvorite combo box, i ako je sve u redu postavljeno u tačkama 4 i 5, trebali bi da u listi vidite i selektujete tabelu Klijenti. Dakle, vrednost ovog svojstva je *Klijenti*
- Pre nego što snimite projekt postavite još neka svojstva:
  svojstvo **Name** Data kontrole postavite na *datKlijenti*
  svojstvo **Name** forme postavite na *frmKlijenti*
  svojstvo **Caption** forme postavite na *KLIJENTI*
  U Windows Explorer-u kreirajte direktorijum Klijenti i u njemu snimite ceo projekt, prihvatajući ponuđene nazive za formu i projekt.

Data kontrola, kao korisnički interfejs poseduje ćetiri komandna dugmeta koja služe za navigaciju kroz slogove. Funkcija ovih dugmadi je prikazana na slici:



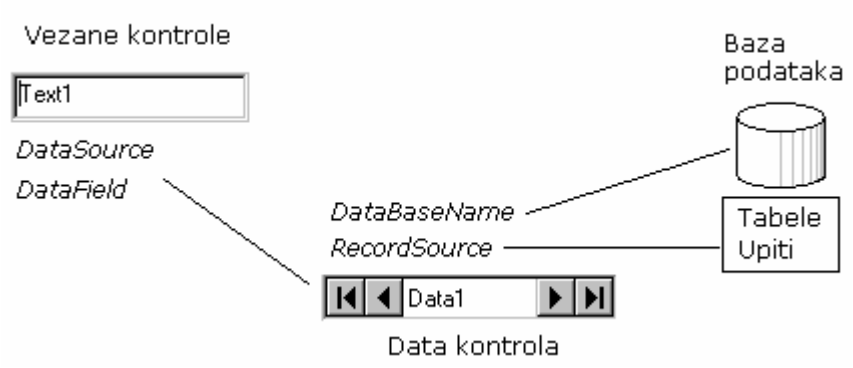
Kada bi startovali projekt, Data kontrola bi funkcionisala i pritiskom na odgovarajuću dugmad bi se zaista kretali kroz tabelu Klijenti, međutim, u ovoj fazi se ne vide podaci iz sloga. Da bi imali mogućnost pregleda i izmena podataka moramo uvesti novi termin, **vezane (bound) kontrole**.

Vezane kontrole imaju mogućnost vezivanja za Data kontrolu i prikaz vrednosti odgovarajućeg polja iz baze odnosno tabele zavisno od konekcije data kontrole. Po ovom pristupu imamo tri člana koja su međusobno povezana:

1. Baza podataka

2. Data kontrola koja se svojstvom DataBaseName vezuje za bazu podataka i pomoću svojstva RecordSource vezuje za tabelu ili upit u toj bazi.
3. Vezane kontrole koje se vezuju za data kontrolu i koje prikazuju vrednost određenog polja iz izvora podataka data kontrole. Dva svojstva vezanih kontrola su od značaja za uspostavljanje veze ka data kontroli:
   DataSource: određuje ime data kontrole (isključivo na istoj formi), za koju vezujemo kontrolu.
   DataField: određuje ime polja iz izvora podataka vezane data kontrole čiju će vrednost prikazivati vezana kontrola.

Da bi bolje shvatili vezu pogledajte narednu sliku u kojoj su prikazana sva tri člana sa svojim odgovarajućim svojstvima.



Vezane kontrole se koriste za prikaz konkretnih podataka. Praktično svaka kontrola koja poseduje svojstva **DataSource** i **DataField** ili samo **DataSource** se naziva vezana kontrola. Zavisno od samog tipa kontrole i načina upotrebe, vezujemo je za određeni tip polja iz tabele ili upita. Svaka izmena u vezanoj kontroli se reflektuje u bazi i obrnuto. Sledi lista vezanih kontrola, koje smo gotovo sve do sada upoznali sa opisom moguće upotrebe:

Lista bound kontrola

Label
Služi za prikaz polja tipa Text, Number, Currency, Date/Time, Autonumber. S obzirom da se vrednost u labeli ne može interaktivno menjati, najčešća upotreba je za prikaz polja koja su samo za čitanje;
Text box
Isto kao za Label kontrolu, ali korisnik sada može menjati vrednost u samoj kontroli (a samim tim i u bazi podataka). Po potrebi ovo možete onemogućiti postavljanjem svojstva Locked na True. Ako je koristite za prikaz Memo polja iz baze, obično se postavljaju svojstva: Multiline = True i ScrollBars=2 – Vertical;
Check box
Vezuje se za polje tipa Yes/No;
Combo, List box
Isto kao za Text box, sa mogućnošću da izbor iz liste prebacimo u bazu podataka;
Picture box, Image
Vezuju se za polja tipa OLE Object i služe isključivo za prikaz slika u formatima koje prepoznaju;
OLE kontrola
Takođe se vezuje za polja tipa OLE Object i prikazuje bilo koji validni objekt iz baze podataka

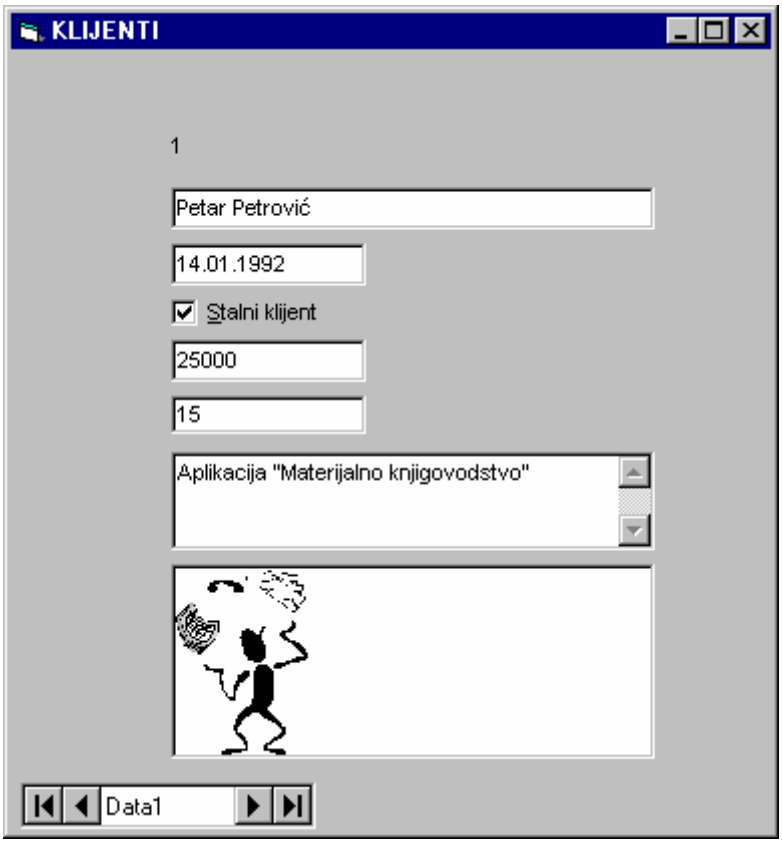
Osim ovih vezanih kontrola koje se nalaze u standardnom VB setu kontrola postoje dodatne kontrole za rad sa bazam podataka koje se dodaju pomoću *Components* dijaloga. O ovim kontrolama će biti govora kasnije.
Sada možemo da se vratimo našem projektu, rasporedimo odgovarajuće kontrole i postavimo njihova svojstva. Sledi lista potrebnih kontrola, i njihovih svojstava koje treba postaviti na formu *frmKlijenti*. Svojstvo DataSource svih kontrola treba postaviti na ime jedine Data kontrole u formi, tj. na datKlijenti.

Kontrola	Name	DataField	Svojstva
Label	lblIDKlijenta	ID Klijenta	-
TextBox	txtImeKlijenta	Ime klijenta	MaxLength = 80

TextBox	txtSaradnja	Saradnja	-
CheckBox	chkStalniKlijent	Stalni klijent	Caption = Stalni klijent
TextBox	txtSaldo	Saldo	-
TextBox	txtUgovori	Ugovori	-
TextBox	txtKomentar	Komentar	Multiline = True ScrollBars = 2 – Vertical
OLE	oleInformacije	Informacije	-

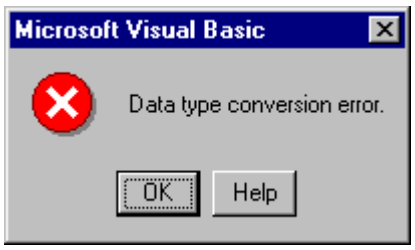
NAPOMENA:
Obratite pažnju na TextBox txtImeKlijenta. Svojstvo MaxLength je postavljeno na 80 karaktera zato što je polje u tabeli Klijenti definisano kao Text polje sa svojstvom FieldSize = 80. Ovim je onemogućeno korisniku da unese više karaktera nego što je definisano u polju za koje je TextBox vezan, što bi prouzrokovalo run-time grešku. Obavezno praktikujte za sva Text polja u tabeli.

Kada rasporedite sve kontrole, postavite svojstva i startujete aplikaciju, treba da dobijete prozor kao na slici:

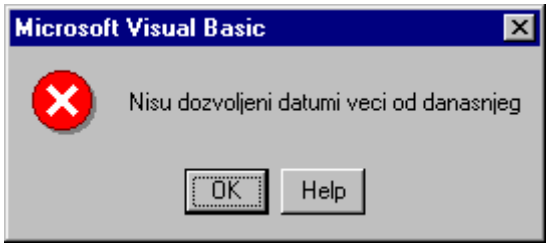


Sada se, kada kliknete na dugmad Data kontrole, vide podaci iz tabele. Na ovaj način omogućeno je jednostavno kretanje kroz slogove. Takođe možete menjati postojeće podatke. Pokušajte da izmenite, na primer, ime klijenta. Ako se sada pomerite sa ovog sloga u bilo kom pravcu i opet vratite na njega, videćete da su izmene uvažene, tj. da su zaiste snimljene u bazu podataka.

