POGLAVLJE

2

Visual Basic projekti

U prethodnom poglavlju predstavljeni su Visual Studio IDE, Toolbox - kutija sa alatima i principi programiranja rukovođenog događajima. U ovom poglavlju proširujemo ovo predstavljanje jezika izgradnjom nekih "pravih" aplikacija. Između ostalog, naučićete kako se pišu aplikacije koje vrše proveru unosa korisnika i rutine za rukovanje greškama. Takođe ćemo predstaviti nekoliko tehnika koje će biti potrebne u aplikacijama koje razvijamo u nastavku knjige. U poslednjem delu poglavlja naučićete kako da distribuirate aplikaciju sa pravilnim Windows instalerom (program koji instalira aplikaciju na ciljnu mašinu).

U većem delu poglavlja biće demonstrirane najosnovnije programerske tehnike, kao što su izgradnja korisničkih interfejsa, programiranje događaja, provera korisničkog ulaza i rukovanje greškama. Cilj je da se shvatite kako da pišete jednostavne aplikacije korišćenjem najosnovnijih elemenata jezika. U ovom poglavlju će biti objašnjena metodologija građenja aplikacija. Prilično jednostavan kod aplikacije će demonstrirati osnove provere unetih podataka i hvatanja grešaka.

Ako ste početnik, možete razmišljati ovako: "Sve što sada želim je da napišem jednostavnu aplikaciju koja radi - za proveru unetih podataka ću se pobrinuti kasnije". Nikada nije suviše rano početi razmišljanje o proveri podataka za kod i o hvatanju grešaka. Kao što ćete videti, obezbeđivanje da aplikacija ne pada može da zahteva više koda od stvarnih operacija koje izvršava! Aplikacija koja se dobro ponaša mora da uhvati i rukuje svakom greškom na skladan način, uključujući i greške korisnika.

Izgradnja kalkulatora zajma

Jedna praktična aplikacija laka za implementaciju je program koji proračunava parametre zajma. Visual Basic obezbeđuje ugrađene funkcije za izvršavanje mnogo tipova finansijskih proračuna i potrebna Vam je samo jedna linija koda za izračunavanje mesečne rate za konkretni iznos zajma, rok otplate i kamatu. Za dizajniranje korisničkog interfejsa, međutim, potrebno je mnogo više truda.

Bez obzira na jezik koji koristite, da biste razvili aplikaciju, morate proći kroz proces koji se sastoji od:

1. odlučivanja šta će aplikacija da radi i kako će stupati u interakciju sa korisnikom
2. dizajniranja korisničkog interfejsa prema zahtevima iz koraka 1
3. pisanja koda iza događaja kojim želite da rukujete.

Kako Loan aplikacija radi?

Praćenjem prvog koraka procesa, vi odlučujete da korisnik treba da može da definiše iznos zajma, kamatu i trajanje zajma u mesecima. Morate, stoga, obezbediti tri TextBoxa gde korisnik može da unese ove vrednosti.

Drugi parametar koji utiče na mesečna plaćanja je da li se uplate vrše na početku, ili na kraju svakog meseca, tako da morate da obezbedite način da korisnik odredi da li će uplate biti rane (prvi dan u mesecu), ili kasne (poslednji dan u mesecu). Najprigodniji tip kontrole za ovo je unošenje Da/Ne, ili True/False tipa informacije pomoću CheckBox kontrole. Ova kontrola je prekidač: ako je potvrđena, možete je obrisati klikom na nju. Ako nije potvrđena, možete je potvrditi ako je ponovo kliknete. Korisnik ne unosi nikakve podatke u ovu kontrolu (što znači da ne morate da očekujete greške korisnika sa ove kontrole) i to je najjednostavniji metod za definisanje vrednosti sa dva moguća stanja. Na slici 2.1 prikazan je korisnički interfejs koji se podudara sa našom specifikacijom dizajna. Ovo je glavna forma LoanCalculator projekta, koju ćete naći u fascikli ovog poglavlja na CD-u.

SLIKA 2.1 *LoanCalculator je jednostavna finansijska aplikacija.*

Pošto korisnik unese sve informacije na formu, može da klikne dugme Show Payment da bi bila proračunata mesečna uplata. Program će proračunati mesečnu uplatu i prikazati je u donjoj TextBox kontroli. Čitava akcija se dešava u Click podrutini dugmeta. Funkcija za proračunavanje mesečnih plaćanja se naziva Pmt() i mora biti pozvana ovako:

```
MonthlyPayment = Pmt(InterestRate, Periods, Amount, FutureValue, Due)
```

Kamata (argumenat InterestRate) je definisana kao mesečna. Ako ona iznosi 16,5 odsto, vrednost koju je uneo korisnik u Interest Rate boks treba da je 16,5, a mesečna rata će biti 0,165/12. Trajanje zajma (*Periods*) je definisano brojem meseci, a Amount je iznos zajma. *FutureValue* vrednost zajma je nula (bila bi pozitivna vrednost za investiciju), a poslednji parametar Due definiše kada plaćanje dospeva.

Vrednost Due može da bude jedna od konstanti `DueDate.BegOfPeriod` i `DueDate.EndOfPeriod`. Ove dve konstante su ugrađene u jezik i možete ih koristiti, a da ne znate njihovu tačnu vrednost. To je suština korišćenja imenovanih konstanti: kucate ime koje samo sebe objašnjava i ostavljate VB-u da ga konvertuje u numeričku vrednost. Kao što ćete videti, .NET koristi brojne konstante, kategorisane u grupe nazvane *enumerations* - nabrajanja. Konstante koje se primenjuju na Due argumenat Pmt() funkcije pripadaju DueDate() nabrajanju, koja ima dva člana: `BegOfPeriod` i `EndOfPeriod`.

Sadašnja vrednost zajma je iznos sa negativnim znakom. Ona je negativna, jer nemate novac sada. Vi ga pozajmljujete; to je novac koji dugujete banci. Buduća vrednost predstavlja vrednost nečega u definisanom vremenskom trenutku - u ovom slučaju koliko će zajam vredeti kada bude otplaćen. Ovo je ono što jedna strana duguje drugoj na kraju definisanog perioda. Na osnovu toga, konstatujemo da je buduća vrednost zajma nula.

Pmt() je ugrađena funkcija koja koristi pet vrednosti u zagradama za proračunavanje mesečne rate. Vrednosti prosleđene funkciji nazivaju se argumenti. Argumenti su vrednosti koji su potrebni funkciji (ili podrutini) za izvođenje akcije, ili proračuna. Prosleđivanjem različitih vrednosti funkciji korisnik može da definiše parametre bilo kog zajma i da proračuna njegove mesečne rate. Funkcija Pmt() i druge finansijske funkcije Visual Basica su objašnjene u tekstu "VB.NET Funkcije i iskazi" na CD-u koji ste dobili uz ovu knjigu.

Vi ne morate da znate kako Pmt() funkcija proračunava mesečnu ratu. Ova funkcija izvršava proračune i vraća rezultat. Da biste proračunali mesečnu ratu zajma od 25.000 dolara, sa kamatom od 14,5 odsto, koji treba otplatiti za 48 meseci, sa dospećem u poslednjem danu perioda za plaćanje (što je u našem slučaju mesec), pozovite Pmt() funkciju:

```
Console.WriteLine(Pmt(0.145 / 12, 48, -25000, 0, DueDate.EndOfPeriod))
```

Vrednost 689,448821287218 će biti prikazana u Output prozoru (videćete kasnije kako možete da ograničite cifre posle decimalnog zareza na dva, pošto je to sva preciznost koja je potrebna za dolarski iznos). Primetićete znak za negativno ispred *Amount* argumenta u iskazu. Ako definišete pozitivan iznos, rezultat će biti negativno plaćanje. Plaćanje i iznos zajma imaju različite znakove, jer predstavljaju različit tok novca. Iznos zajma je novac koji dugujete banci, dok je plaćanje novac koji plaćate banci.

Poslednja dva argumenta Pmt() funkcije su opciona. Ako ih izostavite, Visual Basic koristi podrazumevane vrednosti, koje su 0 za *FutureValue* argumenat i DueDate.BegOfPeriod za Due argumenat. Možete potpuno izostaviti ove argumente i pozvati Pmt() funkciju ovako:

```
Console.WriteLine(Pmt(0.145 / 12, 48, -25000))
```

Proračunavanje iznosa mesečne rate sa konkretnim parametrima zajma je sasvim jednostavno. Ono što treba da znate, ili razumete su parametri zajma i kako da ih prosledite Pmt() funkciji. Morate, takođe, znati kako je definisana kamata da biste izbegli pojedinačne vrednosti. Ono što nije potrebno da znate je kako se rata proračunava - to za Vas radi Visual Basic. Ovo je suština funkcija: one su "crne kutije" koje izvode komplikovane proračune na svojim argumentima i vraćaju rezultat. Ne morate da znate kako one rade, već samo kako da unesete vrednosti potrebne za proračune.

Dizajniranje korisničkog interfejsa

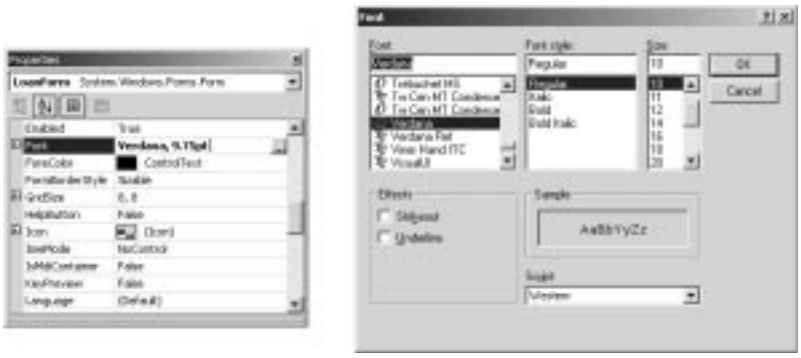
Sada, kada znate kako da proračunate mesečnu ratu, možete dizajnirati korisniči interfejs. Da biste to uradili, započnite novi projekat, nazovite ga LoanCalculator i preimenujte njegovu formu u LoanForm. Forma i fajlovi projekta mogu se naći u fascikli ovog poglavlja na CD-u.

Vaš prvi zadatak je da se odlučite za font i veličinu teksta koji ćete koristiti za većinu kontrola na formi. Iako nećemo ništa direktno prikazivati na formi, sve kontrole koje postavljamo na nju će podrazumevano imati isti font kao i ona. Forma je kontejner kontrola - one nasleđuju neka od njenih svojstava, kao što je Font. Možete da promenite font kasnije, za vreme dizajniranja, ali je bolje započeti sa odgovarajućim fontom. U svakom slučaju ne pokušavajte da poravnate kontrole ako planirate da promenite njihove fontove. Ovo će, najverovatnije, poništiti Vaš trud da poravnate kontrole.

SAVET

Pokušajte da ne pomešate fontove na formi. Forma, ili štampana strana, koja uključuje nekoliko vrsta fonta, izgleda kao da je kreirana slučajno i teška je za čitanje. Međutim, možete koristiti različite veličine za neke od kontrola na formi. ■

Loan aplikacija koju ćete naći na CD-u koristi Verdana font sa 10 tačaka. Da biste to promenili, izaberite mišem formu, duplo kliknite ime Font svojstva u Properties prozoru da biste otvorili Font okvir za dijalog i izaberite željeni font i atribute. Kada je forma izabrana, njeno ime se pojavljuje u ComboBoxu na vrhu prozora, kao što je prikazano na slici 2.2.



SLIKA 2.2 *Podešavanje Font svojstva forme*

Da biste dizajnirali formu prikazanu na slici 2.1, sledite ove korake:

1. Postavite četiri Label kontrole na formu i pridružite im sledeće natpise:

Label	Caption – natpis
Label1	Loan Amount - iznos zajma
Label2	Duration (u mesecima) - trajanje
Label3	Interest Rate - kamata
Label4	Monthly Payment - mesečna rata

   Label kontrole treba da su dovoljno velike da prikažu natpise na sebi. Ne morate da promenite podrazumevana imena četiri Label kontrole na formi, jer su njihovi natpisi sve što Vam je potrebno. Vi ih nećete programirati.

2. Postavite TextBox kontrolu pored svake Label kontrole. Postavite njihova Name i Text svojstva na sledeće vrednosti. Ove inicijalne vrednosti odgovaraju zajmu od 25.000 dolara, sa kamatom od 14,5 odsto i periodom otplate od 48 meseci.

TextBox	Ime	Text
TextBox1	txtAmount	25.000
TextBox2	txtDuration	48
TextBox3	txtRate	14,5
TextBox4	txtPayment	

3. Četvrta TextBox kontrola je gde će se pojaviti mesečna rata. Korisnik ne unosi nikakve podatke u ovaj boks, tako da morate da podesite njegovo ReadOnly svojstvo na True. Moći ćete da promenite njegovu vrednost iz Vašeg koda, ali korisnici neće moći da štampaju bilo šta u njega (možete da koristite Label kontrolu, ali se češće koristi TextBox, zbog svog uobičajenog izgleda).

4. Sledeće, postavite CheckBox kontrolu na formu. Natpis kontrole je podrazumevano Check1 i pojavljuje se sa desne strane polja za potvrdu. Zbog toga što želimo da natpisi budu sa leve strane od odgovarajućih kontrola, promenićemo ovaj podrazumevani prikaz.

5. Izaberite polje za potvrdu pomoću miša (ako već nije izabrano) i u Properties prozoru pronađite CheckAlign svojstvo. Njegova vrednost je `MiddleLeft`. Ako proširite padajuću listu klikom na dugme sa strelicom, videćete da ovo svojstvo ima mnoga različita podešavanja (svako je prikazano kao kvadratić). Izaberite kvadratić u srednjem redu desne kolone. String `MiddleRight` će se pojaviti u boksu svojstva kada kliknete odgovarajuće dugme. Prva komponenta vrednosti CheckAlign svojstva pokazuje vertikalno poravnanje polja za potvrdu, a druga horizontalno poravnanje. `MiddleRight` znači da polje za potvrdu treba da bude centrirano vertikalno i poravnato udesno po horizontali.

6. Sa izabranim poljem za potvrdu, pronađite Name svojstvo u Properties prozoru i podesite ga na chkPayEarly.

7. Promenite natpis CheckBoxa unošenjem stringa Early Payment u njegovo Text polje svojstva.

8. Postavite Button kontrolu u donjem levom uglu forme. Nazovite je bttnShowPayment i podesite njen natpis na Show Payment.

9. Na kraju, postavite drugu Button kontrolu na formu, nazovite je bttnExit i podesite njeno Text svojstvo na Exit.

Poravnavanje kontrola

Vaš sledeći korak je da poravnate kontrole na formi. Prvo, proverite da li su natpisi na Label kontrolama vidljivi. Naši natpisi su dugački i ako Label kontrole ne budu dovoljno dugačke, deo natpisa se može naći na drugoj liniji i postati nevidljiv.

SAVET

Proverite da li su Vaše Label kontrole dovoljno dugačke da čuvaju svoje natpise, naročito ako koristite nestandardni font. Računar korisnika može da supstituiše nestandardni font sa drugim fontom i odgovarajući natpisi mogu da postanu duži. ■

IDE obezbeđuje komande za poravnavanje kontrola na formi, od kojih se svima može pristupiti pomoću Format menija. Da biste poravnali kontrole koje su već na formi LoanForm, sledite ove korake:

1. Izaberite četiri Label kontrole na formi sa mišem i poravnajte ih po levoj strani biranjem Format➥Align➥Lefts. Kvadratići koji obuhvataju kontrole će biti beli, izuzev jedne kontrole gde će biti crni. Sve kontrole će biti poravnate po levoj strani sa ovom kontrolom. Da biste definisali kontrolu koja će biti korišćena kao referentna tačka za poravnavanje drugih kontrola, kliknite je pošto ste napravili izbor (možete izabrati više kontrola crtanjem pravougaonika koji ih obuhvata pomoću miša, ili klikom na svaku kontrolu dok držite pritisnut Ctrl taster.)
2. Sa četiri izabrana TextBoxa, izaberite Format➥Align➥Lefts. Nemojte uključiti CheckBox u ovaj izbor.