Ovim smo došli do veoma važne osobine Data kontrole. Svaka izmena u bilo kom polju sloga će biti uzeta u obzir i Data kontrola će pokušati da je snimi u bazu podataka bilo u momentu pozicionaranja na drugi slog ili kada zatvorimo formu. Kažemo pokušati, jer u slučaju da vrednost koju ste uneli nije validna, biće prijavljena poruka o grešci i izmene neće biti uvažene. Pokušajte da izmenite datum u neki koji nije korektan, na primer u 20.20.1998, i pozicionirajte se na neki drugi slog. Dobijate poruku o grešci:



Naravno u realnoj aplikaciji nećemo dozvoliti ovakve poruke, već ćemo u kodu presresti greške i odgovarajuće reagovati. U svakom slučaju Data kontrola nije snimila pogrešni podatak i ostavlja nas na slogu koji je problematičan.
Sada u isto polje unesite datum koji je veći od trenutnog i pokušajte da se pozicionirate na neki drugi slog (tj. da naterate Data kontrolu da snimi nov podatak). Iako je datum tehnički ispravan, pojavljuje se sledeća poruka:



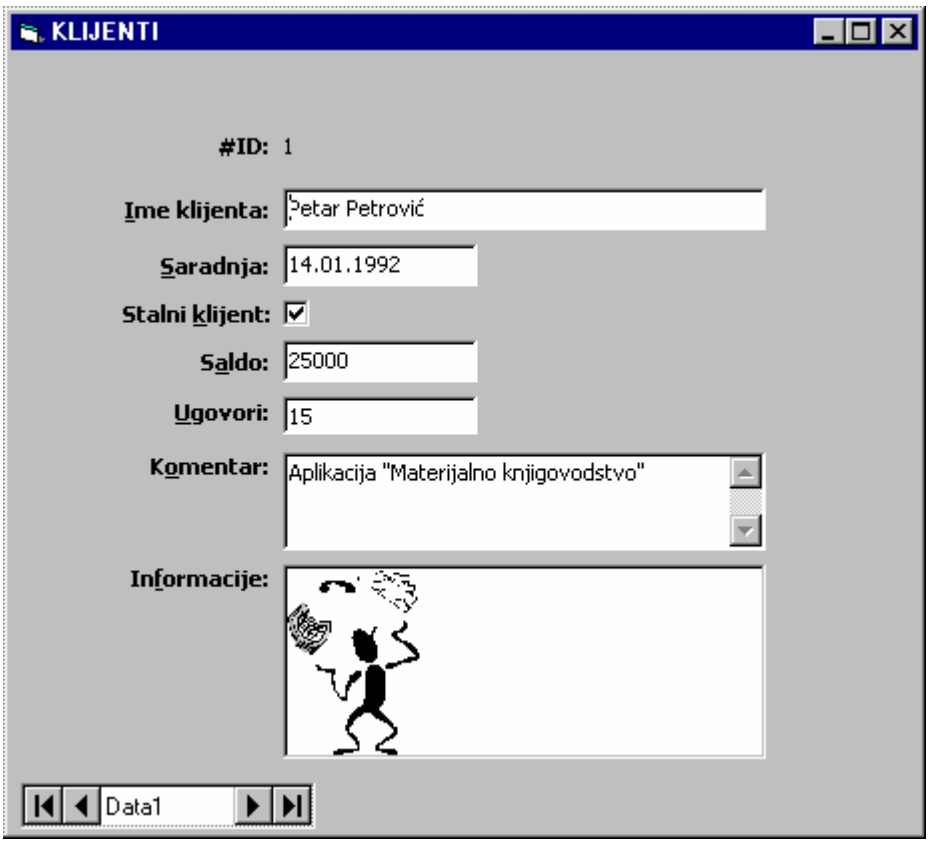
Sada se setite da smo za svojstvo **Validation Rule** datumskog polja **Saradnja** u Access-u postavili da datum mora biti manji ili jednak današnjem (*<= Date()*), a tekst koji se pojavljuje je upravo tekst koji smo definisali u **Validation Text** svojstvu istog polja.

VAŽNO:

Veoma je bitno znati i razumeti da sva svojstva polja tabele baze podataka, koja su pomenuta u predhodnom delu, ostaju u važnosti kada Visual Basic vezujemo na Access bazu. Ovo je takođe značajno kod postavljanja relacija između tabela. Iz ovoga se ponovo zaključuje da je inicijalni dizajn baze podataka i odgovarajuće postavljena svojstva različitih objekata baze fundament svake ozbiljne aplikacije.

Koncept nije nov, odavno je zastupljen u drugim značajnim RDBM sistemima, pa i u Access-u počev od prve verzije 1.0 Činjenica je samo da se često previdi značaj. Validacija podataka koji se unose u bazu je uobičajena stvar, međutim, na ovaj način dobar deo provere podataka možemo ugraditi u samu bazu, čime rasterećujemo VB aplikaciju, i obezbeđujemo da su samo ispravni podaci i veze zaista snimljeni. Logički korumpirana baza podataka može postati vaša noćna mora, jer ju je teško, a ponekad i nemoguće oporaviti.

Sada je ostalo da doteramo formu, potrebne su nam Label kontrole čija je funkcija samo objašnjenje vrednosti u tekstualnim poljima. One nisu vezane za bazu podataka, predstavljaju statičan tekst. Treba voditi računa o Hot-key kombincijama i redosledu tj. Tab order svojstvu, kako bi Hot-key kombinacije korektno funkcionisale. Konačni izgled forme je prikazan u narednoj slici:



Još uvek nedostaje mogućnost za obavljanje osnovnih operacija:

- unošenje novog sloga
- brisanje postojećeg
- pretraživanje po jednom ili više kriterijuma
- standardna Undo operacija

Da bi ih realizovali, potrebno je proučiti svojstva i metode same Data kontrole, kao i njenog podobjekta pod nazivom Recordset.

Svojstva Data kontrole:

Data kontrola poseduje standardni set svojstava koje smo imali prilike da upoznamo ranije:

Align
Appearance
BackColor
Caption
Enabled
Font
ForeColor
Height
Index
Left
MouseIcon
MousePointer
Tag
ToolTipText
Top
Visible
Width

Sva pobrojana svojstva imaju istu namenu kao i do sada. Svojstva specifična za Data kontrolu su:
Connect ; određuje tip baze podataka za koju se vezuje
DataBaseName ; Ime baze podataka (u nekim slučajevima može biti i direktorijum gde se podaci nalaze ili može biti prazan string, zavisno od Connect svojstva;
RecordSource; određuje izvor podataka iz izabrane baze podataka. Izvor može biti ime tabele, snimljenog upita ili direktno unet SQL upit kao tekstualno svojstvo
ReadOnly; Ako je postavljeno na True baza podataka se otvara samo za čitanje kada je moguće samo pregledati postojeće podatke bez mogućnosti izmena. Ako je ovo svojstvo postavljeno na False (inicijalna vrednost), onda su dozvoljene izmene nad podacima. Korisno kada želite da obezbedite podatke od korisnika, kome je na primer, potreban samo izveštaj ili neka statistika.
Exclusive; Bazu podataka generalno možete otvoriti u dva režima:
Eksluzivnom (Exclusive), gde samo vi možete manipulisati sa njom. Ostali korisnici neće moći da je otvore sve dok je vi ne zatvorite. Na ovaj način ste sigurni da ste jedini korisnik baze u mreži.
Deljenom (Shared), kada više korisnika može istovremeno otvoriti i manipulisati bazom.
Svojstvo Exclusive definiše ova dva načina, i inicijalno je postavljeno na False, dakle omogućen je istovremeni rad više korisnika.
Situacije u kojima ima potrebe otvoriti bazu ekskluzivno (Exclusive=True) su uglavnom administrativne prirode. Svaka izmena definicije objekata u bazi (na primer izmena tipa polja, dodavanje novih polja ili brisanje postojećih u definiciji tabele), zahteva da baza bude otvorena u eksluzivnom (postoji i termin "Single User") režimu.
RecordsetType; svojstvo ima tri moguće vrednosti i definiše tip izvora podataka. Vezano je za Recordsource svojstvo:
Table; Recordsource svojstvo može biti samo ime tabele. Ako želite da ažurirate podatke iz samo jedne tabele ovo je dobar izbor jer se baza podataka i tabela brže otvaraju;
Dynaset; inicijalno svojstvo. Za recordsource je moguće izabrati ime tabele, snimljenog upita ili direktno uneti SQL upit;
Snapshot; isto kao Dynaset, osim što su podaci samo za čitanje (bez obzira kako postavite ReadOnly svojstvo). Ovde je neophodno pružiti dodatna objašnjenja u vezi kreiranja Dynaset-a i Snapshot-a.
Dynaset se kreira dinamički i predstavlja niz pokazivača u memoriji koji pokazuju na aktuelne slogove u bazi podataka. Kada se krećete po Dynaset-u, potrebno je u memoriju sa diska (lokalnog računara ili servera), učitati podatke.
Snapshot je statički objekat. Aktuelni podaci su u memoriji, a ne pokazivači na njih. Rezultat ovoga je brži prikaz podataka jer ih nije potrebno učitavati sa diska. Takođe, Snapshot posle kreiranja gubi sinhronizaciju sa aktuelnim podacima, pa zbog ovoga i jeste Read Only. Snapshot koristite za statičke prikaze podatak radi izveštaja, statistika i slično. Vodite računa da, pošto se Snapshot kreira u memoriji, ne preterate sa količinom podataka u njemu. U tom slučaju slogovi se prebacuju u swap datoteku, pa se gubi prednost brzog čitanja podataka.
BOFAction; Svojstvo koje definiše kako se Data kontrola ponaša kada se pozicioniramo na prvi slog (BOF – Begin Of File), pa pokušamo da odemo na jedan pre njega (nepostojeći prazan slog). Postoje dve mogućnosti:
MoveFirst; Data kontrola nas vraća na prvi slog i flag BOF recordset-a postavlja na False. O BOF svojstvu čitajte kod prikaza Recordset objekta. MoveFirst je inicijalno postavljeno

EOF Action: - Move Last
- EOF
- AddNew

Options: (za napredno korišćenje data kontrole)

DbDenyWrite	1	U višekorisničkom okruženju, ostali korisnici ne mogu menjati Recordset.
DbDenyRead	2	U višekorisničkom okruženju, ostali korisnici ne mogu čitati slogove.
DbReadOnly	4	Nije moguće izmeniti slogove u Recordset-u.
DbAppendOnly	8	Možete dodavati slogove ali ne možete čitati i menjati postojeće.
DbInconsistent	16	Ažuriranje će se izvršiti i pored narušavanja konzistentnosti baze podataka.
DbConsistent	32	(Default) Ažirira samo ona polja koja ne narušavaju konzistentnost baze.
DbSQLPassThrough	64	SQL izraz se direktno prosleđuje ODBC izvoru podataka (SQL Server ili Oracle).

DbForwardOnly 256 Recordset može da prihvati samo MoveNext metodu. Ova opcija ne može biti korišćena nad Recordset objektima kojima se manipuliše preko Data kontrole.
DbSeeChanges 512 Generiše grešku ako drugi korisnik menja bazu koji editujete.

Metode data kontrole:

- UpdateRecord
- UpdateControls
- Refresh

Događaji data kontrole:

- Reposition
- Validate

Recordset objekat:

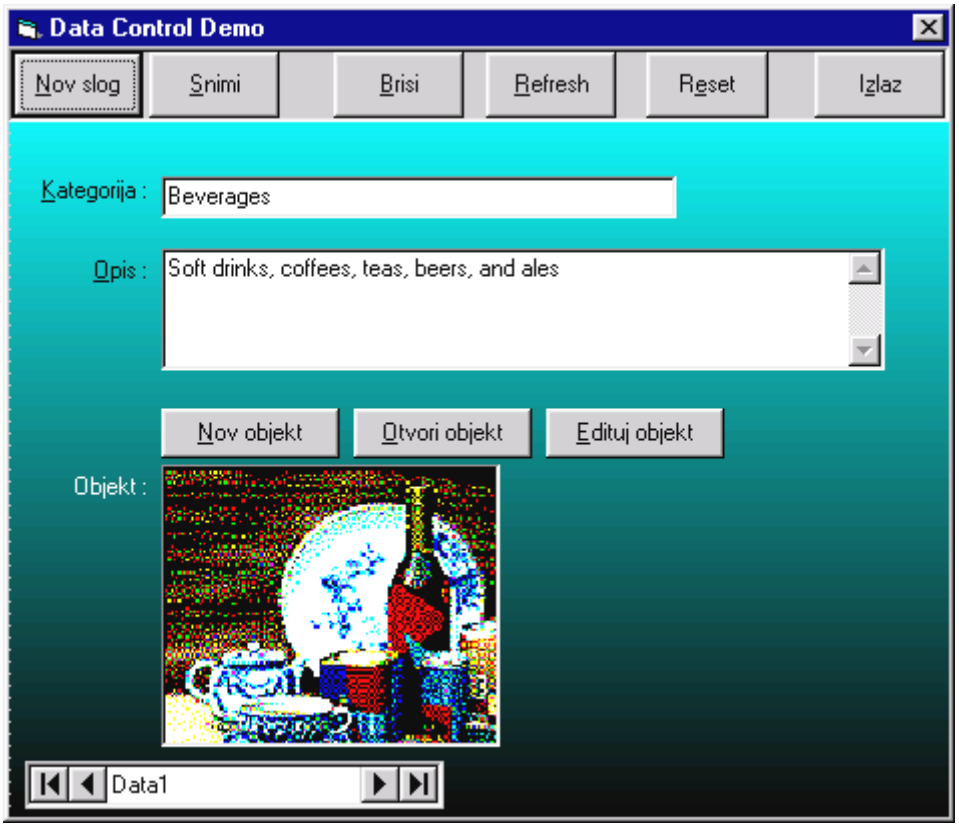
Metode:

- AddNew (nov slog u bazi)
- Delete (brisanje aktivnog sloga)
- MoveNext, MoveFirst, MoveLast, MovePrevious (pomeranje na sledeći, prvi, poslednji, predhodni respektivno)
- FindFirst, FindNext (nalaženje prvog i sledećeg sloga po kriterijumu)

Osobine:

- EOF (True ako je na kraju podataka)
- BOF (True ako je na početku podataka)
- RecordCount (ukupno slogova u recordset-u)
- AbsolutePosition (pozicija aktivnog sloga)
- Bookmark (označavanje aktivnog sloga)
- LastUpdated (bookmark poslednjeg snimljenog sloga)

Primer – pristup postojećoj bazi podataka:
Forma za pristup bazi podataka i standardne operacije sa slogovima, koristimo nwind.mdb, postojeću bazu podataka i njenu tabelu Categories.



KOD:

```
Private Sub bNovo_Click()
    On Error GoTo ErrNovo
    Data1.Recordset.AddNew
    Text1.SetFocus

NovoIzlaz:
    Exit Sub

ErrNovo:
    MsgBox "Greska ! " & Err.Description, vbCritical, "Data Control Demo"
    Resume NovoIzlaz
End Sub
```

```
Private Sub bSnimi_Click()
    On Error GoTo ErrSnimi
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified

SnimiIzlaz:
    Exit Sub

ErrSnimi:
    MsgBox "Greska ! " & Err.Description, vbCritical, "Data Control Demo"
    Resume SnimiIzlaz
End Sub
```

```
Private Sub bBrisi_Click()
    On Error GoTo ErrBrisi
    If MsgBox("Obrisati slog", vbQuestion + vbYesNo + vbDefaultButton2, "Data
ControlDemo") = vbNo Then Exit Sub

    With Data1.Recordset
```

```
        .Delete
        .MoveNext
        If .EOF Then
          If .RecordCount > 0 Then
            .MoveLast
          Else
            Data1.Refresh
          End If
        End If
    End With

BrisiIzlaz:
    Exit Sub

ErrBrisi:
    MsgBox "Greska ! " & Err.Description, vbCritical, "Data Control Demo"
    Resume BrisiIzlaz
End Sub
```

```
Private Sub bRefresh_Click()
On Error Resume Next
Dim CurPos As Long
CurPos = Data1.Recordset!CategoryID

Data1.Refresh
If Err.Number Then
    MsgBox "Greska ! " & Err.Description, vbCritical, "Data Control Demo"
    Err.Clear
    Exit Sub
End If
Data1.Recordset.FindFirst "CategoryID = " & Str$(CurPos)
End Sub
```

```
Private Sub bReset_Click()
    If MsgBox("Ponisti izmene", vbQuestion + vbYesNo + vbDefaultButton2,
"Data Control Demo") = vbNo Then Exit Sub
    Data1.UpdateControls

End Sub
```

```
Private Sub bInsObject_Click()
    OLE1.InsertObjDlg
End Sub
```

```
Private Sub bOpenObject_Click()
    Screen.MousePointer = 11
    OLE1.DoVerb (vbOLEOpen)
    Screen.MousePointer = 0
End Sub
```

```
Private Sub bEditObject_Click()
On Error GoTo ErrEdit
    Screen.MousePointer = 11
    OLE1.DoVerb (vbOLEUIActivate)

EditIzlaz:
    Screen.MousePointer = 0
    Exit Sub

ErrEdit:
    MsgBox "Ne mogu da otvorim objekt" & Chr$(13) & " za 'in place'
editovanje", vbExclamation, "Data Control Demo"
```

```
    Resume EditIzlaz

End Sub
```

```
Sub Dither(vForm As Form)
  Dim i As Integer
    vForm.DrawStyle = vbInsideSolid
    vForm.DrawMode = vbCopyPen
    vForm.ScaleMode = vbPixels
    vForm.DrawWidth = 2
    vForm.ScaleHeight = 256
    For i = 0 To 255
      vForm.Line (0, i)-(Screen.Width, i - 1), RGB(0, 255 - i, 255 - i), B
    Next
End Sub
```

Primer za analizu baze podataka (napredno korišćenje Data kontrole)

U ovom primeru će biti prikazana mogućnost analize struktura baze podataka korišćenjem Data kontrole. Izborom Access baze podataka, dobijamo njenu strukturu: tabele, polja i indekse

```
Private Sub cmdOpenDatabase_Click()
Dim i As Long
CD.Flags = &H1000
CD.ShowOpen
If CD.filename = "" Then Exit Sub
Data1.DatabaseName = CD.filename
Data1.Refresh
list1.Clear
For i = 0 To Data1.Database.TableDefs.Count - 1
      If Data1.Database.TableDefs(i).Attributes = 0 Then
            list1.AddItem Data1.Database.TableDefs(i).Name
End If
Next
End Sub
```

```
Private Sub list1_Click()
Dim i As Long
Dim Tabela As String
Tabela = list1.List(list1.ListIndex)
List2.Clear
For i = 0 To Data1.Database.TableDefs(Tabela).Fields.Count - 1
            List2.AddItem Data1.Database.TableDefs(Tabela).Fields(i).Name _
& "  (Size:" & Data1.Database.TableDefs(Tabela).Fields(i).Size _
& "  Type:" & Data1.Database.TableDefs(Tabela).Fields(i).Type & ")"
Next i
List3.Clear
For i = 0 To Data1.Database.TableDefs(Tabela).Indexes.Count - 1
List3.AddItem Data1.Database.TableDefs(Tabela).Indexes(i).Name
List3.AddItem "   -> " & Data1.Database.TableDefs(Tabela).Indexes(i).Fields
Next i
End Sub
```