SAVET

Kada izaberete više kontrola za grupno poravnavanje, koristite kontrolu sa crnim kvadratićima kao vodilju za poravnavanje drugih kontrola.

3. Sa sva četiri TextBoxa još uvek izabrana, koristite miš da ih poravnate iznad i ispod boksa CheckBox kontrole.

Vaša forma treba sada da izgleda kao ona sa slike 2.1. Dobro je pogledajte i proverite da vidite da li je neka od kontrola loše poravnata. U procesu dizajniranja interfejsa dešava se da se previde mali problemi, kao što su mala odstupanja u poravnanju kontrola. Korisnik aplikacije, međutim, odmah primeti takve greške. Nije uopšte važno koliko dobro su poravnate ostale kontrole na formi; ako jedna od njih nije dobro poravnata, ona će privući pažnju korisnika.

Programiranje Loan aplikacije

Sada pokrenite aplikaciju i pogledajte kako se ona ponaša. Unesite nekoliko vrednosti u TextBoxeve, promenite stanje polja za potvrdu i testirajte funkcionalnost već ugrađenu u aplikaciji. Klik na Show Payment neće imati nikakvog efekta, jer još niste dodali bilo kakav kod. Ako ste zadovoljni korisničkim interfejsom, zaustavite aplikaciju, otvorite formu i duplo kliknite Show Payment Button kontrolu. Visual Basic otvara prozor koda i prikazuje definiciju Show_Payment_Click događaja:

```
Private Sub bttnShowPayment_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnShowPayment.Click
End Sub
```

NAPOMENA

Prelomio sam prvu liniju sa donjom crticom, jer ne bi stala na strani. Karakter donja crtica je karakter za nastavak linije, koji omogućava da prelomite dugačke linije koda u više linija teksta. ■

Ovo je deklaracija hendlera Click događaja Button kontrole. Ova podrutina će biti pozvana kada korisnik klikne Show Payment dugme. Iznad definicije hendlera događaja videćete dva sledeća iskaza:

```
Public Class LoanForm
   Inherits System.Windows.Forms.Form
```

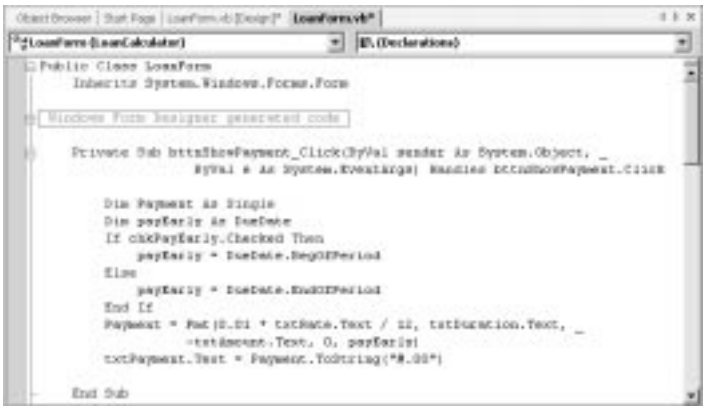
Prvi iskaz kreira novu klasu za formu projekta, a drugi nasleđuje funkcionalnost Form objekta. Ove iskaze postavlja IDE i ne treba ih menjati. Kada naučite više o klasama i nasleđivanju, koji su predstavljeni u drugom delu knjige, moći ćete da bolje razumete ulogu ovih iskaza.

Postavite pokazivač između linija `Private Sub` i `End Sub` i unesite ostatak linija listinga 2.1 (ne morate da ponovo unesete prvu i poslednju liniju koje deklarišu hendler događaja).

Listing 2.1: Dugme Show Payment

```
Private Sub bttnShowPayment_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnShowPayment.Click
   Dim Payment As Single
   Dim payEarly As DueDate
   If chkPayEarly.Checked Then
      payEarly = DueDate.BegOfPeriod
   Else
      payEarly = DueDate.EndOfPeriod
   End If
   Payment = Pmt(0.01 * txtRate.Text / 12, txtDuration.Text, _
                 -txtAmount.Text, 0, payEarly)
   txtPayment.Text = Payment.ToString("#.00")
End Sub
```

Prozor koda treba sada da izgleda kao onaj prikazan na slici 2.3. Primetićete karakter crtica dole na kraju prvog dela dugačke linije. Donja crtica omogućava da prelomite dugačke linije, tako da one stanu u prozor koda. Koristim dosta ovu konvenciju u ovoj knjizi da bih stavio dugačke linije na štampanu stranu. Isti iskaz koji vidite kao više linija u knjizi može se pojaviti u jednoj dugačkoj liniji projekta.



SLIKA 2.3 *Podrutina Click događaja Show Payment dugmeta*

Ne morate da prelamate dugačke linije ručno dok unosite kod u prozor editora. Otvorite Edit meni i izaberite Advanced➥Word Wrap. Editor će automatski kod dugačkih linija pomerati pregled udesno. Dok je ova opcija uključena, znak za potvrdu će se pojaviti ispred komande Edit➥Advanced➥Word Wrap. Da biste isključili ovu opciju, izaberite istu komandu ponovo.

U listingu 2.1 prva linija koda u podrutini deklariše promenljivu. Omogućava aplikaciji da zna da je *Payment* mesto za čuvanje broja sa pokretnim zarezom (*broj sa decimalnim delom*) - Single tipa podataka. Druga linija deklariše promenljivu tipa DueDate. Ovo je tip argumenta koji određuje da li se plaćanje vrši na početku, ili na kraju meseca. Poslednji argumenat Pmt() funkcije mora biti promenljiva ovog tipa, tako da deklarišete promenljivu DueDate tipa. Kao što je ranije pomenuto u ovom poglavlju, DueDate je nabrajanje sa dva člana: BegOfPeriod i EndOfPeriod. Ukratko, poslednji argumenat Pmt() funkcije može da bude jedna od sledećih vrednosti:

```
DueDate.BegOfPeriod
DueDate.EndOfPeriod
```

Prva stvarno izvršna linija u podrutini je If iskaz koji pregleda vrednost chkPayEarly CheckBox kontrole. Ako je kontrola potvrđena, kod podešava *payEarly* promenljivu na `DueDate.BegOfPeriod`. Ako nije, kod podešava istu promenljivu na `DueDate.EndOfPeriod`. Checked svojstvo ComboBox kontrole vraća True ako je kontrola potvrđena u tom trenutku; u suprotnom, vraća False. Posle podešavanja vrednosti payEarly promenljive, kod poziva Pmt() funkciju prosleđivanjem vrednosti kontrola kao argumenata:

- Prvi argumenat je kamata. Vrednost koju unosi korisnik u txtRate TextBoxu se množi sa 0,01, tako da se vrednost 14,5 (što odgovara 14,5 odsto) prosleđuje Pmt() funkciji kao 0,145. Iako više volimo da određujemo kamatu celim brojevima (osam odsto), ili brojevima sa pokretnim zarezom većim od 1 (8,24 odsto), Pmt() funkcija očekuje da čita broj manji od 1. Vrednost 1 odgovara 100 odsto. Stoga, vrednost 0,1 je 10 odsto. Ova vrednost se deli sa 12 da bi se dobila mesečna kamata.
- Drugi argumenat je trajanje zajma u mesecima (vrednost uneta u txtDuration TextBoxu).

- Treći argumenat je iznos zajma (vrednost uneta u txtAmount TextBox).
- Četvrti argumenat (buduća vrednost zajma) je, po definiciji, 0.
- Poslednji argumenat je *payEarly* promenljiva, koja je podešena prema statusu chkPayEarly kontrole.

Sledeća dva iskaza konvertuju numeričku vrednost koju je vratila Pmt() funkcija u string i prikazuju ovaj string u četvrtoj TextBox kontroli. Rezultat je odgovarajuće formatiran sa sledećim izrazom:

```
Payment.ToString("#.00")
```

Payment promenljiva je numerička; sve numeričke promenljive obezbeđuju metod ToString, koji formatira numeričku vrednost i konvertuje je u string. Karakter # menja celobrojni deo promenljive. Tačka odvaja celobrojni od dela iza zareza, koji je zaokružen na dve decimale. Zbog toga što Pmt() funkcija vraća precizan broj, kao što je 372,2235687646345, morate da ga zaokružite i formatirate pre prikazivanja. Pošto banka ne može zameniti ništa manje od penija, nija potrebna ekstremna tačnost. Dve cifre iza decimalnog zareza su dovoljne. Za više informacija o formatiranju numeričkih (i drugih) vrednosti pogledajte odeljak "Formatiranje brojeva" u Poglavlju 3.

Da biste direktno na *txtPayment* TextBox kontroli prikazali rezultat koji je vratila Pmt() funkcija, koristite sledeći iskaz:

```
txtPayment.Text = Pmt(0.01 * txtRate.Text / 12, txtDuration.Text, _
                      -txtAmount.Text, 0, payEarly)
```

Ovaj iskaz pridružuje vrednost koju je vratila funkcija Pmt() direktno Text svojstvu kontrole. Mesečna rata će biti prikazana sa četiri decimalne cifre, ali ovo nije pravilan dolarski iznos.

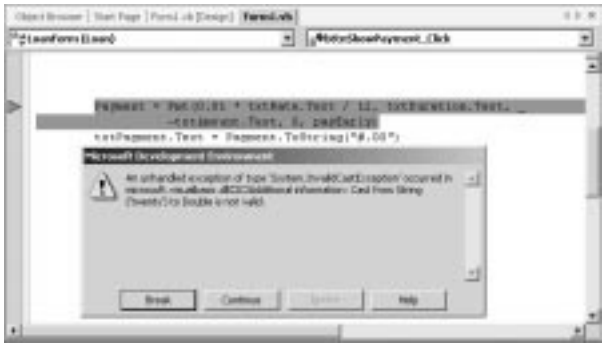
SAVET

Skoro uvek koristite ToString metod, ili Format() funkciju kada želite da prikažete rezultate numeričkih proračuna, jer najčešće Vam nije potrebna ekstremna tačnost Visual Basica. Nekoliko cifara iza zareza je sve što je potrebno. Osim brojeva, ToString metod može da formatira datume i vreme. Mogućnosti za formatiranje ToString metoda su obrađene u Poglavlju 12, a Format() funkcija je objašnjena u tekstu "VB.NET Funkcije i iskazi" na propratnom CD-u. ■

Kod LoanCalculator projekta na CD-u se razlikuje i značajno je duži od onoga što sam predstavio ovde. Iskazi obrađeni u prethodnom tekstu su minimum za proračunavanje isplate zajma. Korisnik može da unese bilo koje vrednosti na formu i izazove pad programa. U sledećem odeljku videćete kako možete da proverite podatke koje je uneo korisnik, da uhvatite greške i da skladno sa njima rukujete (to jest, date korisniku šansu da ispravi podatke i nastavi rad), nasuprot završavanju aplikacije sa greškom za vreme izvršavanja.

Provera vrednosti podataka

Ako unesete nenumeričku vrednost u jedno od polja, program će pasti i prikaziti poruku o grešci. Na primer, ako unesete reč dvadeset u Duration TextBox, program će prikazati poruku o grešci prikazanu na slici 2.4. Jednostavna greška u kucanju može da obori program. To nije način na koji treba da radi Windows aplikacija. Vaša aplikacija mora da bude osposobljena da rukuje većinom grešaka korisnika, da obezbedi poruke pomoći i da vodi korisnika kroz efikasno izvršavanje aplikacije. Ako greška korisnika prođe neprimećeno, Vaša aplikacija će neočekivano prekinuti da radi, ili će proizvesti netačne rezultate bez indikacije.



SLIKA 2.4 *Cast Exception poruka znači da ste uneli string gde je očekivana numerička vrednost.*

Kliknite Break dugme i Visual Basic će Vas vratiti nazad u prozor koda aplikacije, gde će iskazi koji su prouzrokovali grešku biti istaknuti zelenom bojom. Očigledno je da moramo nešto uraditi u vezi sa korisnikovim greškama. Jedan od načina da se pobrinemo za one koje nastaju pri kucanju je da pregledamo sadržaj svake kontrole - ako ne sadrže važeće numeričke vrednosti, prikažite sopstvene opisne poruke i dajte korisniku drugu šansu. Listing 2.2 je hendler Click dugmeta posle revizije, koji pregleda vrednost svakog TextBoxa pre pokušaja da ga koristi za bilo koji proračun.

Listing 2.2: Show Payment dugme posle revizije

```
Private Sub bttnShowPayment_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnShowPayment.Click
   Dim Payment As Single
   Dim LoanIRate As Single
   Dim LoanDuration As Integer
   Dim LoanAmount As Integer
' Validate amount
   If IsNumeric(txtAmount.Text) Then
      LoanAmount = txtAmount.Text
   Else
      MsgBox("Please enter a valid amount")
      Exit Sub
   End If
' Validate interest rate
```

```
    If IsNumeric(txtRate.Text) Then
       LoanIRate = 0.01 * txtRate.Text / 12
    Else
       MsgBox("Invalid interest rate, please re-enter")
       Exit Sub
    End If
' Validate loan's duration
    If IsNumeric(txtDuration.Text) Then
       LoanDuration = txtDuration.Text
    Else
       MsgBox("Please specify the loan's duration as a number of months")
       Exit Sub
    End If
' If all data were validated, proceed with calculations
    Dim payEarly As DueDate
    If chkPayEarly.Checked Then
       payEarly = DueDate.BegOfPeriod
    Else
       payEarly = DueDate.EndOfPeriod
    End If
    Payment = Pmt(LoanIRate, LoanDuration, -LoanAmount, 0, payEarly)
    txtPayment.Text = Payment.ToString("#.00")
End Sub
```

Prvo, deklarišemo tri promenljive u kojima će parametri zajma biti čuvani: *LoanAmount*, *LoanIRate* i *LoanDuration*. Ove vrednosti će biti prosleđene Pmt() funkciji kao argumenti. Svaka vrednost TextBoxa je pregledana sa jednom If strukturom. Ako odgovarajući TextBox čuva važeći broj, njegova vrednost se pridružuje numeričkoj promenljivoj. Ako ne čuva, program prikazuje upozorenje i izlazi iz podrutine, bez pokušaja da proračuna mesečnu ratu. Korisnik može da ispravi netačnu vrednost i klikne Show Payment dugme ponovo. IsNumeric() je druga ugrađena funkcija koja prihvata promenljivu i vraća True ako je promenljiva broj; u suprotnom, vraća False.

Ako Amount TextBox čuva numeričku vrednost, kao što je 21.000, ili 21,50, funkcija `IsNumeric(txtAmount.Text)` vraća True i iskaz koji sledi se izvršava. Iskaz koji sledi pridružuje unetu vrednost u *Amount* TextBoxu promenljivoj *LoanAmount*. Ako ne čuva, izvršava se Else deo iskaza, koji prikazuje poruku upozorenja, pa napušta podrutinu. `Exit Sub` iskaz "kaže" Visual Basicu da odmah zaustavi izvršavanje podrutine, kao kada se naiđe na `End Sub` liniju.

Možete pokrenuti aplikaciju posle revizije i testirati je unošenjem nevažećih vrednosti u polja. Primetićete da ne možete da definišete nevažeću vrednost za poslednji argumenat; CheckBox kontrola neće dopustiti da unesete vrednost. Možete je samo potvrditi, ili poništiti i obe opcije su važeće. LoanCalculator aplikacija koju ćete naći na CD-u sadrži ovu poslednju verziju sa kodom za hvatanje grešaka.

Stvarni proračun mesečne rate zauzima jednu liniju Visual Basic koda. Njegov prikaz zahteva još jednu liniju koda. Dodavanje koda za proveru podataka koje je uneo korisnik, međutim, predstavlja čitav program.