Konstante za Fields(i).Type:

1	*dbBoolean*	*Yes/No*
2	*dbByte*	*Byte*
3	*dbInteger*	*Integer*
4	*dbLong*	*Long*
5	*dbCurrency*	*Currency*

6	*dbSingle*	*Single*
7	*dbDouble*	*Double*
8	*dbDate*	*Date/Time*
10	*dbText*	*Text*
11	*dbLongBinary*	*Long Binary (OLE Object)*
12	*dbMemo*	*Memo*

DAO – Data Access Objects

DAO predstavlja hijerahijsku strukturu objekata koji služe za pristup bazama podataka. U odnosu na Data kontrolu, DAO pristup je složeniji, zahteva više kodiranja, ali je istovremeno i fleksibilniji, sa više mogućnosti

Hijerarhija:

<table>
<tr><td colspan="4">DBENGINE</td></tr>
<tr><td colspan="3"></td><td>Error</td></tr>
<tr><td colspan="4">WorkSpace</td></tr>
<tr><td colspan="3" rowspan="4"></td><td>User</td></tr>
<tr><td>Group</td></tr>
<tr><td>Group</td></tr>
<tr><td>User</td></tr>
<tr><td colspan="4">DataBase</td></tr>
<tr><td>TableDef</td><td>QueyDef</td><td>Recordset</td><td rowspan="4"></td></tr>
<tr><td>Field</td><td>Field</td><td>Field</td></tr>
<tr><td>Index</td><td>Parameter</td><td>Relation</td></tr>
<tr><td>Field</td><td></td><td>Field</td></tr>
</table>

DBENGINE OBJEKT:

Metode:

- BeginTrans
- CommitTrans
- RollBack

CreateDatabase (Name, Locale, [Options])

Name - FullPathName

Locale - Jezik za DB

dbLangGeneral	English, German, French, Portuguese, Italian, and Modern Spanish
dbLangCyrillic	Russian
dbLangGreek	Greek
dbLangSlovenian	Slovenian

Options - Verzija Access-a

dbEncrypt	Creates an encrypted database.
dbVersion10	Microsoft Jet database engine version 1.0 file format.
dbVersion11	Microsoft Jet database engine version 1.1 file format.
dbVersion20	Microsoft Jet database engine version 2.0 file format.
dbVersion30	(Default) Microsoft Jet database engine version 3.0 file format (compatible with version 3.5).

```
CompactDatabase olddb, newdb, locale, options, password
```

CreateWorkspace (name, user, password, type)

type:

dbUseJet Creates a Microsoft Jet workspace.
dbUseODBC Creates an ODBCDirect workspace.

`Idle [dbRefreshCache]|[dbFreeLocks]`

OpenDatabase (dbname, [options], [read-only], [connect])

`RepairDatabase dbname`

Svojstva:

Version
SystemDB

DataBase objekt

Svojstva:

Name (read only)
CollatingOrder (readonly)
Connect

Microsoft Jet Database	[database];	drive:\path\filename.mdb
dBASE III	dBASE III;	drive:\path
dBASE IV	dBASE IV;	drive:\path
dBASE 5	dBASE 5.0;	drive:\path
Paradox 3.x	Paradox 3.x;	drive:\path
Paradox 4.x	Paradox 4.x;	drive:\path
Paradox 5.x	Paradox 5.x;	drive:\path
FoxPro 2.0	FoxPro 2.0;	drive:\path
FoxPro 2.5	FoxPro 2.5;	drive:\path
FoxPro 2.6	FoxPro 2.6;	drive:\path
Excel 3.0	Excel 3.0;	drive:\path\filename.xls
Excel 4.0	Excel 4.0;	drive:\path\filename.xls
Excel 5.0 / Excel 95	Excel 5.0;	drive:\path\filename.xls
Excel 97	Excel 97;	drive:\path\filename.xls
HTML Import	HTML Import;	drive:\path\filename
HTML Export	HTML Export;	drive:\path
Text Text;	drive:\path	

ODBC ODBC;
DATABASE=database;
UID=user;
PWD=password;
DSN= datasourcename;
[LOGINTIMEOUT=seconds;] None

RecordsAffected - broj slogova koji su izmenjeni poslednjima Action upitom
Transactions (True/False)
Updatable (True/False)

Metode

OpenRecordset (source, type, options, lockedits)
Source: Table, StoredSQL, SQL statement
Type: dbOpenTable
dbOpenDynamic (ODBCDirect workspaces only)
dbOpenDynaset
dbOpenSnapshot

dbOpenForwardOnly
Options: (isto kao Options svojstvo data kontrole)
Lockedits: dbReadOnly
dbPessimistic
dbOptimistic

Execute source, options
Source: StoredSQL, Action SQL
Options:
dbDenyWrite
dbSQLPassThrough
dbFailOnError Rolls back updates.
dbSeeChanges Generates a error if another user is changing data you are editing

TableDefs objekt

Metode

Append
CreateField ([name[, type [, size]]])
CreateIndex ([name])
Delete

Fields Kolekcija

Svojstva

AllowZeroLength
dbAutoIncrField
Count
DataUpdatable
DefaultValue
Name
Required
Type
ValidationRule
ValidationText
Value

Metode

Append
Delete

Index objekt, Indexes kolekcija

Svojstva

IgnoreNulls
Name
Primary
Unique

Metode

Append
CreateIndex
Delete

Primer - Kreiranje baze podataka pomoću DAO objekata

```
Screen.MousePointer = 11
Dim WSPC As Workspace
Dim Baza As Database
Dim Tabela As TableDef
Dim Indeks As Index
Dim Polje As Field
Dim Relacija As Relation
Dim ImeBaze As String
ImeBaze = "C:\MYDB.MDB"
If Dir("C:\MYDB.MDB") <> "" Then Kill ("C:\MYDB.MDB")
Set WSPC = DBEngine.Workspaces(0)

Set Baza = WSPC.CreateDatabase (ImeBaze, dbLangGeneral, dbVersion30)
Baza.Close

Set Baza = WSPC.OpenDatabase(ImeBaze)

'TABELA KLIJENTI
Set Tabela = Baza.CreateTableDef("Klijenti")
    Set Polje = Tabela.CreateField("ID", dbLong)
    Polje.Attributes = dbAutoIncrField
    Polje.Required = True
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("Ime", dbText, 50)
    Polje.Required = True
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("Adresa", dbText, 30)
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("Starost", dbLong, 20)
    Tabela.Fields.Append Polje
Baza.TableDefs.Append Tabela

'Indeksi za klijente
Set Tabela = Baza.TableDefs("Klijenti")
    Set Indeks = Tabela.CreateIndex("Primarni indeks Klijenti")
  Set Polje = Indeks.CreateField("ID")
  Indeks.Fields.Append Polje
        Indeks.Primary = True
        Tabela.Indexes.Append Indeks

    Set Indeks = Tabela.CreateIndex("Indeks Ime")
        Set Polje = Indeks.CreateField("Ime")
        Polje.Attributes = dbDescending              'ili dbAscending
        Indeks.Fields.Append Polje
        Tabela.Indexes.Append Indeks

    Set Indeks = Tabela.CreateIndex("Ime i adresa")
        Set Polje = Indeks.CreateField("Ime")
        Polje.Attributes = dbAscending
        Indeks.Fields.Append Polje
        Set Polje = Indeks.CreateField("Adresa")
        Polje.Attributes = dbDescending
        Indeks.Fields.Append Polje
        Indeks.Unique = True
        Tabela.Indexes.Append Indeks

'TABELA UPLATE
```

```
Set Tabela = Baza.CreateTableDef("Uplate")
    Set Polje = Tabela.CreateField("ID Uplate", dbLong)
    Polje.Attributes = dbAutoIncrField
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("ID Klijenta", dbLong)
    Polje.Required = True
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("Datum", dbDate)
    Polje.Required = True
    Polje.DefaultValue = "Date()"
    Tabela.Fields.Append Polje

    Set Polje = Tabela.CreateField("Iznos uplate", dbCurrency)
    Polje.ValidationRule = "<>0"
    Polje.ValidationText = "Morate uneti uplatu"
    Tabela.Fields.Append Polje
Baza.TableDefs.Append Tabela

'INDEKSI ZA UPLATE
Set Tabela = Baza.TableDefs("Uplate")
    Set Indeks = Tabela.CreateIndex("Primarni indeks Uplate")
        Set Polje = Indeks.CreateField("ID Uplate")
        Indeks.Fields.Append Polje
        Indeks.Primary = True
        Tabela.Indexes.Append Indeks

    Set Indeks = Tabela.CreateIndex("ID Klijenta")
        Set Polje = Indeks.CreateField("ID Klijenta")
        Indeks.Fields.Append Polje
        Tabela.Indexes.Append Indeks

    Set Indeks = Tabela.CreateIndex("Datum")
        Set Polje = Indeks.CreateField("Datum")
        Indeks.Fields.Append Polje

        Tabela.Indexes.Append Indeks


'RELACIJA "Klijenti -> Uplate"
Set Relacija = Baza.CreateRelation("Klijenti i uplate", "Klijenti", "Uplate")
Set Polje = Relacija.CreateField("ID")
    Polje.ForeignName = "ID Klijenta"
    Relacija.Fields.Append Polje
    Relacija.Attributes = dbRelationDeleteCascade + dbRelationUpdateCascade
Baza.Relations.Append Relacija

Baza.Close
WSPC.Close
Screen.MousePointer = 0
Msgbox "Baza podataka je kreirana"
```

SQL (Structured Query Language)

SQL (Structured Query Language), u prevodu struktuirani jezik za zadavanje upita bazi podataka, je jezik kojim se ostvaruju potrebni zadaci za definiciju (DDL), manipulaciju (DML) i bezbednost podataka. SQL je de facto industrijski standard koji omogućava komunikaciju između aplikacija i RDBMS-a. Da ne bi bilo zabune, SQL nije jezik koga "razume" Visual Basic. VB samo ima mogućnost da RDBMS-u pošalje zahtev u SQL-u, prihvati rezultate njegovog izvršavanja i prikaže ih u odgovarajućem obliku.