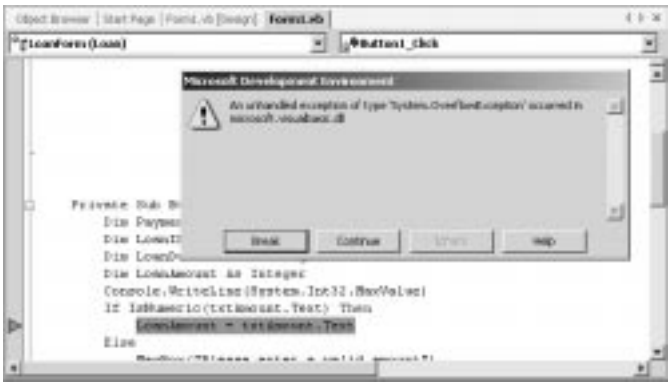
NAPOMENA

Aplikacije u ovoj knjizi ne sadrže mnogo koda za proveru podataka, jer bi to sakrilo "koristan" kod koji obrađuje temu o kojoj se raspravlja. Umesto toga, one demonstriraju specifične tehnike. Možete koristiti delove primera u Vašim aplikacijama, ali treba da obezbedite sopstveni kod za proveru podataka (kod za rukovanje greškama, kao što ćete videti u sledećem odeljku). ■

PISANJE APLIKACIJA KOJE SE "LEPO PONAŠAJU"

Aplikacija koja se "lepo ponaša" mora da sadrži kod za proveru podataka. Ako aplikacija, kao što je LoanCalculator, padne zbog greške u kucanju, neće se ništa mnogo loše desiti. Korisnik će pokušati ponovo, ili će, u suprotnom, odustati od Vaše aplikacije i tražiti neku koja je "profesionalnija". Međutim, ako je korisnik unosio podatke satima, situacija je mnogo ozbiljnija. Vaša je odgovornost kao programera da proverite da li aplikacija koristi samo važeće podatke i da li nastavlja da radi, bez obzira da li je korisnik pogrešno koristi, ili zloupotrebljava.

Sada pokrenite aplikaciju poslednji put i unesite ogroman iznos zajma. Pokušajte da pronađete koliko je potrebno da se otplati nacionalni dug sa razumnom kamatom u roku od, na primer, 72 meseca. Program će pasti ponovo (kao da niste znali). Ovoga puta program će pasti sa različitom porukom o grešci. Visual Basic će se "žaliti" na overflow. Tačna poruka je prikazana na slici 2.5 i program će stati na liniji koja pridružuje sadržaj *txtAmount* TextBoxa promenljivoj *LoanAmount*. Pritisnite Break dugme i iskaz koji je izazvao grešku će biti istaknut.



SLIKA 2.5 *Vrlo velike vrednosti mogu da izazovu da aplikacija padne sa ovom porukom o grešci.*

SAVET

Overflow je numerička vrednost prevelika da bi program njome rukovao. Ova greška se obično pojavljuje kada podelite broj sa vrlo malom vrednošću. Kada pokušate da pridružite vrlo veliku vrednost Integer promenljivoj, takođe ćete dobiti overflow (prekoračenje) izuzetak. ■

Zapravo, u LoanCalculator aplikaciji bilo koji iznos veći od 2.147.483.647 će izazvati overflow uslov. Ovo je najveća vrednost koju možete da pridružite Integer promenljivoj; to je dovoljno za Vaše potrebe u banci, ali ni blizu dovoljno za rukovanjem budžetom države. Kao što ćete videti u sledećem poglavlju, Visual Basic obezbeđuje druge tipove promenljivih koje mogu da čuvaju ogromne vrednosti. U međuvremenu, ako želite da koristite LoanCalculator, promenite deklaraciju *LoanAmount* promenljive na:

```
Dim LoanAmount As Single
```

Single tip podataka može da čuva mnogo veće vrednosti. Pored toga, on može da čuva necelobrojne vrednosti. Pretpostavljam da nećete tražiti zajam od 25.000 dolara i nekoliko centi, ali, ako želite da proračunate precizno mesečnu ratu za dug koji ste akumulirali, treba da možete da definišete iznos koji nije celobrojan. Trebalo je odmah da deklarišem *LoanAmount* promenljivu sa Single tipom podatka (ali onda ne bih mogao da demonstriram izuzetak prekoračenja vrednosti).

Greška prekoračenja vrednosti ne može biti "uhvaćena" kodom za proveru vrednosti podataka. Postoji uvek mogućnost da Vaša kalkulacija proizvede prekoračenje, ili druge tipove matemetičkih grešaka. Ovde ne pomaže provera podataka; Vi, jednostavno, ne znate rezultat pre nego što izvršite proračune. Potrebno je nešto što se zove rukovanje greškama (*error handling*), ili hvatanje grešaka (*error trapping*). Ovo je dodatni kod koji može da rukuje greškama pošto se one pojave. Posledica toga je da "kažete" VB-u da ne treba da stopira poruku o grešci. To bi za Vas bilo neprijatno i ni malo ne bi pomoglo korisniku. Umesto toga, VB treba da detektuje grešku i da izvrši pravilne iskaze koji će rukovati njom. Očigledno je da morate da obezbedite ove iskaze, a u sledećem odeljku videćete primere rukovanja greškama za vreme izvršavanja.

Izgradnja matematičkog kalkulatora

Naša sledeća aplikacija je naprednija, ali ne toliko napredna koliko izgleda da jeste. To je matematički kalkulator sa tipičnim vizuelnim interfejsom, koji demonstrira kako Visual Basic može da pojednostavi programiranje prilično naprednih operacija. Ako to niste pokušali, možete da pomislite da je pisanje aplikacije kao što je ova previše komplikovano, ali nije. MathCalculator aplikacija je prikazana na slici 2.6 i naći ćete je u fascikli ovog poglavlja na propratnom CD-u. Aplikacija simulira rad ručnog kalkulatora i implementira osnovne aritmetičke operacije. Ima strukturu matematičkog kalkulatora i možete je lako proširiti dodavanjem osobina. Dodavanje osobina kao što su kosinus i logaritmi je jednostavnije od izvršavanja osnovnih aritmetičkih operacija.

SLIKA 2.6 *Prozor Calculator aplikacije*

Dizajniranje korisničkog interfejsa

Interfejs aplikacije je jednostavan, ali je potrebno dosta napora da se napravi. Morate da poravnate dugmad na formi i da napravite kalkulator koji će izgledati kao i svi ostali ručni kalkulatori. Započnite novi projekat (MathCalculator) i nazovite njegovu glavnu formu CalculatorForm.

Dizajniranje interfejsa aplikacije nije jednostavno, jer se on sastoji od mnogo dugmadi, koja su perfektno poravnata na formi. Da biste pojednostavili dizajn, sledite ove korake:

1. Izaberite font koji želite za formu. Sva Command dugmad koju postavite na formu će naslediti ovaj font. MathCalculator aplikacija na CD-u koristi Verdana font sa 10 tačaka.
2. Dodajte Label kontrolu, koja će postati displej kalkulatora. Podesite njeno BorderStyle svojstvo na Fixed 3D da ima 3-D izgled, kao što je prikazano na slici 2.6. Promenite i njena svojstva ForeColor i BackColor ako želite da izgleda različito od ostatka forme. Projekat koji ćete naći na CD-u koristi boje koje simuliraju sada već nepostojeći zeleni CRT monitor.
3. Postavite Button kontrolu na formi, promenite njen natpis (Text svojstvo) na 1 i nazovite je bttn1. Promenite pažljivo veličinu dugmeta da njegov natpis ostane centriran na kontroli. Druga dugmad na formi će biti kopije ovoga, tako da treba da proverite da li ste prvo dugme dizajnirali na najbolji mogući način, pre nego što počnete da pravite njegove kopije.
4. Kada postavite dugme na konačnu poziciju na formi, spremni ste da kreirate drugu dugmad za cifre kalkulatora. Kliknite desnim tasterom miša iznad dugmeta i izaberite Copy. Button kontrola je kopirana na Clipboard i sada je možete prebaciti na formu (što je mnogo brže od od dizajniranja identičnog dugmeta).
5. Kliknite desnim tasterom miša negde na formi i izaberite Paste da biste kreirali kopiju dugmeta koje ste ranije kopirali. Dugme koje ste kopirali na Clipboard će biti prebačeno na formu pomoću originalnog dugmeta. Kopija će imati isti natpis kao i dugme koje je prvobitno kopirano na Clipboard - njegovo ime će biti Button1.
6. Sada podesite Name svojstvo dugmeta na bttn2 i njegovo Text svojstvo na 2. Ovo dugme je cifra 2. Postavite novo dugme desno od prethodnog. Ne morate da ih poravnavate perfektno (kasnije ćemo koristiti Format meni za poravnavanje dugmadi na formi).
7. Ponovite korake 5 i 6 još osam puta, po jednom za svaku cifru. Svaki put nova Button kontrola se postavlja na formu, a Visual Basic je imenuje Button1 i podešava njen naslov na 1; Vi morate da promenite Name i Text svojstva. Možete da imenujete dugmad kako Vi želite; njihov Click događaj će biti obrađen istom podrutinom, koja će čitati Text svojstvo dugmeta da bi pronašla koja cifra je kliknuta.
8. Kada su sva dugmad cifara na formi, postavite još dva dugmeta: za C (Clear - brisanje) operaciju i za Period (tačka) dugme. Nazovite ih bttnClear i bttnPeriod i podesite njihove natpise na odgovarajući način. Koristite veću veličinu fonta za Period dugme da biste njegov natpis načinili lakšim za čitanje.

9. Kada su sva dugmad cifara prve grupe na formi i na svojim približnim pozicijama, poravnajte ih komandama Format menija.

   A. Prvo, poravnajte dugmad gornjeg reda. Počnite poravnavanjem dugmeta 1 sa levom stranom lblDisplay Label kontrole. Zatim, izaberite svu dugmad gornjeg reda i načinite njihovo horizontalno rastojanje jednakim (izaberite Format➥Horizontal Spacing➥Make Equal). Uradite isto sa dugmetima u prvoj koloni - proverite da li su njihove vertikalne razdaljine jednake (Format➥Vertical Spacing➥Make Equal).

   B. Sada možete odvojeno da poravnate dugmad u svakom redu i svakoj koloni. Koristite jedno od dugmadi koje ste poravnali u poslednjem koraku kao vođicu za preostalu dugmad. Dugmad mogu biti poravnata na mnoge načine, tako da ne treba da brinete ako negde u procesu narušite poravnanje. Uvak možete da koristite Undo komandu iz Edit menija. Izaberite tri dugmeta u drugom redu i, korišćenjem prvog kao reference, poravnajte njihove gornje ivice. Uradite isto za treći i četvrti red dugmadi. Učinite isto za četiri kolone dugmadi.

Sada postavite dugmad za aritmetičke operacije na formu: dodavanje (+), oduzimanje (-), množenje (*) i deljenje (/). Koristite komande na Format meniju da biste poravnali ovu dugmad, kao što je ranije prikazano na slici 2.6. Kontrola sa crnim kvadratićima može biti korišćena kao referenca za poravnavanje drugih kontrola u redove i kolone. Forma prikazana na slici 2.6 ima još nekoliko dugmadi, koje možete poravnati korišćenjem istih tehnika koje ste koristili za poravnavanje numeričkih dugmadi.

Equals dugme na dnu se naziva bttnEquals; morate ga načiniti dovoljno širokim da biste prekrili prostor tri dugmeta iznad njega.

Programiranje MathCalculator aplikacije

Sada ste spremni da dodate nešto koda aplikaciji. Duplo kliknite jedno od dugadi za cifre na formi i u prozoru koda videćete sledeće:

```
Private Sub bttn1_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttn1.Click
End Sub
```

Ovo je hendler Click događaja za dugme jedne cifre. Vaš prvi pokušaj je da programirate hendler Click događaja dugmeta svake cifre, ali ponavljanje istog koda 10 puta nije mnogo produktivno. Mi ćemo koristiti isti hendler događaja za svu dugmad koja predstavljaju cifre. Sve što je potrebno da uradite je da pridodate imena događaja koje obrađujete istom podrutinom posle Handles ključne reči. Treba, takođe, da promenite ime hendlera događaja na nešto što pokazuje na njegovu ulogu. Pošto ova podrutina rukuje Click događajem za svu dugmad koja predstavljaju cifre, nazovite je Digit_Click(). Ovo je deklaracija posle revizije podrutine koja može da rukuje sa dugmadima koja predstavljaju cifre:

```
Private Sub Digit_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttn1.Click,
bttn2.Click, _
```

```
                bttn3.Click, bttn4.Click, bttn5.Click, bttn6.Click, _
                bttn7.Click, bttn8.Click, bttn9.Click, bttn0.Click
End Sub
```

Kada pritisnete dugme za cifru na ručnom kalkulatoru, odgovarajuća cifra se pridoda na displeju. Da biste simulirali ovo ponašanje, ubacite sledeću liniju u hendler Click događaja:

```
lblDisplay.Text = lblDisplay.Text + sender.Text
```

Ova linija pridodaje displaju kalkulatora cifru koja je kliknuta. Argumenat sender Click događaja predstavlja kontrolu koja je kliknuta (kontrola koja je aktivirala događaj). Text svojstvo ove kontrole je cifra dugmeta koje je kliknuto. Na primer, ako ste već uneli vrednost 345, klikom na cifru 0 prikazuje se 3450 na Label kontroli koja radi kao displej kalkulatora.

Izraz `sender.Text` nije najbolji metod za pristupanje Text svojstvu dugmeta koje je kliknuto, ali će raditi sve dok je Strict opcija isključena. Vratićemo se na ovu temu kasnije u knjizi, a za sada se zadovoljite kratkim objašnjenjem da treba da konvertujete sender objekat u TextBox objekat, pa pristupite njegovom Text svojstvu sledećim iskazom:

```
CType(sender, TextBox).Text
```

CType() funkcija je obrađena u sledećem poglavlju. Za sada, imajte na umu da ona konvertuje jedan oblik u oblik drugog tipa. Takođe ćete primetiti da će se, posle ukucavanja tačke iza zatvorene zagrade, svi članovi TextBox kontrole pojaviti na listi, kao da ste uneli ime TextBox kontrole iza koje sledi tačka.

Kodu iza dugmadi cifara je potrebno još nekoliko linija. Posle određenih akcija displej treba da bude obrisan. Posle pritiskanja jednog od dugmadi koje odgovara matematičkim operacijama displej treba da bude obrisan u očekivanju drugog operanda. Zapravo, on mora biti obrisan odmah kada se pritisne prva cifra drugog operanda. Izvršite reviziju hendlera Digit_Click događaja, kao što je prikazano u listingu 2.3.

Listing 2.3: Digit_Click događaj

```
Private Sub Digit_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles bttn1.Click,
bttn2.Click, _
                bttn3.Click, bttn4.Click, bttn5.Click, bttn6.Click, _
                bttn7.Click, bttn8.Click, bttn9.Click, bttn0.Click
   If clearDisplay Then
      lblDisplay.Text = ""
      clearDisplay = False
   End If
   lblDisplay.Text = lblDisplay.Text + sender.text
End Sub
```

Promenljiva *clearDisplay* je deklarisana kao Boolean, što znači da može da ima True, ili False vrednost. Pretpostavimo da je korisnik izvršio jednu operaciju i da je rezultat na displeju kalkulatora. Korisnik sada počinje da ukucava drugi broj. Bez If iskaza, program bi nastavio da pridodaje cifre broju koji je već na displeju. Ovako kalkulator ne radi. Kada se unese novi broj, displej se mora obrisati. I naš program koristi *clearDisplay* promenljivu da bi znao kada da obriše displej.

Equals dugme podešava *clearDisplay* na True, pokazujući da displej sadrži rezultat operacije. DigitClick() podrutina pregleda vrednost ove promenljive svaki put kada se nova cifra pritisne. Ako je vrednost True, Digit_Click() briše displej, pa štampa novu cifru na njemu. Podrutina podešava *clearDisplay* na False - kada se pritisne sledeća cifra, program neće ponovo da obriše displej.

Šta učiniti ako korisnik pravi greške i želi da poništi unos? Tipičan ručni kalkulator nema backspace taster. Taster Clear briše trenutni broj na displeju. Implementirajte ovu osobinu. Duplo kliknite C dugme i unesite kod listinga 2.4 u njegov Click događaj.