Istorijat SQL-a počinje 1970 u IBM laboratorijama. Iako je kasnije predstavljeno više komercijalnih varijanti, prvi standard je usvojen tek 1986 od strane ANSI i ISO organizacija. Ovaj standard je poboljšan 1989 i poneo je naziv SQL-89. Od tada ISO je izdao novi standard 1992 pod nazivom SQL-92 ili SQL2. Trenutno je u razvoju SQL3 standard. SQL je takođe formalno prihvatila organizacija IEC (International Electrotechnical Commision).

Svaki proizvođač RDBMS-a podržava SQL2 standard, ali ga i obično proširuje kako bi podržao nove mogućnosti. Ako projektujete aplikacije koje rade sa različitim RDBM sistemima, vodite računa o ovim malim, ali potencijalno problematičnim razlikama. Samo jezgro SQL-a je čvrsto standardizovano i ne bi trebalo da imate problema u tom smislu.

Rad sa bazama podataka nije moguć bez temeljnog poznavanja SQL-a.

Primer: SQL TESTER

KONTROLE:

- Textbox (Multiline,Vertical SB)
- List1
- Data1 DatabaseName: Fakture, Recordsource: (none)
- Command Button x 3
- (Izvrsi upit, Snimi Upit, Brisi Upit)
- DBGrid1 (bound to data1)

KOD:

Form1_General

```
Sub PrikaziUpite()
      'Prikazuje upite iz baze u List1 box-u
      MOff
      Dim B As Database
      Dim Q As QueryDefs
      Dim i As Integer
      Set B = OpenDatabase(Data1.DatabaseName)
      List1.Clear
      For i = 0 To B.QueryDefs.Count - 1
          List1.AddItem B.QueryDefs(i).Name
      Next
      B.Close
      MOn
End Sub
```

```
Sub MOn()
      screen.mousepointer = 0
end sub
```

```
Sub MOff()
      screen.mousepointer = 11
end sub
```

```
Izvrsi upit button:
Private Sub Command1_Click()
      Dim i As Long
      MOff
      On Error GoTo Greska
      Data1.RecordSource = Text1.Text
      Data1.Refresh
      If Data1.Recordset.RecordCount <> 0 Then
          Data1.Recordset.MoveLast
          i = Data1.Recordset.RecordCount
          Form1.Caption = "Records: " & i
          Data1.Recordset.MoveFirst
      Else
            Form1.Caption = "Records: 0"
```

```
      End If

Izlaz:
      MOn
Exit Sub

Greska:
      MsgBox Error$, 16
      Resume Izlaz
End Sub
```

```
Snimi upit button:
Private Sub Command2_Click()
      On Error GoTo Greska
      Dim NazivUpita As String
      NazivUpita = InputBox("Unesite naziv upita", "Snimanje upita")
      If Trim$(NazivUpita) = "" Then Exit Sub
      MOff
      Dim B As Database
      Dim Q As QueryDef
      Set B = OpenDatabase(Data1.DatabaseName)
      Set Q = B.CreateQueryDef(NazivUpita, Text1.Text)
      B.Close
      PrikaziUpite
Izlaz:
      MOn
      Exit Sub

Greska:
      MsgBox Error$, 16, "Snimanje upita"
      Resume Izlaz
End Sub
```

```
Brisi Upit Button:
Private Sub Command3_Click()
      On Error GoTo Greska
      If List1.ListCount = 0 Then
          MsgBox "Nema upita u bazi", 48
          Exit Sub
      End If
      If List1.ListIndex < 0 Then
            MsgBox "Morate izabrati upit iz liste"
            Exit Sub
      End If
      Dim NazivUpita As String
      Dim B As Database
      NazivUpita = List1.List(List1.ListIndex)
      Set B = OpenDatabase(Data1.DatabaseName)
      Set Q = B.QueryDefs(NazivUpita)
      B.QueryDefs.Delete NazivUpita
      B.Close
      PrikaziUpite
Izlaz:
      MOn
      Exit Sub

Greska:
      MsgBox Error$, 16
      Resume Izlaz
End Sub
```

```
List1_Click event:
Private Sub List1_Click()
      Dim NazivUpita As String
      NazivUpita = List1.List(List1.ListIndex)
      Dim B As Database
      Dim Q As QueryDef
      Set B = OpenDatabase(Data1.DatabaseName)
      Set Q = B.QueryDefs(NazivUpita)
      Text1.Text = Q.SQL
      B.Close
End Sub
```

Akcioni upiti:

```
Data1.Database.Execute S, dbFailOnError
```

UPDATE *ImeTabele* **SET** *ImePolja1* = *Vrednost1*, *ImePolja2*=*Vrednost2*,...
WHERE Uslov
UPDATE Categories SET CategoryName = "Nova kategorija"
WHERE CategoryID=3

DELETE FROM *ImeTabele*
WHERE Uslov
DELETE * FROM Categories
WHERE CategoryID=3

INSERT INTO DestinacionaTabela (Polje1, Polje2,...)
SELECT Polje1, Polje2,...
FROM IzvornaTabela
WHERE UslovDestinacioneTabele
INSERT INTO Kategorije (CategoryName, Description, CategoryID)
SELECT Categories.CategoryName, Categories.Description, Categories.CategoryID
FROM Categories
WHERE Categories.CategoryID In (2,3,4)

Transakcije:
BeginTrans
CommitTrans
Rollback

DDL SQL: Data Definition Language

CREATE TABLE *ImeTabele* (*Polje1* tip [(velicina)] [NOT NULL],
Polje2 tip [(velicina)] [NOT NULL]...)

text
byte
integer
long
single
double
currency
datetime
yesno
OLEObject
Memo

```
On Error GoTo QER
Screen.MousePointer = 11
BeginTrans
Data1.Database.Execute Text1.Text, dbFailOnError
If MsgBox("Potvrda ", vbQuestion + vbYesNo, "Potvrda izmena") = vbYes Then
        CommitTrans: Call PregledTabela(Data1.DatabaseName): Call PregledIndeksa
  Else
        Rollback
End If
QERI:
      Screen.MousePointer = 0
      Exit Sub
QER:
   Rollback: MsgBox "SQL greska :" & Chr$(13) & Error$, 16: Resume QERI
```

DROP TABLE ***ImeTabele***

CREATE [UNIQUE] **INDEX** *ImeIndeksa* **ON** *ImeTabele* (*Polje1* [ASC|DESC][, *Polje2* [ASC|DESC], ...])

[**WITH** { PRIMARY | DISALLOW NULL | IGNORE NULL }]

DROP INDEX *ImeIndeksa* **ON** *ImeTabele*

ALTER TABLE ImeTabele **ADD COLUMN** ImePolja Tip[(Velicina)] [NOT NULL]

ALTER TABLE ImeTabele **DROP COLUMN** ImePolja

SQL Varijanta kreiranja tabela:

```
If Dir("C:\MYDB.MDB") <> "" Then Kill ("C:\MYDB.MDB")
Dim Baza As Database

Screen.MousePointer = 11
Set Baza = CreateDatabase("C:\MYDB.MDB", dbLangGeneral, dbVersion20)

Baza.Execute "CREATE TABLE Klijenti (ID COUNTER ,Ime TEXT (50),  Adresa TEXT
(30),  _
Starost SINGLE)", dbFailOnError
Baza.Execute "CREATE INDEX [Primarni indeks] ON Klijenti (ID ASC) WITH
PRIMARY", dbFailOnError
Baza.Execute "CREATE INDEX [Ime klijenta] ON Klijenti (Ime ASC)",
dbFailOnError
Baza.Execute "CREATE INDEX [Ime i adresa] ON Klijenti (ID ASC, Adresa ASC)",
dbFailOnError

Baza.Execute "CREATE TABLE UPLATE ([ID Uplate] COUNTER, [ID Klijenta] LONG,
Datum DATE,  _
Iznos CURRENCY)", dbFailOnError
Baza.Execute "CREATE INDEX [Primarni indeks] ON UPLATE ([ID Uplate]) WITH
PRIMARY", dbFailOnError
Baza.Execute "CREATE INDEX [ID Klijenta] ON UPLATE ([ID Klijenta])",
dbFailOnError
Baza.Execute "CREATE INDEX [Datum uplate] ON UPLATE (Datum DESC)",
dbFailOnError

Baza.Execute "ALTER TABLE Uplate ADD CONSTRAINT Relacija Foreign Key([ID
Klijenta]) _
References Klijenti(ID)", dbFailOnError

Baza.Close
Screen.MousePointer = 0
```

Primeri SQL naredbi:

SELECT [ALL|DISTINCT] lista polja
FROM lista tabela [relacije izmedju tabela]
[WHERE kriterijumi]
[ORDER BY lista polja]
[GROUP BY lista polja]
[HAVING kriterijumi agregatnih f-ja]