Lisitng 2.4: Clear dugme

```
Private Sub bttnClear_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnClear.Click
   lblDisplay.Text = ""
End Sub
```

Sada možete da pogledate Period dugme. Kalkulator, bez obzira koliko je jednostavan, treba da je osposobljen da rukuje brojevima sa decimalnim zarezom. Period dugme radi kao i dugmad cifara, sa jednim izuzetkom. Cifra se može pojaviti bezbroj puta u numeričkoj vrednosti, a tačka samo jednom. Broj kao što je 99,991 je važeći, ali morate proveriti da korisnik ne unese brojeve kao što je 23.456.55. Kada je tačka jednom uneta, ovo dugme ne sme da ubaci još jednu. Kod u listingu 2.5 vodi računa o tome.

Listing 2.5: Period dugme

```
Private Sub bttnPeriod_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnPeriod.Click
   If lblDisplay.Text.IndexOf(".") > 0 Then
      Exit Sub
   Else
      lblDisplay.Text = lblDisplay.Text & "."
   End If
End Sub
```

IndexOf je metod koji se može primeniti na bilo koji string. Izraz `lblDisplay.Text` je string (tekst na Label kontroli), tako da možete da pozovete njegov IndexOf metod. Kod `IndexOf(".")` vraća lokaciju prvog primerka tačke u natpisu Label kontrole. Ako je ovaj broj pozitivan, uneti broj već sadrži tačku, a druga ne može biti uneta. U ovom slučaju program napušta podrutinu. Ako metod vrati 0, tačka se pridodaje broju koji je do sada unet, baš kao regularna cifra.

Proverite rad aplikacije. Već ste kreirali funkcionalni korisnički interfejs koji simulira ručni kalkulator sa mogućnostima za unos podataka. On još ne izvršava nikakve operacije, ali ste već kreirali funkcionalni korisnički interfejs sa malim brojem iskaza.

Matematičke operacije

Sada se možete prebaciti na interesantan deo aplikacije. Počnite definisanjem tri promenljive:

Operand1	Prvi broj u operaciji
Operator	Željena operacija
Operand2	Drugi broj u operaciji

Kada korisnik klikne matematičke simbole, vrednost sa displeja se čuva u promenljivoj *Operand1*. Ako korisnik onda klikne Plus dugme, program mora da zabeleži za sebe da je trenutna operacija sabiranje i da obriše displej. Tada korisnik može da unese drugu vrednost. Simbol operacije se čuva u *Operator* promenljivoj. Korisnik unosi drugu vrednost, pa, kliknuvši Equals dugme, može da vidi rezultat. U ovom trenutku naš program mora da uradi sledeće:

1. da pročita *Operand2* vrednost na displeju
2. da doda vrednost promenljivoj *Operand1*
3. da prikaže rezultat.

Equals dugme mora da izvrši sledeću operaciju:

```
Operand1 Operator Operand2
```

Pretpostavimo da je, kada korisnik klikne Plus dugme, broj na displeju 3342. Korisnik, zatim, unosi vrednost 23, pa klikne Equals dugme. Program mora da izvrši dodavanje:

```
3342 + 23
```

Da je korisnik kliknuo dugme Division (deljenje), operacija bi bila:

```
3342/23
```

U oba slučaja, kada je Equals kliknuto, rezultat je prikazan (i mora postati prvi operand za sledeću operaciju).

Promenljive su lokalne u podrutinama gde su deklarisane. Druge podrutine ne mogu im pristupiti i ne mogu da čitaju, ili podešavaju svoje vrednosti. Ponekad, međutim, promenljivim se mora pristupati sa mnogo mesta u programu. Ako se promenljivim Operand1, Opeand2 i Operator u ovoj aplikaciji mora pristupati iz više podrutina, moraju biti deklarisane van bilo koje podrutine. Isto važi i za clearDisplay promenljivu. Njihove deklaracije, stoga, moraju se pojaviti van bilo koje procedure i obično se pojavljuju na početku koda sa sledećim iskazima:

```
Dim clearDisplay As Boolean
Dim Operand1 As Double
Dim Operand2 As Double
Dim Operator As String
```

Pogledajmo kako program koristi *Operator* promenljivu. Kada korisnik klikne Plus dugme, program mora da sačuva "+" u *Operator* promenljivoj. Ovo se dešava u Click događaju Plus dugmeta. Ali, kasnije Equals dugme mora da ima pristup vrednosti *Operator* promenljive da bi moglo da izvrši operaciju (drugačije rečeno, mora da zna koji tip operacije je korisnik definisao).

Zbog toga što ove promenljive moraju da budu manipulisane iz više podrutina, one su deklarisane van bilo koje od njih.

Ključna reč Double je za Vas nova. Ona "kaže" VB-u da kreira numeričku promenljivu sa najvećom mogućom preciznošću za čuvanje vrednosti operatora (mumeričke promenljive i njihovi tipovi su detaljno obrađeni u sledećem poglavlju). Boolean tip uzima dve vrednosti: True i False. Već ste videli kako je korišćena *clearDisplay* promenljiva.

Promenljive *Operand1*, *Operand2* i *Operator* se nazivaju *Form-wide*, ili, jednostavno, *Form* promenljive, jer su vidljive iz svake podrutine na formi. Ako Vaša aplikcija ima drugu formu, ove promenljive ne bi bile vidljive iz te forme. Drugačije rečeno, svaka podrutina na formi na kojoj su promenljive deklarisane može da čita, ili podešava vrednosti promenljivih, ali to ne mogu da rade podrutine van te forme.

Sa završenim deklaracijama promenljivih, možete sada da implementirate Operator dugmad. Duplo kliknite Plus dugme i u hendleru Click događaja unesite linije prikazane u listingu 2.6.

Listing 2.6: Plus dugme

```
Private Sub bttnPlus_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnPlus.Click
   Operand1 = Val(lblDisplay.Text)
   Operator = "+"
   clearDisplay = True
End Sub
```

Promenljiva *Operand1* je pridružena vrednosti koja je trenutno na displeju. Funkcija Val() vraća numeričku vrednost svog argumenta. Text svojstvo Label kontrole je string. Na primer, možete pridružiti vrednost "My Label" Text svojstvu natpisa. Stvarna vrednost sačuvana u Text svojstvu nije broj. To je string kao što je "428", koji se razlikuje od numeričke vrednosti 428. Zbog toga, koristite Val() funkciju za konvertovanje vrednosti natpisa Label kontrole u numeričku vrednost. Preostala dugmad rade isto i ovde neću prikazati njihove listinge.

Do sada smo implementirali sledeće funkcionalnosti u našu aplikaciju. Kada se klikne dugme operatora, program čuva vrednost sa displeja u *Operand1* promenljivoj, a operator u *Operator* promenljivoj. Potom se briše displej tako da korisnik može da unese drugi operand. Pošto je drugi operand unet, korisnik može da klikne Equals dugme da bi proračunao rezultat. Kada se to desi, kod listinga 2.7 se izvršava.

Listing 2.7: Equals dugme

```
Private Sub bttnEquals_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnEquals.Click
   Dim result As Double
   Operand2 = Val(lblDisplay.Text)
   Select Case Operator
      Case "+"
         result = Operand1 + Operand2
      Case "-"
         result = Operand1 - Operand2
```

```
            Case "*"
               result = Operand1 * Operand2
            Case "/"
               If Operand2 <> "0" Then result = Operand1 / Operand2
         End Select
         lblDisplay.Text = result
         clearDisplay = True
      End Sub
```

Promenljiva *result* je deklarisana kao Double, tako da će rezultat operacije biti sačuvan sa maksimalnom preciznošću. Kod izvlači vrednost prikazanu na Label kontroli i čuva je u promenljivoj *Operand2*. On, zatim, izvršava operaciju sa `Select Case` iskazom, koji upoređuje vrednost Operator promenljive sa vrednostima iza svakog Case iskaza. Ako vrednost *Operator* promenljive odgovara jednoj od Case vrednosti, izvršava se sledeći iskaz.

- Ako je operator "+", promenljiva *result* se podešava na sumu od dva operanda.
- Ako je operator "-", promenljiva *result* se podešava na razliku prvi operand minus drugi.
- Ako je operator "*", promenljiva *result* se podešava na proizvod dva operanda.
- Ako je operator "/", promenljiva *result* se podešava na vrednost prvog operanda podeljenog sa drugim, gde delilac nije nula.

NAPOMENA

Deljenje uzima u obzir vrednost drugog operanda, jer, ako je ona nula, ono se ne može izvršiti. Poslednji If iskaz vrši deljenje samo ako delilac nije nula. Ako se desi da je Operand2 nula, ništa se ne dešava. ■

Sada pokrenite aplikaciju i proverite je. Ona radi baš kao i ručni kalkulator i ne možete je oboriti definisanjem nevažećih podataka. Mi nismo morali da koristimo bilo kakav kod za proveru vrednosti podataka u ovom primeru, jer korisnik ne dobija šansu da ukuca nevažeće podatke. Mehanizam za unos podataka je potpuno siguran. Korisnik može da unese samo numeričke vrednosti, jer postoje samo cifre na kalkulatoru. Jedina moguća greška je deljenje sa nulom, što je obrađeno u Equals dugmetu.

Alati za dibagovanje

Naša aplikacija radi lepo i vrlo je laka za testiranje i popravku (ako otkrijete nešto pogrešno u njoj). Ali, to je samo zbog toga što je vrlo jednostavna. Dok pišete kod, brzo ćete otkriti da nešto ne radi kao što je očekivano i treba umete da pronađete zašto i da to popravite. Proces eliminisanja grešaka se naziva *dibagovanje*. Visual Studio obezbeđuje alate za pojednostavljenje tog procesa, koji su obrađeni u Poglavlju 17. Postoji nekoliko jednostavnih operacija koje treba da znate, čak i ako radite sa jednostavnim projektima, kao što je ovaj.

Otvorite MathCalculator projekat u editoru koda i postavite kursor na liniju koja proračunava razliku između dva operanda. Možete da "odglumite" da postoji problem u ovoj liniji i da zbog toga želite da pratite izvršavanje programa iz blizine da biste pronašli grešku u aplikaciji.

Pritisnite F9 i linija će biti istaknuta braon bojom. Ona je postala *breakpoint* (tačka prekida): kada se do nje dođe, program se zaustavlja.

Pritisnite F5 da biste pokrenuli aplikaciju i izvršili oduzimanje. Unesite broj, pa kliknite minus dugme, onda unesite drugi broj i na kraju kliknite Equals dugme. Aplikacija će stati i otvoriće se editor koda. Tačka prekida će biti istaknuta žutom bojom. Postavite i zadržite (hover) pokazivač miša iznad promenljivih *Operand1* i *Operand2* u prozoru editora koda. Vrednost odgovarajućih promenljivih će se pojaviti u malom boksu. Pomerite pokazivač miša do bilo koje promenljive u aktuelnom hendleru događaja da biste videli njihove vrednosti. Ovo su vrednosti promenljivih upravo pre izvršenja istaknutog iskaza.

Promenljiva *result* će najverovatnije biti nula, jer iskaz još uvek nije izvršen. Ako promenljive uključene u ovaj iskaz imaju svoje pravilne vrednosti (ako nemaju, znate da je problem pre ovog iskaza i možda u drugom hendleru događaja), možete izvršiti ovaj iskaz pritiskom na F10. Pritiskanjem F10, izvršavate samo istaknuti iskaz. Program će stati na sledećoj liniji. Sledeći iskaz koji treba izvršiti je `End Select`.

Pronađite primerak promenljive *result* u aktuelnom hendleru događaja, zadržite miša iznad i videćete vrednost promenljive pošto joj je pridružena vrednost. Sada možete pritisnuti F10 da biste izvršili još jedan iskaz, ili F5 da biste se vratili u normalan mod za izvršavanje.
Možete, takođe, da proverite vrednost izraza, uključujući bilo koju od promenljivih u aktuelnom hendleru događaja, unošenjem odgovarajućeg iskaza u Command prozoru. Command prozor se pojavljuje u dnu IDE-a. Ako nije vidljiv, sa glavnog menija izaberite View➥Other Windows➥Command Window. Aktuelna linija u Output prozoru je prefiksovana sa znakom veće od (podseća na DOS dane). Postavite kursor pored njega i unesite sledeći iskaz:

```
? Operand1 / Operand2
```

Rezultat deljenja dve vrednosti će se pojaviti u sledećoj liniji. Znak pitanja je samo skraćena notacija za Print komandu. Ako želite da znate aktuelnu vrednost na displeju kalkulatora, unesite sledeći iskaz:

```
? lblDisplay.Text
```

Ovaj iskaz traži vrednost svojstva kontrole na formi. Aktuelna vrednost Text svojstva Label kontrole će se pojaviti na sledećoj liniji. Možete, takođe, izračunati vrednost matematičkog izraza iskazima, kao što je sledeći:

```
? Math.Log(3/4)
```

Log() je logaritamska funkcija i to je metod Math klase. Da biste kreirali slučajnu vrednost između 0 i 1, unesite iskaz:

```
? Rnd()
```

Vremenom ćete otkriti da je Command prozor vrlo pogodan alat pri dibagovanju aplikacija. Ako imate iskaz sa komplikovanim izrazom, možete tražiti vrednosti pojedinačnih komponenata izraza i stoga biti sigurni da on može biti proračunat.

Sada pomerite pokazivač sa tačke prekida i pritisnite F9 ponovo. Ovo će promeniti status tačke prekida i izvršavanje programa se neće zaustaviti kada se sledećeg puta ovaj iskaz izvršava.

Ako se izvršenje programa ne zaustavi u tački prekida, to znači da se nikada nije ni došlo do iskaza. U ovom slučaju morate potražiti grešku u iskazima koji su izvršeni pre tačke prekida. Ako niste pridružili pravilnu vrednost promenljivoj *Operator*, do `Case "-"` iskaza nikada neće da se dođe. Treba da postavite tačku prekida na prvi izvršni iskaz hendlera Click događaja Equalse dugmeta da biste pregledali vrednosti svih promenljivih u momentu kada počne izvršavanje ove podrutine. Ako sve promenljive imaju očekivane vrednosti, nastavićete da testirate kod. Ako nemaju, treba da testirate iskaze koji vode do ovog iskaza - iskaze u hendlerima događaja različitih dugmadi.

Druga jednostavna tehnika za dibagovanje aplikacija je Output prozor. Iako to nije alat za dibagovanje, vrlo je čest među VB programerima (i vrlo praktičan). Mnogi programeri štampaju vrednosti izabranih promenljivih posle izvršavanja nekih komplikovanih iskaza. Da biste to uradili, koristite iskaz:

```
Console.WriteLine
```

iza koga sledi ime promenljive koju želite da štampate, ili izraz:

```
Console.WriteLine(Operand1)
```

Ovaj iskaz šalje svoj izlaz u Output prozor, koji je prikazan pored Command prozora - kliknite Output jezičak na dnu IDE-a da biste videli ovaj prozor. Alternativa je da možete da izaberete komandu View➥Other Windows➥Output. Ovo je vrlo jednostavna tehnika, ali funkcioniše. Možete je koristiti i za testiranje funkcije, ili poziva metoda. Ako niste sigurni u vezi sa sintaksom funkcije, prosledite izraz koji sadrži specifičnu funkciju `Console.WriteLine` iskazu kao argumenat. Ako se očekivana vrednost pojavi u Output prozoru, možete da idete napred i koristite je u svom kodu.

Pogledajte DateDiff() funkciju, koja sadrži razliku između dva datuma. Najjednostavnija sintaksa ove funkcije je

```
DateDiff(interval, date1, date2)
```

Nikada ne znam da li ona oduzima date1 od date2, ili obratno - ako prvi put to tačno ne shvatite, onda će svaki put kada želite da koristite ovu funkciju postojati kod Vas nedoumica. Pre korišćenja ove funkcije u mom kodu, ubacujem iskaz, kao što je ovaj

```
Console.WriteLine(DateDiff(DateInterval.Day, #1/1/2000#, #1/2/2000#))
```

Vrednost odštampana na Output prozoru je 1, što pokazuje da je prvi datum oduzet od drugog.

Pronaći ćete više informacija o dibagovanju u Poglavlju 17. Ja sam Vam samo pokazao nekoliko jednostavnih tehnika koje će pomoći da iskoristite jednostavnije alate za dibagovanje Visual Studia kada budete pisali svoje prve aplikacije.