01> SELECT * FROM Fakture;
02> SELECT * FROM Fakture ORDER BY Datum;
03> SELECT Datum FROM Fakture;

04> SELECT DISTINCTROW * FROM Stavke;
05> SELECT DISTINCT (Stavka) FROM Stavke;
06> SELECT * FROM Stavke WHERE [ID Fakture]=5;
07> SELECT * FROM Fakture WHERE Datum > #20/10/1993#;
SELECT * FROM Fakture WHERE Datum Between #20/03/1992# and #15/04/1993#
ORDER BY Datum;
09> SELECT Fakture.ID, Fakture.Datum, Stavke.Stavka FROM Fakture, Stavke WHERE Fakture.ID = Stavke.[ID FAKTURE] ORDER BY Fakture.Datum;
10> SELECT ID & ', ' & Datum FROM Fakture;
11> SELECT ID & ', ' & Datum AS [ID i Datum] FROM Fakture;
12> SELECT Ucase$(Stavka) FROM Stavke;
13> SELECT Format$(Cena,"###,##0.00 din") FROM Stavke;
14> SELECT 'Godina: ' & Year(Datum) & ' Mesec: ' & Month(Datum)
& ' Dan: ' & Day(Datum) AS [Godina Mesec Dan] FROM FAKTURE;
15> SELECT * FROM Fakture WHERE Year(Datum)=1993;
16> SELECT * FROM STAVKE WHERE Stavka like 'a*';
17> SELECT * FROM STAVKE WHERE Stavka like 'a*' AND Cena >10;
18> SELECT SUM (Cena) AS [Ukupna cena] FROM Stavke;
19> SELECT SUM (Cena) AS [Ukupna cena], AVG (Cena) AS [Prosecna cena],
VAR (Cena) AS [Varijansa Cene], STDEV (Cena) AS [Standardna greska cene],
COUNT (Cena) AS [Ukupno uplata], MIN (Cena) AS [Najmanja cena],
MAX (Cena) AS [Najveca cena] FROM Stavke;
20> SELECT * FROM FAKTURE WHERE DATUM IN (#10/02/1992#, #20/01/1993#);
21> SELECT Fakture.Datum, SUM (Stavke.Cena) AS Ukupno
FROM Fakture,Stavke WHERE Stavke.[ID Fakture] = Fakture.ID
GROUP BY Fakture.Datum;
22> SELECT Year(Fakture.Datum) AS Godina, SUM (Stavke.Cena) AS Ukupno
FROM Fakture,Stavke WHERE Stavke.[ID Fakture] = Fakture.ID
GROUP BY Year(Fakture.Datum);
23> SELECT Year(Fakture.Datum) AS Godina, SUM (Stavke.Cena) AS Ukupno
FROM Fakture,Stavke WHERE Stavke.[ID Fakture] = Fakture.ID
GROUP BY Year(Fakture.Datum) HAVING SUM (Stavke.Cena) > 15000;

ACTION Upiti:

Data1.Database.Execute
24> DELETE * FROM Fakture WHERE Fakture.ID=25;
25> UPDATE Stavke SET Stavke.Cena = Stavke.Cena * 1.10;

KLASE

Klase omogućavaju kreiranje objekata u VB-u koji mogu posedovati svoj set svojstava i metoda. Realizuju se kroz Class Module, i predstavljaju šablon na osnovu kojeg se u kodu kreiraju objektne varijable.

PRIMER: Klasa KvadJednacina

```
Option Explicit
Private PdblA As Double
Private PdblB As Double
Private PdblC As Double
Private PdblD1 As Double
Private PdblD2 As Double
Private PNemaR As Boolean
Private PTekst As String
Private D As Double

Private Sub Class_Initialize()
      MsgBox "Klasa inicijalizovana", vbInformation
```

```
      PdblA = 0
      PdblB = 0
      PdblC = 0
End Sub
```

```
Private Sub Class_Terminate()
      MsgBox "Klasa zavrsena", vbInformation
End Sub
```

```
Public Property Get A() As Double
      A = PdblA
End Property
```

```
Public Property Let A(locA As Double)
      PdblA = locA
End Property
```

```
Public Property Get B() As Double
      B = PdblB
End Property
```

```
Public Property Let B(locB As Double)
      PdblB = locB
End Property
```

```
Public Property Let C(locC As Double)
      PdblC = locC
End Property
```

```
Public Property Get C() As Double
      C = PdblC
End Property
```

```
Public Sub Izracunaj()
      If PdblA = 0 And PdblB = 0 And PdblC = 0 Then
            PdblD1 = 0
            PdblD2 = 0
            Exit Sub
      End If

      D = PdblB ^ 2 - 4 * PdblA * PdblC
      If D < 0 Then
            PdblD1 = 0
            PdblD2 = 0
            PNemaR = True
            PTekst = "Nema resenja"
            Exit Sub
      Else
            PNemaR = False
      End If

      D = Sqr(D)
      PdblD1 = (-PdblB + D) / (2 * PdblA)
      PdblD2 = (-PdblB - D) / (2 * PdblA)
      PTekst = PdblA & "x^2 + " & PdblB & "x +" & PdblC & "=0: " _
            & Chr$(13) & "x1:" & PdblD1 & "  x2:" & PdblD2
End Sub
```

```
Public Property Get NemaResenja()
      NemaResenja = PNemaR
End Property
```

```
Public Property Get Resenje1() As Double
      Resenje1 = PdblD1
End Property
```

```
Public Property Get Resenje2() As Double
      Resenje2 = PdblD2
End Property
```

```
Public Property Get TekstResenja()
      TekstResenja = PTekst
End Property
```

Korišćenje kreirane klase

```
Private Sub cmdRacun_Click()
      Dim X As New KvadJednacina
      X.A = CDbl(txtA)
      X.B = CDbl(txtB)
      X.C = CDbl(txtC)
      X.Izracunaj
      If X.NemaResenja Then
            MsgBox "Jednacina nema resenje", vbExclamation
      Else
            MsgBox "x1=" & X.Resenje1 & Chr$(13) & _
                  "x2=" & X.Resenje2, vbInformation, "Resenja"
      End If

      MsgBox X.TekstResenja, vbInformation, "Tekst resenja"
End Sub
```

Primer postavljanja *Tools* -> *Procedure* Attributes (Description polje)

OCX KONTROLE

Visual Basic ima mogućnost kreiranja OCX kontrola, koje kasnije možete koristiti kako u VB-u, tako i u drugim aplikacijama kojjie podržavaju ovaj mehanizam (Access, Excel, Delphi ...). Koncept kreiranja se zasniva na klasi, kao jezgru koje obezbeđuje svojstva , metode i događaje i UserForm objektu koji obezbeđuje korisnički interfejs kontrole.

OCX kontrole se sastoje od jedne ili više standardnih VB kontrola, ali takođe mogu sadržati i druge OCX kontrole. U sledećem primeru ćemo demonstrirati kreiranje jednostavne kontrole za postavljanje boje "Host" objekta, njeno korišćenje u VB-u i Word Form objektu.

PRIMER: RGBOCX

```
Option Explicit
Event IzmenaBoje()

Private Sub sBlue_Change()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sBlue_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sGreen_Change()
```

```
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sGreen_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sRed_Change()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub sRed_Scroll()
      RaiseEvent IzmenaBoje
      PostaviBoju
End Sub
```

```
Private Sub UserControl_Resize()
      sRed.Width = UserControl.Width
      sGreen.Width = UserControl.Width
      sBlue.Width = UserControl.Width
      UserControl.Height = 255 * 3
End Sub
```

```
Public Property Get Red() As Integer
    Red = sRed.Value
End Property
```

```
Public Property Let Red(ByVal New_Red As Integer)
    If New_Red < 0 Or New_Red > 255 Then
        MsgBox "Invalid red", vbCritical, "Red"
        Exit Property
    End If
    sRed.Value() = New_Red
    PropertyChanged "Red"
End Property
```

```
Public Property Get Green() As Integer
      Green = sGreen.Value
End Property
```

```
Public Property Let Green(ByVal New_Green As Integer)
      If New_Green < 0 Or New_Green > 255 Then
            MsgBox "Invalid green", vbCritical, "Green"
            Exit Property
      End If
      sGreen.Value() = New_Green
      PropertyChanged "Green"
End Property
```

```
Public Property Get Blue() As Integer
      Blue = sBlue.Value
End Property
```

```
Public Property Let Blue(ByVal New_Blue As Integer)
      If New_Blue < 0 Or New_Blue > 255 Then
            MsgBox "Invalid blue", vbCritical, "Blue"
            Exit Property
      End If
```

```
      sBlue.Value() = New_Blue
      PropertyChanged "Blue"
End Property
```