Dodavanje još nekoliko osobina

Sada, kada ste implementirali osnovne funkcionalnosti ručnog kalkulatora, možete dodati još osobina aplikaciji. Dodajte još dva korisna dugmeta:

- +/-, ili Negate dugme, koje menja znak broja na displeju
- 1/x, ili Inverse dugme, koje invertuje sam broj na displeju.

Otvorite prozor koda za svako od Command dugmadi i ubacite kod iz listinga 2.8 u odgovarajućim hendlerima Click događaja. Za +/- dugme unesite hendler događaja bttnNegate_Click, a za 1/x dugme unesite hendler bttnInverse_Click.

Listing 2.8: Negate i Inverse dugmad

```
Private Sub bttnNegate_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnNegate.Click
   lblDisplay.Text = -Val(lblDisplay.Text)
   clearDisplay = True
End Sub
Private Sub bttnInverse_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnInverse.Click
   If Val(lblDisplay.Text) <> 0 Then lblDisplay.Text = 1 /
Val(lblDisplay.Text)
   clearDisplay = True
End Sub
```

Kao i sa Division dugmetom, ne pokušavamo da invertujemo nultu vrednost. Operacija (1/0) je nedefinisana i prouzrokuje grešku pri izvršavanju. Primetićete da ja koristim vrednost prikazanu na Label kontroli direktno u kodu. Mogao sam da sačuvam `lblDisplay.Text` vrednost u promenljivoj i da, umesto nje, koristim promenljivu:

```
TempValue = Val(lblDisplay.Text)
If TempValue <> 0 Then lblDisplay.Text = 1 / TempValue
```

Ovo je, takođe, bolje kodiranje, ali u kratkim delovima koda, jer svi mi pokušavamo da minimiziramo broj iskaza. Možete lako da proširite Math aplikaciju dodavanjem Function dugmadi. Na primer, možete dodati dugmad za proračunavanje zajedničkih funkcija, kao što su Cos, Sin i Log. Cos dugme proračunava kosinus broja na displeju. Kod iza Click događaja dugmeta je jednolinijski:

```
lblDisplay.Text = Math.Cos(Val(lblDisplay.Text))
```

On ne zahteva drugi operand i vodi računa o operaciji. Možete implementirati sve matematičke funkcije sa jednom linijom koda.

Naravno, treba da dodate nešto koda za hvatanje grešaka i u nekim slučajevima možete koristiti tehnike za proveru podataka. Na primer, Sqrt() funkcija, koja proračunava kvadratni koren broja, očekuje pozitivan argumenat. Ako je broj na displeju negativan, možete da prikažete upozorenje:

```
If lblDisplay.Text < 0 Then
   MsgBox("Can't calculate the square root of a negative number")
Else
   lblDisplay.Text = Math.Sqrt(Val(lblDisplay.Text))
End If
```

Sve matematičke klase su deo Math klase; zbog toga su one prefiksovane imenom klase. Možete, takođe, uvesti Math klasu u projekat sa sledećim iskazom i tako izbeći prefiksovanje matematičkih funkcija:

```
Imports System.Math
```

Log() funkcija može da proračunava samo logaritme pozitivnih brojeva. Ako dodate dugme za proračunavanje logaritama i pokušate da proračunate logaritam negativnog broja, rezultat će biti string "NaN". Ova vrednost je slična sa beskonačnošću i "kaže" da rezultat nije važeći broj (NaN znači not a number - nije broj; detaljno se obrađuje u sledećem poglavlju). Naravno, prikazivanje vrednosti kao što je NaN na displeju kalkulatora nije najbolji metod za rukovanje matematičkim greškama. Ja bih proverio vrednost podataka i prikazao poruku sa odgovarajućim opisom, kao što je prikazano u listingu 2.9.

Listing 2.9: Proračunavanje logaritma broja

```
Private Sub bttnLog_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnLog.Click
   If Val(lblDisplay.Text) < 0 Then
      MsgBox("Can't calculate the logarithm of a negative number")
   Else
      lblDisplay.Text = Math.Log(lblDisplay.Text)
   End If
   clearDisplay = True
End Sub
```

Još jedna osobina koju dodajete kalkulatoru je mogućnost da ograničite broj cifara na displeju. Većina kalkulatora može samo da prikaže ograničeni broj cifara. Radi dodavanja ove osobine Math aplikaciji (ako ovo smatrate "osobinom"), koristite Len() funkciju da biste pronašli broj cifara na displeju i ignorisali bilo koju cifru unetu pošto je broj dostigao maksimalan broj dozvoljenih cifara.

Rukovanje izuzecima

Obaranje ove aplikacije neće biti lako kao obaranje Loan aplikacije. Ako počnete da množite vrlo velike brojeve, nećete dobiti grešku u prekoračenju. Unesite vrlo veliki broj, tako što ćete više puta ukucavati cifru 9, pa pomnožite ovu vrednost sa drugom jednako velikom vrednošću. Kada se rezultat pojavi, kliknite simbol za množenje i unesite još jednu vrlo veliku vrednost. Množite rezultat sa vrlo velikim brojevima, dok ne iscrpite opseg vrednosti Double tipa podataka (to jest, dok rezultat ne postane toliko veliki da ne može da se smesti u promenljivu Double tipa). Kada se ovo desi, pojaviće se na displeju string "infinity" (beskonačnost). Naš kod ne uključuje iskaze za hvatanje prekoračenja vrednosti, pa odakle onda dolazi string "infinity"? Kao što ćete učiti u sledećem poglavlju, moguće je da numeričke proračune vrate string "infinity". To je način Visual Basica da Vam "kaže" da ne može da rukuje vrlo velikim brojevima. Ovo nije ograničenje VB-a; to je način na koji računari skladište numeričke vrednosti: oni obezbeđuju ograničeni broj bajtova za to. Naći ćete više detalja o neobičnostima kao što je infinity u sledećem poglavlju.

Takođe ne možete da napravite izuzetak prekoračenja deljenjem broja sa nulom, jer kod neće ni pokušati da izvrši ovu operaciju. Ukratko, Calculator aplikacija je prilično robustna. Međutim, ne možete biti sigurni da korisnici neće prouzrokovati da aplikacija generiše izuzetak, tako da morate da obezbedite neki kod za rukovanje svim tipovima grešaka. Greške se sada zovu izuzeci. Možete ih zamisliti kao odstupanja od normalnog (ili nameravanog) toka izvršenja. Ako se izuzetak pojavi, program mora da izvrši specijalni iskaz za rukovanje izuzetkom - iskaz koji normalno ne bio izvršen. Mislim da se zovu izuzetak, jer reč "greška" niko od nas ne voli i većina ljudi ne može da prizna da

je napisala kod koji sadrži grešku. Izraz exception (izuzetak) može da bude neodređen. Šta biste radije rekli svojim kupcima: da aplikacija koju ste napisali ima greške, ili da je Vaš kod prijavio izuzetke? Možda to niste primetili, ali reč bug (još jedan izraz za grešku) više se ne koristi često. Međutim, izraz *dibagovanje* još nije promenjen.

VB6 programeri su koristili izraz *error* da opišu nešto pogrešno u svom kodu i imali su običaj da pišu kod za hvatanje grešaka. Sa VB.NET-om, vaš kod je bez grešaka - on tu i tamo podiže izuzetke. I kod za hvatanje grešaka u VB6 i osobine VB.NET-a za rukovanje izuzecima su podržani. Kod za hvatanje grešaka u VB6 je mogao da postane zbunjujući, tako da je Microsoft dodao ono što oni nazivaju *strukturirano rukovanje izuzecima*. To je organizovaniji metod za rukovanje greškama pri izvršavanju (izuzecima). Osnovna pretpostavka je da se program, kada se izuzetak pojavi, ne završava sa porukom o grešci. Umesto toga, on izvršava deo koda koji programer obezbedi.

SAVET

Uzgred budi rečeno, ako Vam je teško da priznate grešku u svom kodu, koristite iskaz "mea culpa". To je latinski izraz i on zvuči veoma sofisticirano, a većina ljudi Vas neće ni pitati šta on znači. ■

Kako da sprečite izuzetak koji podiže kalkulacija? Provera podataka neće pomoći. Vi, jednostavno, ne možete da predvidite rezultat operacije, a da nju stvarno ne izvršite. I ako operacija izazove prekoračenje vrednosti, to ne možete sprečiti. Odgovor je dodavanje strukturiranog hendlera izuzetaka. Veći deo koda aplikacije je jednostavan i ne možete da generišete izuzetak. Jedino mesto gde se izuzetak može pojaviti je hendler Equals dugmeta, gde se proračun dešava. Ovo je mesto gde morate dodati hendler izuzetaka. Nacrt strukture greške je sledeći:

```
Try
   { statements block }
Catch Exception
   { handler block }
Finally
   { clean-up statements block }
End Try
```

Program će pokušati da izvrši proračune koji su kodirani u `statements blocku`. Ako uspe u tome, nastavlja sa `clean-up-statements`. Ovi iskazi su, uglavnom, kod za čišćenje, a Finally deo iskaza je opcioni. Ako nedostaje, izvršavanje programa se nastavlja sa iskazom koji sledi iza `End Try` iskaza. Ako se pojavi greška u prvom bloku iskaza, `Catch Exception` deo se aktivira i iskazi u `handler` bloku se izvršavaju.

Catch blok je mesto gde rukujete greškom. Ne možete mnogo učiniti sa greškama koje su rezultat kalkulacija. Sve što možete da uradite je da prikažete upozorenje i date korisniku šansu da promeni vrednosti. Postoje drugi tipovi grešaka, koji mogu biti obrađeni mnogo skladnije. Ako Vaš program ne može da pročita fajl sa CD-a, možete korisniku dati šansu da ubaci CD u drajv. U drugim situacijama možete zatražiti od korisnika nedostajuću vrednost i nastaviti. Uopšteno govoreći, ne postoji jedinstven način za rukovanje svim izuzecima. Morate da uzmete u obzir sve tipove izuzetaka koje Vaša aplikacija može da prouzrokuje i da rukujete sa njima pojedinačno.

Hendler grešaka Math aplikacije mora da informiše korisnika da se pojavila greška i da prekine operaciju - bez pokušaja da prikaže rezultat. Ako otvorite hendler Click događaja Equals dugmeta, naći ćete iskaze navedene u listingu 2.10.

Listing 2.10: Equals dugme posle revizije

```
Private Sub bttnEquals_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnEquals.Click
   Dim result As Double
   Operand2 = Val(lblDisplay.Text)
   Try
      Select Case Operator
         Case "+"
            result = Operand1 + Operand2
         Case "-"
            result = Operand1 - Operand2
         Case "*"
            result = Operand1 * Operand2
         Case "/"
            If Operand2 <> "0" Then result = Operand1 / Operand2
        End Select
        lblDisplay.Text = result
   Catch exc As Exception
      MsgBox(exc.Message)
      lblDisplay.Text= "ERROR"
   Finally
      clearDisplay = True
   End Try
End Sub
```

U dužem periodu hendler grešaka stoji neaktivan i ne učestvuje u radu programa. Ako se pojavi greška, koja će najverovatnije biti greška u prekoračenju vrednosti, biće izvršen deo za rukovanje greškama `Try...Catch...End Try` iskaza. Ovaj kod prikazuje boks sa porukom sa opisom greške i prikazuje string "ERROR" na displeju. Deo Finally se izvršava bez obzira da li se greška pojavila, ili nije. U ovom primeru Finally deo podešava *clearDisplay* promenljivu na True - kada se klikne dugme druge cifre, pojavljuje se novi broj na displeju.

NAPOMENA

Promenljiva exc predstavlja izuzetak; ona izlaže nekoliko svojstava pored Message svojstva, koje je opis izuzetka. Za više informacija o članovima Exception klase i za objašnjenje kako da rukujete izuzecima pogledajte Poglavlje 17. ■

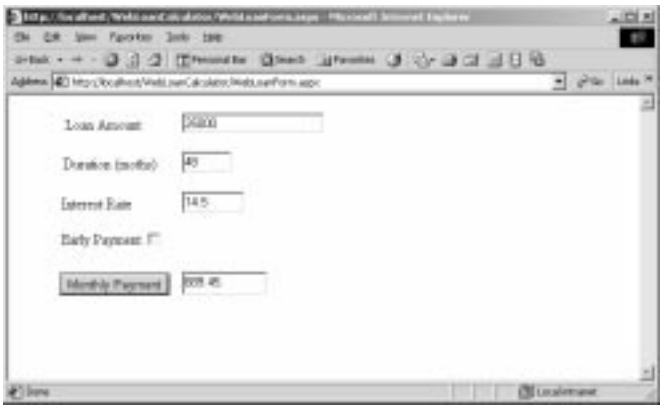
Postavljanje LoanCalculatora na Web

U ovom odeljku izgradićemo novi projekat za proračunavanje zajma sličan onom koji smo gradili ranije. Ovoga puta aplikacija će se pokretati na browseru i bilo koji korisnik koji se može povezati sa Vašim serverom moći će da ga koristi, a da ne mora da ga instalira na svom računaru. Kao što možete razumeti, treba da konvertujete LoanCalculator od Windows aplikacije u web aplikaciju. Malo je prerano da bismo obrađivali web aplikacije, ali sam želeo da pokažem da je izgradnja tih aplikacija sasvim slična sa izgradnjom Windows aplikacija.

Web aplikacije su obrađene detaljno u poslednjem delu knjige, ali, pošto su one među najvažnijim novim osobinama .NET platforme, demonstriraću zašto su veoma značajne. VisualStudio.NET je prvi pokušaj da se razvoj web aplikacija načini lakim, kao što je razvoj VB aplikacija. Uskoro ćete videti da možete da kreirate interfejs web forme (HTML stranu sa kontrolama koje stupaju u interakciju sa korisnikom) baš kao i kada kreirate Windows formu. Što se tiče koda aplikacije, on je isti kao pisanje VB koda za rukovanje događajima Windows forme.

Da biste pisali i testirali web aplikacije, morate imati instaliran i pokrenut Internet Information Server (IIS) na svom računaru. IIS je distribuiran sa Windowsom 2000 i morate proveriti da li on radi. Otvorite Start meni i izaberite Settings➥Control Panel. Duplo kliknite Administrative Tools, pa duplo kliknite ikonu Internet Services Manager alata. Kada se pojavi Internet Services Manager prozor, proširite čvor svog računara, kliknite desnim tasterom miša Default Web Site stavku i iz iskačućeg menija izaberite Start. Ovo će pokrenuti web server.

Započnite novi projekat i u NewProject okviru za dijalog kliknite ASP.NET Web Aplication ikonu. Zatim, ubacite ime aplikacije u Name boks - nazovite je WebLoanCalculator. Kada zatvorite New Project okvir za dijalog, videćete prozor sa mrežom, kao i obično, koji predstavlja web stranu, ili web formu. Ovaj dokumenat se naziva `WebForm1.aspx` (podrazumevano ime web forme). Web forma je ekvivalentna sa Windows formom, ali je prikazana kao HTML u browseru, kao što je Internet Explorer (pogledajte sliku 2.7).



SLIKA 2.7 *WebLoanCalculator Web aplikacija*

Novi Windows projekat je sačuvan u sopsvenoj fascikli ispod fascikle definisane u Location polju u New Project okviru za dijalogu. Web aplikacije su, takođe, sačuvane u sopstvenoj fascikli, koja je kreirana ispod korene fascikle web servera (obično `C:\Inetpub\wwwroot` fascikle).