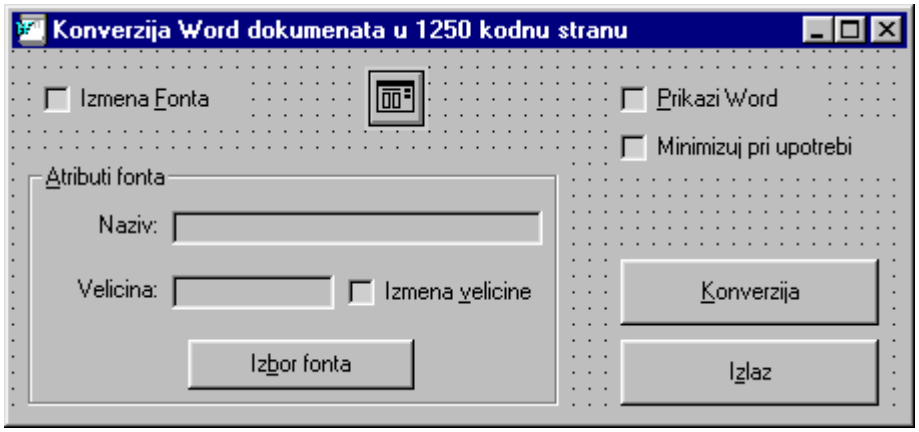
```
' Read property values
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
      sRed.Value = PropBag.ReadProperty("Red", 0)
      sGreen.Value = PropBag.ReadProperty("Green", 0)
      sBlue.Value = PropBag.ReadProperty("Blue", 0)
End Sub
```

```
'Write property values to storage
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
      Call PropBag.WriteProperty("Red", sRed.Value, 0)
      Call PropBag.WriteProperty("Green", sGreen.Value, 0)
      Call PropBag.WriteProperty("Blue", sBlue.Value, 0)
End Sub
```

```
Public Sub PostaviBoju()
      UserControl.Parent.BackColor = RGB(Red, Green, Blue)
End Sub
```

VEZA SA DRUGIM OFFICE APLIKACIJAMA

Word demo



Okvir (Frame) - Visible: False
chkIzmenaFonta
CD
chkWord
chkMin
lFontName
lFontSize
chkSize
cmdIzborFonta
cmdConvert
cmdIzlaz

Kod:

```
Private Sub chkIzmenaFonta_Click()
    Okvir.Visible = chkIzmenaFonta.Value
```

```
End Sub
```

```
Private Sub cmdIzborFonta_Click()
    CD.Flags = 3
    On Error Resume Next
    CD.ShowFont
    If Err.Number Then
        Err.Clear
        LFontName.Caption = ""
        Exit Sub
    End If
    LFontName.Caption = CD.FontName
    LFontSize.Caption = CD.FontSize

End Sub
```

```
Private Sub cmdConvert_Click()
Dim sOrg As String
Dim sNew As String
Dim i As Integer
Dim x As Integer

sOrg = "{[|\~^}]`@"
sNew = "šŠ" & ChrW(273) & ChrW(272) & ChrW(269) & ChrW(268) & ChrW(263) &
ChrW(262) & ChrW(382) & ChrW(381)

    CD.Flags = &H1000 Or &H4 Or &H2
    CD.DialogTitle = "Izaberite izvorni dokument"
    CD.Filter = "Word document (*.doc)|*.doc|Svi fajlovi (*.*)|*.*"
    On Error Resume Next
    CD.ShowOpen
    If Err.Number Then
        Err.Clear
        SourceDoc = ""
        Exit Sub
    End If
    SourceDoc = CD.filename

    CD.DialogTitle = "Unesite destinacioni dokument"
    CD.ShowSave
    If Err.Number Then
        Err.Clear
        DestinationDoc = ""
        Exit Sub
    End If
    DestinationDoc = CD.filename
    If chkMin.Value Then Me.WindowState = vbMinimized

    Dim Word As Object
    On Error GoTo GRESKA
    Me.Caption = "Otvaram Word...": Me.Refresh
    Set Word = CreateObject("Word.application")

    With Word
        If chkWord.Value Then
            .Visible = True
        End If
        x = DoEvents()
        .Documents.Open filename:=SourceDoc,ConfirmConversions:=True,
ReadOnly _
        :=False,
AddToRecentFiles:=False,PasswordDocument:="",PasswordTemplate _
```

```
        :="", Revert:=False, WritePasswordDocument:="",
WritePasswordTemplate:="" _
        , Format:=wdOpenFormatAuto
        Me.Caption = "Otvaram dokument: " & SourceDoc: Me.Refresh
        .Selection.Find.ClearFormatting
        .Selection.Find.Replacement.ClearFormatting
        For i = 1 To Len(sOrg)
            x = DoEvents()
            Me.Caption = "Zamena slova: " & Mid$(sOrg, i, 1): Me.Refresh
            With .Selection.Find
                .Text = Mid$(sOrg, i, 1)
                .Replacement.Text = Mid$(sNew, i, 1)
                .Forward = True
                .Wrap = wdFindContinue
                .Format = False
                .MatchCase = False
                .MatchWholeWord = False
                .MatchWildcards = False
                .MatchSoundsLike = False
                .MatchAllWordForms = False
            End With
            .Selection.Find.Execute Replace:=wdReplaceAll
            x = DoEvents()
        Next i
        If chkIzmenaFonta.Value Then
            If Trim$(LFontName.Caption) <> "" Then
                Me.Caption = "Izmena fonta: " & LFontName.Caption: Me.Refresh
                .Selection.WholeStory
                .Selection.Font.Name = LFontName.Caption
                If chkSize.Value And Val(LFontSize) <> 0 Then
                    Me.Caption = "Izmena velicine fonta: " &
LFontSize.Caption: Me.Refresh
                    .Selection.Font.Size = Val(LFontSize)
                End If
            End If
        End If
        x = DoEvents()
        Me.Caption = "Snimam dokument kao: " & DestinationDoc: Me.Refresh
        .ActiveDocument.SaveAs filename:=DestinationDoc, FileFormat:= _
        wdFormatDocument, LockComments:=False,
Password:="",AddToRecentFiles:= _
        True, WritePassword:="", ReadOnlyRecommended:=False, _
            EmbedTrueTypeFonts:= _
            False, SaveNativePictureFormat:=False, SaveFormsData:=False, _
            SaveAsAOCELetter:=False
        Me.Caption = "Zatvaram Word": Me.Refresh
        x = DoEvents()
        .quit wdDoNotSaveChanges
    End With

    Beep
    MsgBox "Izvor: " & SourceDoc & Chr$(13) & "Destinacija: " &
DestinationDoc, vbInformation, "Zavrsena konverzija"

IZLAZ:
    Me.Caption = "Konverzija Word dokumenta u 1250 kodnu stranu"
    If chkMin.Value Then Me.WindowState = vbNormal
    Exit Sub

GRESKA:
    MsgBox "Greska: " & Err.Description, vbCritical, "Konverzija"
    Resume IZLAZ
```

```
End Sub
```

TREE VIEW KONTROLA

Omogućava pregled stavki na više nivoa isto kao u Windows Explorer-u. Svaka stavka može biti na nekom od nivoa. Ako je na prvom nivou onda kažemo da je ta stavka roditelj, a ako ispod sebe ime podstavke, onda te stavke zovemo "deca" predhodne. Pomoću nje možemo prikazati neku hijerarhisku strukturu identičnu na primer, strukturi direktorijuma na disku. Svaka stavka može imati podstavke, podstavke mogu dalje imati svoje podstavke itd. Jednu stavku zovemo "ćvor" (node) i predstavljamo je objektnom promenjivom tipa Node.

Ova razgranata struktura mora biti u sprezi sa ImageList kontrolom. ImageList obezbeđuje slike (tipa Icon ili BitMap), koje omogućavaju grafičku prezentaciju granjanja. Tipično, to su bitmape:
- zatvorena fascikla (c:\vbasic\bitmaps\outline\closed.bmp)
- otvorena fascikla (c:\vbasic\bitmaps\outline\open.bmp)
- dokument (c:\vbasic\bitmaps\outline\leaf.bmp)

Oubicajeno je da slika leaf.bmp predstavlja čvor koji ispod sebe nema "dece" i poslednji je u toj grani.

Properties dialog ListView kontrole:

- Style:
  više stilova grafičkog prikaza strukture,
  najčešče korišćeni je stil 7 (Lines,Plus/Minus,Image, Text)

- Line style:
  dve varijante prikaza linija koje povezuju čvorove
  Tree lines, Root lines

- Label edit
  Posto svaki čvor sadrži tekst koji se vidi na ekranu, omogućeno je da se taj tekst edituje (potpuno isto kao kada bi ste hteli da u Windows Explorer-u izmenite naziv nekog fajla ili direktorijuma). Dakle, jednostruki klik mišem na predhodno selektovani tekst omogućava njegovo editovanje. Ako je Label edit postavljen na Automatic ovo se upravo ovako i događa, u suprotnom neophodno je programski obezbediti editovanje.

- Image List
  Unosi se (tj. bira iz liste) naziv ImageList kontrole čije slike će ListView koristiti za prikaz

- Sorted
  Ako je postavljeno na True. čvorovi u ListView kontroli se automatski sortiraju po tekstu koji je na njima.

Sintaksa za dodavanje novih čvorova u ListView kontroli je sledeca:

Predhodno dimenzionišemo Node objekat:
Dim A as Node

zatim,

Set A = TreeView1.Nodes.Add ([predak], [veza sa predkom],
key, tekst, slika, [Selektovana slika])

[predak]: ako dodajemo stavku koja treba da je "dete" neke ranije dodate stavke, ovde upisujemo Key te predhodne stavke "roditelja". Ako je stavka na prvom nivou ovaj parametar treba izostaviti.

[veza sa predkom]: ako smo definisali predhodnu opciju [predak] onda je obavezno definisati kakva je veza nove stavke sa svojim predkom. Postoji više mogucnosti, ali najčešće koristimo opciju tvwChild. Ona označava da je nova stavke dete stavke čiji je key [predak].

kljuc: jedinstveni identifikator svakog čvora u kontroli. Ne mozemo imati dva ista identifikatora na jednoj list View kontroli. Tip podataka je string i za njega važe isti kriterijumi kao i za naziv promenjive u Visual Basic-u.

Tekst: Tekst čvora.

Slika: Slika koja prezentuje ovaj čvor. Tip podataka je broj. Na primer 3, sto znaci da ovaj čvor prezentuje treća slika iz ImageList kontrole koju smo vezali za ListView kontrolu.