Otvaranje web projekta nije samo dupli klik na ikonu Solution fajla. Savetujem da pratite korake za kreiranje projekta opisane u ovom poglavlju. Ako želite da otvorite WebLoanCalculator projekat na CD-u, kopirajte čitavu WebLoanCalculator fasciklu u korenu fasciklu WebServera. Zatim, pokrenite Visual Studio.NET i otvorite WebLoanCalculator fajl rešenja. Tekst objašnjava kako od početka treba kreirati projekat. Glavna forma aplikacije se naziva `WebLoanForm.aspx` (ekvivalentna sa Windows formom). Možete otvoriti aplikaciju pokretanjem Internet Explorera i unošenjem sledeće URL adrese u njegov Address boks:

```
http://localhost/WebLoanCalculator/WebLoanForm.aspx
```

Dozvolite da opišem proces izgradnje web aplikacije od samog početka. Promenite ime WebForm1 u WebLoanForm. Otvorite Toolbox i videćete da je Web Forms jezičak aktiviran, umesto Windows Forms jezička. Web Forms jezičak sadrži ikone kontrola koje možete da postavite na web formu, koje su slične sa Windows kontrolama, ali nisu toliko kompleksne, ili bogate funkcionalnostima. Kao što već znate, web strane koriste mnogo jednostavniji model za interakciju sa korisnicima. Korisnik može da unese tekst na određene kontrole, potvrdi, ili poništi nekoliko opcija i klikne dugme za prenos forme na server. Server čita vrednosti sa kontrola, obrađuje ih i vraća novu stranu sa rezultatima. Možete očekivati da će aplikacije koje se izvršavaju preko Interneta postajati sve kompleksnije, ali za sada se ne dovodi u pitanje HTML model koji je i do sada korišćen. Sve dok browser može da rukuje samo HTML fajlovima izgled web aplikacija je ograničen HTML stranama.

Postoji i drugi jezičak na Toolboxu - HTML tab. To su standardne HTML kontrole koje možete koristiti na bilo kojoj web strani. Web Forms jezičak sadrži takozvane web kontrole; postoji samo nekoliko web kontrola, nasuprot ograničenom broju HTML kontrola. Neke od web kontrola su sasvim napredne u poređenju sa vrlo ograničenim mogućnostima HTML kontrola. Da li ovo znači da strana koja sadrži web kontrole ne može da bude prikazana na browseru koji nije Internet Explorer? Ne, uopšte ne znači. Web kontrole se automatski prevode u standardni HTML kod koji se može prikazati u bilo kom browseru. Na primer, na Web Forms jezičku naći ćete neke vrlo kompleksne kontrole, kao što je TreeView. HTML ne obezbeđuje nikakve kontrole koje se čak i ne približavaju po funkcionalnosti TreeView kontroli. Pored toga, Web TreeView kontrola se može prikazati na bilo kom browseru. Web Forms Designer će uneti odgovarajuće HTML tagove da bi kreirao nešto što izgleda i ponaša se kao TreeView kontrola. Postoji puno štošta što treba reći o web kontrolama, ali ćete morati da sačekate do poslednjeg dela knjige. Za sada, gradićemo jednostavnu aplikaciju koja koristi web kontrole da zatraži od korisnika parametre zajma i koja će prikazati mesečnu ratu na istoj strani, baš kao i Windows aplikacija.

Počnite sa postavljanjem četiri Label kontrole na web formu (duplo kliknite ikonu Label kontrole na Toolboxu četiri puta i četiri nalepnice će biti postavljene na web formu za Vas). Promenite njihova mesta na formi i uredite ih pomoću miša, baš kao što biste uradili i sa kontrolama na Windows formi. Ne morate da ih sada perfektno poravnavate; koristićete komande Format menija za poravnavanje kontrola na formi. Samo ih grubo postavite na pozicijama prikazanim na slici 2.7. Zatim, izaberite svaku Label kontrolu pomoću miša i u Properties prozoru pronađite Text svojstvo kontrole.

Kao što možete videti, većina osnovnih svojstava web kontrola ima ista imena kao i Windows kontrole. Promenite natpise na četiri Label kontrole na "Loan Amount" - iznos zajma, "Duration (months)" - trajanje u mesecima, "Interest" - kamata i "Monthly Payment" - mesečna rata. Primetićete da se Label Web kontroli dimenzije menjaju automatski da bi prihvatila string koji treba da pridružite njenom Text svojstvu.

Sada postavite četiri TextBox kontrole na web formu, svaku pored jedne Label kontrole. Podrazumevano stanje je da su sve TextBox kontrole prazne (nemaju inicijalni sadržaj). Promenite njihovu veličinu mišem i poravnajte ih sa odgovarajućim Label kontrolama. Onda ih birajte jednu po jednu i promenite njihovo ID svojstvo na txtAmount, txtDuration, txtRate i txtPayment, respektivno. ID svojstvo web kontrole je jedinstveni identifikator kontrole, slično sa Name svojstvom Windows kontrole. Koristićete ID svojstvo za pristupanje članovima kontrole iz Vašeg koda.

Zatim, postavite CheckBox kontrolu, podesite njeno Text svojstvo na Early Payment i nazovite je chkPayEarly. Podesite njeno TextAlign svojstvo na Left, tako da će polje za potvrdu biti postavljeno desno od teksta. Polje za potvrdu će biti postavljeno odmah posle teksta, tako da treba da pridodate nekoliko praznina natpisu kontrole da ga jasno razdvojite od polja za potvrdu.

Poslednja kontrola koju treba postaviti na formu je Button kontrola, čije Text svojstvo će biti "Monthly Payment", a Name svojstvo će biti bttnShowPayment. Ovo dugme će proslediti parametre zajma unete na formu serveru, gde će odgovarajući kod proračunati mesečnu ratu i vratiti je klijentu. Ovo je pravi momenat da poravnate kontrole na web formi. Izaberite Label kontrole i poravnajte ih ulevo pomoću Format➥Align➥Lefts komande. Dok su Label kontrole izabrane, koristite Format➥Vertical Spacing➥Make Equal da biste ih ravnomerno rasporedili jednu od druge. Kada su Label kontrole postavljene, možete da poravnate svaku TextBox kontrolu odgovarajućom Label kontrolom, pomoću Format➥Align➥Middles komande. Izaberite jedan po jedan par Label i TextBox kontrole i poravnajte ih. Samo proverite da se Label kontrola koristi kao referentna kontrola za poravnavanje.

U ovom momentu završili ste dizajaniranje interfejsa aplikacije. Interfejs je sasvim sličan sa interfejsom ekvivalentne Windows aplikacije, samo što je ova dizajnirana na web formi sa web kontrolama. Ostali deo procesa je bio isti; čak su i alati za poravnavanje kontrola na web formi isti kao i oni za Windows formu. Vaš sledeći zadatak je da programirate aplikaciju.

Duplo kliknite dugme na web formi i prozor editora će se otvoriti. Web Form Designer je izabrao Click događaj dugmeta i uneo njegovu definiciju. Sve što treba da uradite je da ubacite isti kod koji smo koristili u LoanCalculator aplikaciji. Možete da pređete na Windows aplikaciju i kopirate kod (koji je ranije prikazan u listingu 2.2). Samo prekopirajte kod iza Show Payment dugmeta LoanCalculator Windows aplikacije u hendler Click događaja Monthly Payment dugmeta web aplikacije i neće biti ni jedne greške. Možete koristiti kod onakav kakav je.

Pritisnite F5 da biste pokrenuli aplikaciju. Proći će nekoliko sekundi pre nego što se pojavi prozor Internet Explorera, prikazujući stranu koju ste dizajnirali. Unesite parametre zajma, pa kliknite Monthly Payment dugme. Nekoliko sekundi kasnije mesečna rata će se pojaviti na formi. Kao što ćete primetiti, na browser će stići potpuno nova strana; ona sadrži parametre zajma (vrednosti koje ste uneli na formu) i rezultat proračuna.

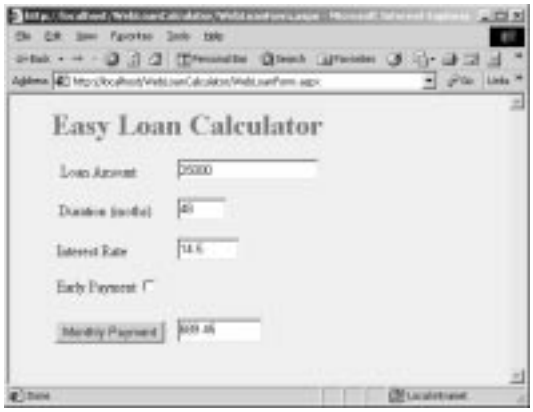
Ako pogledate izvorni kod dokumenta prikazan na Internet Exploreru, videćete čist HTML kod. Interfejs WebLoanCalculator aplikacije izgleda lepo, ali ne sasvim kao web strana. Nema nikakvih boja, ili grafike na koje smo navikli na web stranama. Naša web forma sadrži samo kontrole, ali je to HTML strana i možete dodati bilo koji elemenat koji se može pojaviti na web strani. Drugačije rečeno, web forma može biti editovana kao HTML dokumenat. I ne samo to: IDE omogućava da editujete aavašu stranu ili vizuelno, ili u HTML modu. Dodajte obojene natpise i promenite boju pozadine strane.

Izaberite web formu klikom negde na formi. U Properties prozoru izaberite svojstvo pageLayout. Njegovo podešavanje je `GridLayout`, što objašnjava zašto ste mogli da postavljate kontrole bilo gde na strani i poravnavate ih na sve moguće načine. Oni kojima je HTML poznat znaju da je poravnavanje kontrola na web formi jednostavno. Promenite pageLayout svojstvo od `GridLayout` na `FlowLayout`. Sada ste u normalnom HTML modu za editovanje. Postavite kursor na vrhu strane i počnite kucanje. Unesite string Easy Loan Calculator, pa ga izaberite mišem. Primetićete da su dugmad za formatiranje teksta na panou sa alatima omogućena. Podesite veličinu teksta na 6 i podesite njihove boje ispisa i pozadine. Da biste podesili ova svojstva, koristite dva dugmeta pored Bold/Italic/Underline grupe dugmadi. String će biti istaknut levo na formi, tako da treba da dodate nekoliko praznina ispred stringa da biste ga centrirali iznad kontrola.

NAPOMENA

Kratak komentar za čitaoce koji znaju HTML: browser ignoriše više razmaka, ali editor ih bez obaveštavanja o tome konvertuje u   kodove, koji su HTML ekvivalent za "hard" - to jest, neprelomljene praznine. ■

Možete, takođe, promeniti boje strane. Pronađite bgColor svojstvo strane u Properties prozoru i podesite ga na svetlu boju. Kada se Color Picker okvir za dijalog pojavi na formi, videćete jezičak sa web bojama. One mogu da budu prikazane na svim browserima (takozvane sigurne boje). Forma sada izgleda kao slika 2.8 kada se pogleda u browseru.



SLIKA 2.8 *WebLoanCalculator kao web strana*

Da biste videli kako Web Form Designer rukuje HTML elementima strane, kliknite HTML dugme na dnu Designera. Web forma može da bude pregledana i dizajnirana u Design pregledu (koji je podrazumevan), ili u HTML pregledu. Web Form Designer je ubacio sledeći iskaz u HTML dokumenat za generisanje zaglavlja strane:

```
<FONT style="BACKGROUND-COLOR: #ffff66" color="#996666" size="6">
<STRONG>Easy Loan Calculator</STRONG></FONT>
```

Ovo je čist HTML kod koji se može pojaviti u bilo kojoj web strani i ne koristi ni jednu web kontrolu. Izaberite <FONT> tag i obrišite ga. Zatim, pređite na Design pregled da biste videli da je zaglavlje nestalo. Pređite nazad na HTML pregled i ubacite sledeći iskaz odmah posle <body> taga, a pre <form> taga, kao što je prikazano na slici 2.9:

```
<h1>Easy Loan Calculator</h1>
```

SLIKA 2.9 *Editovanje HTML koda web forme*

Kliknite F5 da biste pokrenuli aplikaciju. Kada se Internet Explorer pojavi, unesite neke vrednosti u okvire za tekst i proverite aplikaciju. Web aplikacija je funkcionalno ekvivalentna sa Windows Loan aplikacijom, koju ste razvili na početku ovog poglavlja. Jedino što se njen korisnički interfejs izvršava u browseru, ali se proračuni dešavaju na serveru (mašina na kojoj se klijenti povezuju da zahtevaju `WebLoanForm.aspx` Web stranu). Svaki put kada kliknete Monthly Payment dugme na strani, strana se šalje na server. Browser prebacuje vrednosti različitih kontrola nazad na server. Server obrađuje ove vrednosti (zapravo, on izvršava hendler događaja koji ste napisali) i kreira novu stranu, koja je poslata klijentu. Ova strana uključuje vrednost mesečne rate. Web aplikacije se obrađuju detaljno kasnije u knjizi; ovim primerom želeo sam da demonstriram sličnosti između Windows formi i web formi i kako isti kod radi sa oba tipa aplikacije.

Rad sa više formi

Vratimo se Windows aplikacijama. Nekoliko aplikacija su izgrađene na istoj formi. Većina koristi dve, tri, ili više formi, koje odgovaraju odvojenim delovima aplikacije. U ovom odeljku gradićemo jednu aplikaciju koja koristi tri forme i omogućava korisniku da se prebacuje između njih po želji. Videćete kako da pišete aplikaciju koja otvara više prozora na Desktopu. U Poglavlju 4 detaljno ćemo istražiti izgradnju Windows aplikacija sa više formi. U ovom poglavlju gradićemo jednostavan primer aplikacije sa više formi kombinovanjem matematičkih i finansijskih kalkulatora koje smo izgradili ranije u poglavlju. Način da kombinujete dve aplikacije je da kreirate novu formu, koja će postati mesto za prebacivanje između dva kalkulatora. Korisnik će moći da aktivira jedan od dva kalkulatora klikom dugmeta na novoj formi. Dizajnirajte jednu aplikaciju koja kombinuje forme dva projekta.

Započnite novi projekat i nazovite ga Calculators. Forma projekta će postati mesto za prebacivanje između druge dve forme, što je prikazano na slici 2.10. Započnite promenu imena nove forme od Form1 na CalculatorsForm. Da biste to dizajnirali, dodajte dve Button kontrole i imenujte ih bttnMath i bttnLoan. Zatim, podesite njihova Text svojstva na Simple Math i Simple Loan, respektivno. Kao što možete pretpostaviti, sve što sada treba da uradite je da dodate kod za aktiviranje svake od postojećih formi iz hendlera Click događaja svakog dugmeta. Dodajte treće dugme na formi (nazovite ga bttnGame) i kasnije možete dodati akcionu igru Calculators projektu.

SLIKA 2.10 *Glavna forma Calculators aplikacije*

U ovom momentu morate dodati forme MathCalculator i LoanCalculator projekata u novi projekat. Kliknite desnim taster om miša iznad imena projekta i iz iskačućeg menija izaberite Add Existing Item. U okvir za dijalogu koji se pojavi izaberite stavku `MathForm.vb` iz fascikle MathCalculator projekta. Uradite isto za LoanForm LoanCalculator projekta. Calculator projekat sada sadrži tri forme. Ako sada pokrenete projekat, videćete Calculators formu, ali klikom na njeno dugme nećete dobiti odgovarajuću formu. Očigledno je da morate dodati nekoliko linija koda u hendler Click događaja svakog dugmeta da biste aktivirali odgovarajuću formu. Da biste prikazali jednu formu iz koda druge forme, morate kreirati objekat koji predstavlja drugu formu, a, zatim, poziva njen Show metod. Kod iza Simple Math dugmeta je prikazan u listingu 2.11.

Listing 2.11: Aktiviranje Math kalkulatora

```
Private Sub bttnMath_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnMath.Click
   Dim calcForm As New CalculatorForm
   calcForm.Show()
End Sub
```

Promenljiva calcForm je objektna promenljiva koja predstavlja CalculatorForm formu Calculators aplikacije. Ime forme se, zapravo, koristi kao tip podataka, što zahteva neka objašnjenja. Forma je implementirana kao Class, pa, stoga, kreirate objekte ovoga tipa.

Dim iskaz kreira novi primerak forme, a Show metod učitava i prikazuje formu. Ako sada pokrenete projekat, videćete glavnu formu, a, ako kliknete prvo dugme, forma matematičkog kalkulatora će se pojaviti. Ako ponovo kliknete isto dugme, pojaviće se još jedan primerak forme. Šta možete da uradite da biste ovo sprečili? Vi biste želeli da prikažete CalculatorForm inicijalno i onda jednostavno pokažete, ali ne i da učitate još jedan primerak forme. Odgovor je da prebacite deklaraciju *calcForm* promenljive van hendlera događaja u deo za deklaracije forme. Promenljiva je deklarisana jednom i sve procedure u formi mogu da pristupaju njenim članovima. Promenljive deklarisane u hendleru događaja imaju efekta samo u hendleru događaja u kome su deklarisane i zbog toga, svaki put kada kliknete dugme, kreira se i prikazuje novi primerak odgovarajuće forme. Ako promenljiva *calcForm* pokazuje na jedan primerak CalculatorForm forme, forma će biti prikazana svaki put kada kliknete Simple Math dugme, ali neće biti kreirani novi primerci. Više informacija o dosegu promenljivih naći ćete u sledećem poglavlju.