Selektovana slika: Opcioni broj slike iz ImageList kontrole koja će prezentovati selektovan čvor.

```
Dim A as Node
Set A = TreeView1.Nodes.Add ( , ,"N1","Prvi cvor",1)
```
Ovim smo dodali prvi čvor ciji je identifikator (key) N1, na kome piše "Prvi cvor" i koga će grafički predstavljati slika broj 1 iz dodeljene ImageList kontrole.

```
Set A = TreeView1.Nodes.Add ("N1", vwChild, "C1", "Dete prvog cvora",3)
```
Sada smo dodali novi čvor koji je dete (vwChild) čvora N1, na kome piše "Dete prvog cvora", i koga će grafički predstavljati slika broj 3 iz dodeljenje ImageList kontrole.

TreeView - PRIMER:

```
Private Sub Command1_Click
' Add Node objects.
Dim nodX As Node
Dim i As Integer

For i = 0 To TreeView1.Nodes.Count - 1
TreeView1.Nodes.Clear
Next i

'Syntax: Add(relative, relationship, key, text, image, selectedimage)
'tvwLast1      Last.                    Nod se dodaje na kraj posle svih ostalih nodova na
'istom nivou na kojem je
'nod koji je naveden pod relative
'tvwNext      2      Next.              Nod se dodaje posle noda koji je relative
'tvwPrevious  3      Previous.     Nod se dodaje pre noda koji je relative
'tvwChild     4                                  Dete noda koji je relative

'Prvi nod "Koren"', slika 2, Key je "k"
Set nodX = TreeView1.Nodes.Add(, , "k", "Koren", 2)
nodX.ExpandedImage = 1        'Slika u otvorenom režimu je slika 1

' Sledeći nod  'Parent', slika 2
Set nodX = TreeView1.Nodes.Add(, , "r", "Roditelj", 2)
nodX.ExpandedImage = 1

Set nodX = TreeView1.Nodes.Add(, , "p", "Poslednji nod", 2)
nodX.ExpandedImage = 1

' Ovaj nod je dete Noda 1 ("Koren"),koristi sliku 3
Set nodX = TreeView1.Nodes.Add(1, tvwChild, "d", "Dete", 3)

'Sledeći nod je dete noda pod imenom "Roditelj"
```

```
'umesto da koristimo indeks koristimo Key noda "Roditelj" ("r.")
Set nodX = TreeView1.Nodes.Add("r", tvwChild, "nes", "Ne sortirano", 2)
nodX.ExpandedImage = 1

' Dodajemo tri noda sve deca noda "Ne sortirano"
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "xz", "Xyz", 3)
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "datum", "1967", 3)
Set nodX = TreeView1.Nodes.Add("nes", tvwChild, "srt", "Sortirano", 2)
nodX.ExpandedImage = 1
' Sledeci kreirani nodovi će biti sortirani
nodX.Sorted = True

' Na kraju dodajemo tri noda, decu noda "Sorted,"
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "x", "Milan", 3)
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "j", "Petar", 3)
Set nodX = TreeView1.Nodes.Add("srt", tvwChild, "a", "Aleksandar", 3)
nodX.EnsureVisible
End Sub
```

```
Private Sub TreeView1_NodeClick(ByVal Node As Node)
Label1.Caption = Node.Text & "  " & Node.Index
End Sub

Private Sub Command2_Click()
Dim I As Integer
For I = 1 To TreeView1.Nodes.Count
TreeView1.Nodes(I).Expanded = True 'ili False
Next I
End Sub
```

Kontrola: PROGRESS BAR

Min
Max
Value

Kontrola: SLIDER

Min
Max
SmallChange
LargeChange
RangeSelection (SelectionStart, SelectionLength)

Appearance
Orientation (Horizontal,Vertical)
TickStyle (Bottom/Right,Top/Left,Both,NoTicks)
TickFrequency (Number)

Kontrola: TOOLBAR

Properties

General:
- AllowCustomize
  - ShowTips
  - Enabled
  - Wrappable
  - ImageList
  - ButtonHeight
  - ButtonWidth

Buttons
- Index
- Insert / Remove
- Caption

Description
Key
Value (Unpressed,Pressed)
Stule (Button,Check,ButtonGroup,Separator,PlaceHolder)
Tag
ToolTipText
Image (Number iz ImageListe)
Visible
Enabled
MixedState
AllowCustomize property
Button.Caption / Tag / Key / Index

Kontrola: StatusBar

General
- Style (Multiple panels, Single panel simple text)
- SimpleText
- Enabled

Panels
- Index
- Insert / Remove
- Text
- Key
- Alignment
- Style (Text,Caps,NumLock,Ins,Scrool,Time,Date,KANA LOCK)
- Bevel (None,Inset,Raised)
- Autosize (None,Spring,Contents)
- MinimumWidth
- ActualWidth
- Enabled
- Visible
- Picture (Browse / No Picture)

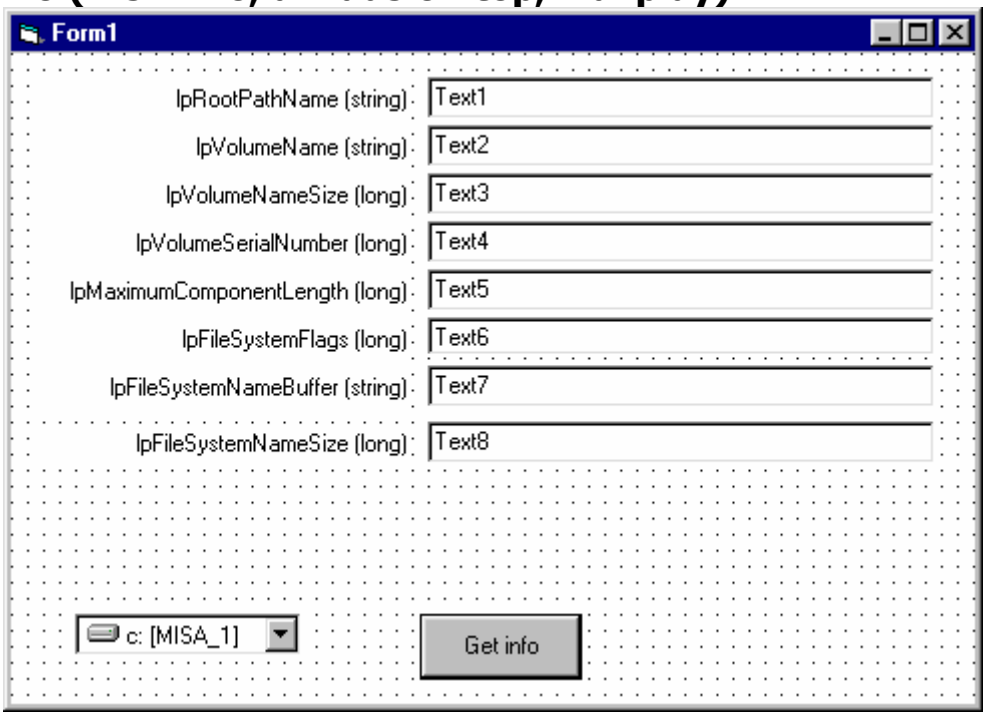
ToolBar1.Panels(1).Text="String"

Veza sa eksternim DLL (Dynamic Link Libraries) bibliotekama

Primer – Windows sistemski direktorijum:

```
Declare Function GetWindowsDirectoryA Lib "kernel32" _
(ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Sub Command1_Click()
Dim a As String * 255
Dim x As Long
x = GetWindowsDirectoryA(a, Len(a))
MsgBox "Windows direktorijum: " & a, 64
End Sub
```

API Demo (Disk info, alwaus on top, wavplay)




```
Declare Function GetVolumeInformation Lib "kernel32" Alias
"GetVolumeInformationA" _
(ByVal lpRootPathName As String, ByVal lpVolumeNameBuffer As String, _
ByVal nVolumeNameSize As Long, lpVolumeSerialNumber As Long, _
lpMaximumComponentLength As Long, lpFileSystemFlags As Long, _
ByVal lpFileSystemNameBuffer As String, ByVal nFileSystemNameSize As Long) As
Long

Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long,
ByVal cy As Long, ByVal wFlags As Long) As Long

Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal
lpszSoundName As String, ByVal uFlags As Long) As Long
'1 asinhrono, 2 sinhrono



Private Sub cmdGetInfo_Click()
      Dim lpRootPathName As String
      Dim lpVolumeName As String * 255
      Dim lpVolumeNameSize As Long
      Dim lpVolumeSerialNumber As Long
      Dim lpMaximumComponentLength As Long
      Dim FileSystemFlags As Long
      Dim FileSystemNameBuffer As String * 255
      Dim nFileSystemNameSize As Long
      Dim nIzlaz As Long

      lpRootPathName = Left(Drive1.Drive, 2) & "\"
      lpVolumeNameSize = Len(lpVolumeName)
      nFileSystemNameSize = Len(FileSystemNameBuffer)
      nIzlaz = GetVolumeInformation(lpRootPathName, lpVolumeName,
lpVolumeNameSize, _
      lpVolumeSerialNumber, lpMaximumComponentLength, FileSystemFlags, _
      FileSystemNameBuffer, nFileSystemNameSize)
```

```
      Text1.Text = lpRootPathName
      Text2.Text = lpVolumeName
      Text3.Text = lpVolumeNameSize
      Text4.Text = lpVolumeSerialNumber
      Text5.Text = lpMaximumComponentLength
      Text6.Text = FileSystemFlags
      Text7.Text = FileSystemNameBuffer
      Text8.Text = nFileSystemNameSize
End Sub


Private Sub Form_Load()
      SetWindowPos Me.hwnd, -1, 0, 0, 0, 3 'ili 1 ili 2 ili 3
      'ako je 1; pozicija in pix
End Sub
```