Kada je jedan od dva kalkulatora prikazan, on ne postaje automatski aktivna forma. Aktivna forma je ona koja ima fokus- to je glavna forma aplikacije. Da biste radili sa kalkulatorom, morate da kliknete odgovarajuću formu, čime ćete je načiniti aktivnom. Da biste aktivirali poslednju prikazanu formu iz koda druge forme, koristićete Activate metod Form objekta. Ponovo napišite hendlere Click događaja dva dugmeta na formi, kao što je prikazano u listingu 2.12 (listing prikazuje čitav kod forme, tako da možete da vidite deklaracije dve promenljive koje predstavljaju forme aplikacije).

Listing 2.12: Calculators projekat

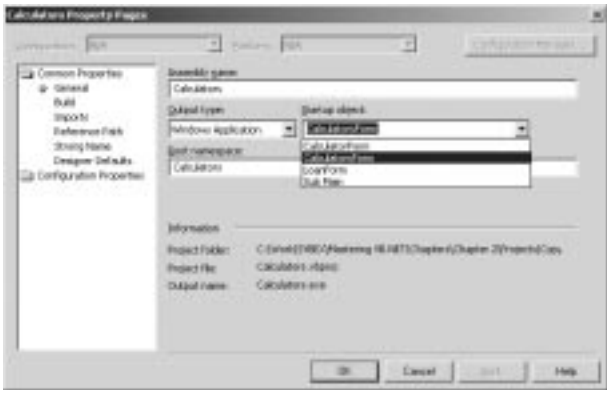
```
Public Class CalculatorsForm
   Inherits System.Windows.Forms.Form
   Dim calcForm As New CalculatorForm()
   Dim loanForm As New loanForm()
   Private Sub bttnMath_Click(ByVal sender As System.Object, _
                  ByVal e As System.EventArgs) Handles bttnMath.Click
      calcForm.Show()
      calcForm.Activate()
   End Sub
   Private Sub bttnLoan_Click(ByVal sender As System.Object, _
                  ByVal e As System.EventArgs) Handles bttnLoan.Click
      loanForm.Show()
      loanForm.Activate()
   End Sub
End Class
```

Primetićete iskaz koji deklariše *loanForm* promenljivu: promenljiva ima isto ime kao i tip podataka, ali to nije problem. Podrazumeva se da ime promenljive može da bude bilo šta. Vaš sledeći zadatak je da definišete koja će forma biti prikazana kada pokrenete aplikaciju. Kliknite desnim tastreom miša ime Calculators projekta i na iskačućem meniju izaberite Properties. Na Calculators Property Pages okviru za dijalog (slika 2.11) je ComboBox nazvan StartUp Object. Proširite ga i videćete imena svih

formi u projektu. Izaberite ime forme koju želite da se pojavi kada se program pokrene, što je CalculatorsForm.

Kod iza Play A Game dugmeta treba, takođe, da pozove Show metod druge forme, ali on to ne radi. Žao mi je što nisam izabrao igru za Vašu zabavu, ali sam implementirao smešnu osobinu. Kada kliknete ovo dugme, ono skoči na drugo mesto na formi. Hendler Click dugmeta je prikazan sledeći:

```
Private Sub bttnGame_Click(ByVal sender As System.Object, _
              ByVal e As System.EventArgs) Handles bttnGame.Click
   bttnGame.Left = Rnd() * Me.Width * 0.8
   bttnGame.Top = Rnd() * Me.Height * 0.8
End Sub
```



SLIKA 2.11 *Otvorite Project Properties okvir za dijalog da biste definisali početni objekat.*

Ova podrutina manipuliše Left i Top svojstvima kontrole da bi pomerila dugme na različitu poziciju. Rnd() funkcija vraća slučajnu vrednost između 0 i 1. Da biste proračunavali horizontalnu poziciju, kod množi slučajnu vrednost sa širinom forme (zapravo, sa 80 procenata širine). Vertikalna pozicija se računa na sličan način.

Svaki Visual Basic projekat čine fajlovi koji su izlistani u Solution Explorer prozoru. Svaki projekat sadrži samo nekoliko fajlova pored fajlova formi i svi su sačuvani u jednoj fascikli, koja je nazvana po projektu. Ako otvorite `Calculators` fasciklu (slika 2.12), videćete da ona sadrži CalculatorForm i LoanForm forme. Ovo su kopije originalnih formi odgovarajućih aplikacija. Kada dodate postojeću stavku projektu, VB pravi kopiju ove stavke u fascikli projekta.

SLIKA 2.12 *Komponente Calculators projekta*

Da biste premestili projekat na drugu lokaciju, morate tamo prebaciti fasciklu projekta. Da biste kreirali kopiju projekta, samo kopirajte fasciklu projekta na različitu lokaciju.

Rad sa više projekata

Kao što ste primetili, svaki novi projekat koji kreirate sa VB-om naziva se *solution* (rešenje). Svako rešenje sadrži projekat, koji, dalje, sadrži jedan, ili više fajlova, reference na .NET, ili korisničke komponente i druge tipove stavki, koje će biti obrađene u sledećim poglavljima. I rešenja i projekti su kontejneri - oni sadrže druge stavke. Rešenje može da sadrži više projekata. Svaki projekat u rešenju je nezavisan od drugih projekata i možete odvojeno distribuirati projekte u rešenju. Zašto kreirati rešenje? Recimo da radite na nekoliko povezanih projekata, koji će verovatno da koriste zajedničke komponente. Umesto da kreirate različito rešenje za svaki projekat, možete kreirati jedno rešenje koje sadrži sve povezane projekte.

Izgradite rešenje sa dva povezana projekta. Dva povezana projekta su dva kalkulatora koja ste izgradili ranije u ovom poglavlju. Oni ne dele nikakve zajedničke komponente, ali su dovoljno dobri za demonstraciju i videćete kako VB rukuje komponentama rešenja.

VB.NET u radu: Calculators rešenje

Kreirajte Empty Project (prazan projekat) i nazovite ga Calculators izborom File➥New➥Blank Solution. U Solution Explorer prozoru videćete ime projekta i ništa više, čak ni listu referenci koje su prisutne u bilo kom drugom tipu projekta. Da biste dodali projekat rešenju, izaberite File➥Add Project➥Existing Project (možete, takođe, da desnim tasterom miša kliknete ime rešenja u Solution Exploreru, izaberete Add Existing Item➥Project i u okviru za dijalog koji se pojavi da izaberete Calculator projekat). Uradite isto za LoanCalculator projekat. Kada se pojavi Add Existing Project okvir za dijalog, dođite do fascikle sa odgovarajućim projektima i izaberite fajl projekta. Sada imate rešenje, nazvano Calculators, koje sadrži dva projekta. Ako pokušate da pokrenete projekat, IDE ne zna koji od dva projekta da izvrši i generisaće poruku o grešci. Moratte odlučiti kako da počnete novi projekat (to jest, koju formu da prikažete kada korisnik pokrene Calculators aplikaciju). Kada rešenje sadrži više od jednog projekta, morate definisati startni projekat. Kliknite desnim tasterm miša jedan od projekata i sa iskačućeg menija izaberite Set As StartUp Project. Da biste testirali različiti projekat, podesite različit StartUp projekat. Normalno,

radićete neko vreme sa istim projektom, tako da prebacivanje sa jednog projekata na drugi nije problem. Moguće je, takođe, da različiti programeri rade na različitim projektima koji pripadaju istom rešenju.

Recimo da želite da dizajnirate fajl dokumentacije za oba projekta. Dobar izbor za kratak fajl dokumentacije je HTML fajl. Da biste dodali HTML fajl rešenju, kliknite desnim tasterom miša ime rešenja i izaberite Add New Item. U okviru za dijalog izaberite HTML Page šablon, pa unesite ime za novu stavku. HTML strana će biti dodata projektu, a prazna strana će se pojaviti u Designeru. Ovo je novododata HTML strana i morate joj dodati nešto sadržine.

Postavite kursor na površinu za dizajniranje i započnite kucanje. Na slici 2.13 prikazana je vrlo jednostavna HTML strana sa predstavljanjem aplikacije. Da biste formatirali tekst, koristite dugmad na panelu sa alatima. Ova dugmad umeću odgovarajuće tagove u tekst, dok Vi vidite stranu kao što bi izgledala u browseru. Ovo je Design pregled dokumenta. Možete da se prebacite na HTML pregled i editujete dokumenat ručno ako vam je HTML poznat. HTML stranu mogu da koriste oba projekta; i, na kraju, možete je distribuirati pomoću aplikacije.

SLIKA 2.13 *Dodavanje HTML dokumenta u rešenje*

Ako otvorite fasciklu kreiranu za projekat, konstatovaćete da ona sadrži neobično mali broj fajlova. Projekti se nalaze u svojim posebnim fasciklama. Napravite promenu na jednom od fajlova projekta. Možete promeniti boju pozadine tri TextBox kontrole na LoanForm na svetlu nijansu, kao što je Bisque. Zatim, otvorite LoanCalculator projekat i videćete da promena ima efekta. VB ne kreira nove kopije formi (ili bilo koje druge komponente) koje su dodate Calculators rešenju. Koristi postojeće fajlove i modifikuje ih, ako je potrebno, na njihovim originalnim lokacijama. Naravno, možete da kreirate rešenje od početka i da postavite sve stavke u istu fasciklu. Svaka fascikla je odvojeni entitet i možete da kreirate izvršne fajlove za svaki projekat i distribuirate ih.

Da biste kreirali izvršne fajlove, otvorite Build meni i izaberite Build Solution, ili Rebuild Solution. Build Solution komanda kompajlira fajlove koji su bili promenjeni od poslednjeg kompajliranja; Rebuild Solution kompajlira sve fajlove u projektu. Izvršni fajlovi će biti kreirani u `Bin` fascikli ispod fascikle svakog projekta. Fajl `Loan.exe` će biti kreiran ispod `\Loan\Bin` fascikle, a `Calculator.exe` fajl ispod `\Calculator\Bin` fascikle. Rešenje je pogodnost za programera. Kada radite na velikom projektu koji uključuje nekoliko povezanih aplikacija, možete ih sve staviti u rešenje i raditi projekat po projekat. Drugi programeri mogu da rade sa drugim projektima koji pripadaju istom rešenju. Dizajner može da kreira grafiku za aplikacije, a Vi ih možete uključiti u rešenje i one će biti dostupne svim projektima koji pripadaju istom rešenju.

Calculators projekat koji ste gradili ranije sadrži kopije formi koje ste dodali projektu. Calculators rešenje sadrži reference na spoljne projekte.

Izvršni fajlovi

Do sada ste izvršavali aplikacije u okviru Visual Basic okruženja. Međutim, ne možete očekivati da svi korisnici Vaše aplikacije imaju Visual Studio instaliran na svojim sistemima. Ako razvijete interesantnu aplikaciju, nećete želeti da date izvorni kod aplikacije. Aplikacije se distribuiraju kao izvršni fajlovi, zajedno sa fajlovima podrške. Korisnici aplikacije ne mogu da vide Vaš izvorni kod, a Vaša aplikacija ne može da bude modifikovana, ili promenjena da izgleda kao aplikacija neke druge osobe (to, naravno, ne znači da ne može da bude kopirana).

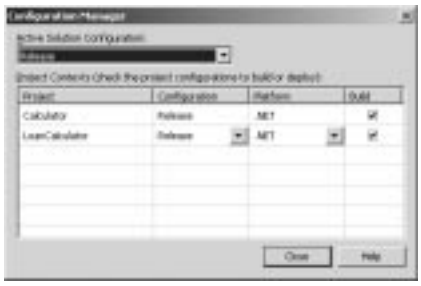
NAPOMENA

Izvršni fajl je binarni fajl sa instrukcijama koje samo mašina može da razume i izvrši. Komande uskladištene u izvršnom fajlu se nazivaju mašinski jezik. ■

Aplikacije dizajnirane za Windows okruženje ne mogu da stanu u jedan fajl. To ne bi imalo smisla. Zajedno sa izvršnim fajlovima, Vaša aplikacija zahteva fajlove podrške, koji mogu već da postoje na mnogim mašinama na kojima će Vaša aplikacija biti instalirana. Zbog toga, nema smisla distribuirati velike fajlove. Svaki korisnik treba da instalira glavnu aplikaciju i samo one fajlove podrške koji nisu već instalirani na računarima.

Izvršni fajl će se izvršavati na sistemu na kome je razvijen, jer su fajlovi za podršku tamo. Ispod fajla projekta naći ćete dve fascikle, nazvane `Bin` i `Obj`. Otvorite `Obj` fasciklu i videćete da ona sadrži potfasciklu `Debug`. Ovo je mesto gde ćete naći izvršni fajl, koji je nazvan po projektu i ima ekstenziju `.exe`. Proverite da li se izvršava neki primerak VS-a na Vašem računaru, pa duplo kliknite ikonu `MathCalculator.exe`, ili `LoanCalculator.exe` fajla. Odgovarajuća aplikacija će se pokrenuti van Visual Studio IDE-a i možete je koristiti kao i bilo koju drugu aplikaciju na Vašem PC-u. Možete kreirati prečice sa desktopa prema dvema aplikacijama.

Fascikla `Debug` sadrži Debug verziju izvršnog fajla. Normalno je, pošto ste završili dibagovanje aplikacije, da treba da promenite podrazumevanu konfiguraciju projekta od Debug na Release. Da biste promenili konfiguraciju projekta, izaberite Build➥Configuration Manager. Configuration Manager okvir za dijalog će se pojaviti, kao što je prikazano na slici 2.14.



SLIKA 2.14 *Configuration Manager prozor*

Podrazumevana konfiguracija za sve projekte je Debug. Ona generiše kod optimizovan za dibagovanje. Drugo moguće podešavanje za konfiguraciju je Release. Promenite konfiguraciju na Release i zatvorite okvir za dijalog. Ako ponovo gradite projekat, ili rešenje, Release fascikla će biti kreirana ispod Obj fascikle i sadržaće novi izvršni fajl. Razlika između ove dve verzije izvršnih fajlova je u tome što Debug fajlovi sadrže simbolične informacije za dibagovanje. Release konfiguracija se izvršava brže, jer ne sadrži nikakve informacije za dibagovanje.

Distribuiranje aplikacije

Distribuiranje samo EXE fajla neće biti dovoljno, jer izvršni fajlovi zahtevaju fajlove podrške. Ako ovi fajlovi nisu instalirani na ciljnom sistemu (računar na kome će Vaša aplikacija biti instalirana), EXE fajl neće raditi. Fajl će biti izvršen samo na sistemu koji na sebi ima Visual Studio.NET. Distrubuiranje velikog broja fajlova i njihovo instaliranje na ciljnom računaru je pravi zadatak. Morate kreirati program za instalaciju koji (skoro) automatski instalira Vašu aplikaciju i tražene fajlove podrške na ciljnom računaru. Ako su neki od ovih fajlova već instalirani, oni neće biti instalirani ponovo.

NAPOMENA

Na kraju će svi fajlovi za podršku postati deo operativnog sistema i onda ćete moći da distribuirate jedan EXE fajl (ili mali broj fajlova). Ovo se nije desilo sa Windowsom 2000, ili Windows XP-om i neće još neko vreme. Dok se to ne desi, morate obezbediti sopstveni instaler. ■

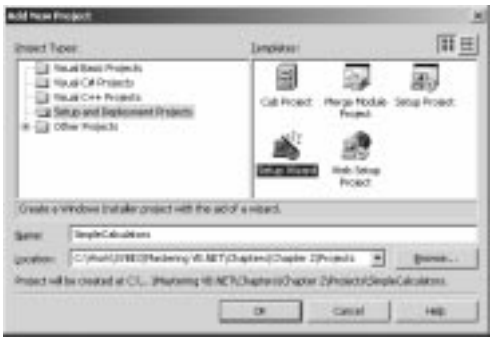
Setup projekat kreira Windows fajl za instalaciju (fajl sa ekstenzijom `.msi`), koji sadrži izvršne fajlove aplikacije i pomoćne fajlove potrebne aplikaciji, ulaze Registrya (ako aplikacija stupa u interakciju sa Registryem), uputstva za instalaciju i tako dalje. Rezultujući MSI fajl je, obično, veoma dugačak; njega distribuirate krajnjim korisnicima. Oni moraju da duplo kliknu ikonu MSI fajla da bi instalirali aplikaciju na svom računaru. Ako pokrenu isti fajl ponovo, aplikacija će biti uklonjena. Osim toga, ako tokom instalacije "krene nešto naopako", instalacija će biti vraćena unazad i sve komponente koje su instalirane u procesu će biti uklonjene.

Tema kreiranja i prilagođavanja Windows instalera je ogromna i već postoji nekoliko knjiga samo o toj temi - na primer, *VB/VBA Developer's Guide to the Windows Installer*, čiji je autor Mike Gunderloy (Sybex, 2000). U ovom poglavlju ćemo samo započeti obradu ove teme. Ja ću Vam pokazati kako da kreirate jednostavan Setup projekat za instaliranje Calculators projekta na drugoj mašini. Vaš glavni prioritet je da sada naučite kako da pišete .NET aplikacije i da ovladate jezikom. Treba da budete sposobni da distribuirate čak i male aplikacije, tako da tema kreiranja Setup projekata ne treba da izostane iz ove knjige. Još uvek nećete da koristite naprednije osobine - ne pre nego što budete mogli da pišete kompleksne aplikacije koje zahtevaju prilagođene procedure instalacije. U ovom odeljku pokazaću kako da kreirate jednostavan Setup projekat za Calculators projekat, koji demonstrira osnovne korake kreiranja Windows instalera korišćenjem podrazumevanih opcija, i moći ćete da instalirate Calculators aplikaciju na ciljni računar.

VB.NET u radu: Kreiranje Windows instalera

Da biste kreirali Windows instaler, morate dodati Setup projekat svom rešenju. Setup projekat će kreirati instalacioni program za projekte u aktuelnom rešenju. Otvorite Calculators rešenje i dodajte novi projekat (File➥Add Project➥New Project). U okviru za dijalog koji će se pojaviti (slika 2.15) kliknite stavku Setup i Deployment projekte. Najjednostavniji tip Setup projekta je Setup Wizard. Ovaj čarobnjak vodi Vas vodi kroz korake kreiranja Setup projekta, drugog čarobnjaka koji vodi korisnika kroz korake instalacije aplikacije na ciljnom računaru. Izaberite ovaj šablon, pa unesite ime projekta u Name boksu: nazovite projekat SimpleCalculators. Kliknite OK i prvi ekran čarobnjaka će se pojaviti. Ovo je ekran dobrodošlice i možete kliknuti Next dugme da biste ga zaobišli.

Na sledećem ekranu biće zatraženo da izaberete tip projekta. Možete kreirati projekat koji instalira aplikaciju, ili projekat koji dodaje komponente postojećoj instalaciji. Ako želite da kreirate projekat koji instalira aplikaciju prvi put, imate dve opcije: da kreirate setup za Windows aplikaciju, ili za web aplikaciju. Izaberite prvu opciju, kao što je prikazano na slici 2.16, i kliknite Next da biste prešli na sledeći ekran čarobnjaka.



SLIKA 2.15 *Dodavanje Setup i Deployment projekta u rešenje*

SLIKA 2.16 *Project Type ekran čarobnjaka*

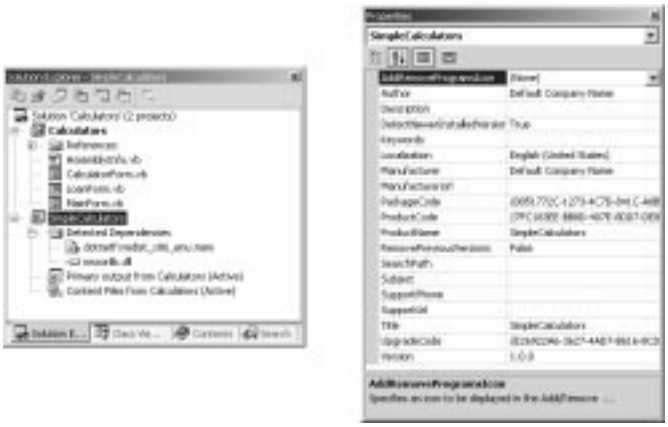
Na sledećem ekranu biće zatraženo da izaberete fajlove koje želite da dodate u instalacioni program. Ovde morate da kliknete stavke potvrđene na slici 2.17. Primary Output je izvršni fajl, a Content Files uključuje "stvari" kao što su HTML fajl, koji smo dodali projektu. U završnoj - release verziji programa obično ne želite da uključite simbole za dibagovanje, ili izvorne fajlove (dpbrp, možda simbole za dibagovanje velikih projekata koji su takođe kreirani na strani klijenta). Ako Vaša aplikacija uključuje lokalizovane fajlove resursa, treba da potvrdite drugu opciju. Lokalizovani resursi omogućavaju da pišete aplikacije koje podešavaju svoj tekst prema kulturi - jeziku krajnjeg korisnika (ovo je specijalna tema, koja nije obrađena u ovoj knjizi).

Setup projekat koji ovde kreirate je deo rešenja sa projektom koji želite da instalirate na ciljnoj mašini. Ja sam uključio Setup projekat u istom rešenju samo zbog udobnosti. Možete, takođe, da kreirate Setup projekat i definišete bilo koji izvršni fajl koji želite da instalirate. Setup projektu je potrebno malo vremena za kompajliranje, tako da ga treba dodati rešenju tek pošto ste dibagovali aplikaciju, ili ukloniti Setup projekat iz rešenja pošto ste kreirali setup fajl.

SLIKA 2.17 *Definisanje stavki koje želite da instalirate*

Kliknite Next ponovo da biste videli još jedan ekran, gde možete da definišete dodatne fajlove koji nisu deo projekta. Možete dodati tekstualne fajlove sa instalacionim instrukcijama, podatke o kompatibilnosti, informacije o registraciji i tako dalje. Kliknite Next ponovo i poslednji ekran čarobnjaka prikazuje kratak pregled projekta koji ste definisali. Kliknite Finish da biste zatvorili čarobnjaka i kreirali Setup projekat. Čarobnjak dodaje Setup projekat Vašem rešenju. Izaberite novi projekat mišem i otvorite Properties prozor da biste videli svojstva novog projekta. Solution Explorer i Properties prozor novog projekta treba da izgledaju kao oni koji su prikazani na slici 2.18. Dobra vest je da ne morate da pišete nikakav kod za ovaj projekat. Sve što treba da uradite je da podesite nekoliko svojstava i završili ste.

Svojstvo AddRemoveProgramsIcon omogućava da definišete ikonu za instalaciju i uklanjanje programa, a VB će takođe kreirati program za deinstalaciju aplikacije. Možete da definišete da li će Setup projekat detektovati novije verzije aplikacije i neće ih zameniti sa starijom verzijom. DetectNewerInstaledVersion svojstvo je podrazumevano True. Možete, takođe, da definišete ime Vašeg preduzeća i URL, liniju za podršku, naslov prozora za instalaciju i tako dalje.



SLIKA 2.18 *Properties projekta Setup*

Manufacturer svojstvo će postati ime fascikle u kojoj će se dešavati instalacija. Podrazumevano stanje je da se ova fascikla kreira u korisnikovoj fascikli Program Files. Pridružite ime koje reflektuje ili vaše preduzeće, ili tip projekta - string kao što je "The Math Experts" za primer Calculators. Author svojstvo je mesto gde treba da se pojavi Vaše ime. ProductName svojstvo je podrazumevano ime Setup projekta; promenite ga u "The EasyCalc Projekat". Title svojstvo je naslov instalera (ono što korisnici vide na panou za naslov čarobnjaka za instalaciju dok se aplikacija instalira).

Solution Explorer dugmad

Primetićete da su uobičajena dugmad na Solution Exploreru zamenjena sa šest novih dugmeta, koji su opisani u sledećim odeljcima.

Dugme File System Editor

Kliknite ovo dugme i videćete četiri dela fajl sistema ciljne mašine na koje može da utiče Vaš setup program. Odlučite da li akcija Vaše aplikacije treba da se pojavi na Desktopu korisnika, ili u Programs meniju. Desnim tasterom miša kliknite bilo koju stavku i videćete iskačući meni koji sadrži komande Add i Create Shortcut. Add komanda vodi prema podmeniju sa četiri objekta koje možete automatski da kreirate iz Vašeg Setup programa: Folder, Project Output, File i Assembly. Za tipične aplikacije možete dodati fasciklu (u koju kasnije možete da čuvate izlaz koji proizvodi projekat), ili izlaz projekta. Manje nametljiva opcija je postavljanje skraćenice u Programs meniju korisnika. Da biste projekat načinili zanimljivijim, instalirajte ne samo Calculators aplikaciju, već dve pojedinačne aplikacije: Calculator i LoanCalculator projekte. Dodaćete tri nove komande Programs meniju korisnika, pa dodajte fasciklu ovom meniju, a, zatim, u ovu fasciklu imena aplikacija. Kliknite desnim tasterom miša stavku User's Programs Menu i izaberite Add Folder. Nova fascikla će biti dodata ispod User's Programs Menu stavke. Promenite njeno ime na Demo `Calculators`, kao što je prikazano na slici 2.19. Izaberite novu fasciklu i pretražite njena svojstva. AlwaysCreate svojstvo treba da je True - ako nije, čarobnjak neće dodati fasciklu Programs meniju korisnika.

Zatim, kliknite desnim tasterom miša novododatu fasciklu i izaberite Add➥File. Pojaviće se okvir za dijalog gde možete da izaberete izvršne fajlove koji se moraju pojaviti u `Demo Calculators` fascikli na Programs meniju. Pretražite disk i locirajte Calculators, Calculator i LoanCalculator izvršne fajlove u `\Obj\Release` fascikli ispod fascikle odgovarajućeg projekta (sva tri fajla imaju ekstenziju EXE).

Nakon dodavanja stavki koje želite da se pojave u `Demo Calculators` fascikli Programs menija, File System Editor treba da izgleda kao onaj na slici 2.19.

SLIKA 2.19 *Definisanje kako će instalacioni program da utiče na korisnikov File System*

Dugme Registry Editor

Kliknite ovo dugme da biste dodali nove ključeve Registryu korisnika. Ne morate da dodajete bilo šta u Registry korisnika, naročito za ovaj projekat. Ali možete da postavite specijalne stringove u Registry, kao što je kodiran datum, da biste pronašli kada ističe demo verzija Vaše aplikacije. Morate prvo upoznati Registry i naučiti kako da ga programirate sa Visual Basicom, pre nego što pokušate da ga koristite sa Vašim aplikacijama.

Dugme File Types Editor

Ako Vaša aplikacija koristi sopstveni tip fajla, možete udružiti taj tip sa njom, tako da se, kada korisnik duplo klikne fajl ovog tipa, ona izvršava automatski. Ovo je siguran način da uništite udruživanja korisnikovih fajlova. Ako Vaša aplikacija može da rukuje GIF slikama, ili HTML fajlovima, nemojte ni razmišljati da preuzmete ove fajlove. Koristite ovu opciju samo sa fajlovima koji su jedinstveni u Vašoj aplikaciji. Da biste dodali novi tip fajla na mašini korisnika, kliknite File Types Editor dugme na Properties prozoru. Na površini Designera videćete jednu stavku: File Types On Target Machine. Kliknite je desnim tasterom i izaberite Add File Type. Ova komanda će dodati novi tip fajla i reč **&Open** ispod njega. Kliknite novi tip fajla i videćete njegova svojstva u Properties prozoru. Možete pridružiti opis novom tipu fajla, njegovu ekstenziju i komandu koja će biti korišćena za otvaranje fajlova ovoga tipa (ime EXE fajla Vaše aplikacije).

Dugme User Interface Editor

Kliknite ovo dugme i videćete korake za instalaciju na površini Designera, kao što je prikazano na slici 2.20. Svaka faza instalacionog procesa ima jedan, ili više koraka - pri svakom je prikazan različit okvir za dijalog. Neki od okvira za dijalog sadrže poruke, kao što su kratak opis aplikacije, ili poruka o pravima kopiranja. Ovi stringovi su izloženi kao svojstva odgovarajućeg okvira za dijalog i možete ih promeniti. Samo kliknite okvir za dijalog u User Interface Editoru i pregledajte njegova svojstva u Properties prozoru.

SLIKA 2.20 *Pregled instalacionog procesa*

Čarobnjak ubacuje sve potrebne okvire za dijalog, ali možete dodati i svoje. Ako to uradite, morate obezbediti i nešto koda za obradu izbora korisnika na korisničkom okviru za dijalog. Za naš jednostavan primer, nije potreban nikakav korisnički okvir za dijalog. Ovde ću ponoviti da je tema kreiranja prilagođenih Windows instalera jedan od glavnih aspekata Visual Studio.NET-a - kada se spremate da gradite instaler za veliku aplikaciju, moraćete dosta da konsultujete dokumentaciju.

Dugmad Custom Actions i System Requierements

Poslednja dva dugmeta na Properties prozoru omogućavaju da definišete korisničke akcije i zahteve na ciljnoj mašini. Na primer, možete da definišete da aplikacija bude instalirana samo na sistemima na kojima je određena komponenta već instalirana. Možete da ignorišete ovu dugmad kod jednostavnih projekata instalacije.

Završavanje Windows instalera

Skoro smo na kraju. Izaberite Build➥Build Solution i VB će kreirati instalacioni program. Prvo, on će kreirati novu fasciklu projekta - `SimpleCalculators`. Ovo je mesto gde će biti skladišteni fajlovi Setup projekta i gde će biti kreiran izvršni fajl instalacionog programa. Proces građenja izvršnih fajlova i kreiranja Setup programa će trajati nekoliko minuta. Izlaz procesa izgradnje je `SimpleCalculators.msi` fajl. Ovo je jedan izvršni fajl (poznat kao Windows Instaler Package); biće kreiran u `\SimpleCalculators\Release` fascikli. Njegova veličina će otprilike iznositi 15 MB. Ako se pitate šta je u ovom fajlu, pogledajte u Output prozor IDE-a i videćete dugačku listu komponenata dodatih paketu.

Izvršavanje Windows instalera

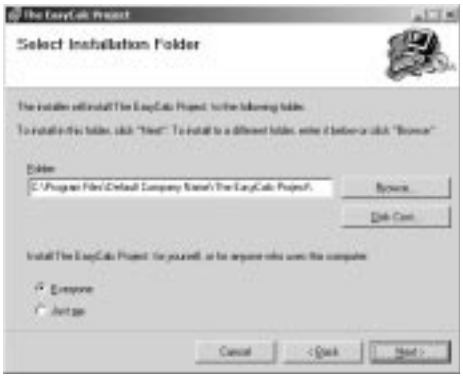
Sada ste spremni da instalirate Calculators projekat na svoj računar. Ako imate pristup drugom računaru koji nema instaliran Visual Studio, treba da kopirate `SimpleCalculators.msi` fajl tamo i instalirate ga. Komponente koje zahteva Vaša aplikacija da bi pravilno radila već su instalirane na mašini za razvoj i bolje možete testirati Setup projekat na drugoj mašini. Dođite do fascikle `\SimpleCalculators\Release` i duplo kliknite ikonu Windows Installer Package (ili fasciklu u koju ste kopirali ovaj fajl na drugoj mašini). MSI fajl je predstavljen tipičnom ikonom za instalaciju (računar i CD). Na sledećim slikama prikazani su koraci pri instalaciji. Uočite gde se na ekranu čarobnjaka za instalaciju pojavljuju naslovi koje ste definisali u svojstvima Setup projekta. Konsultujte

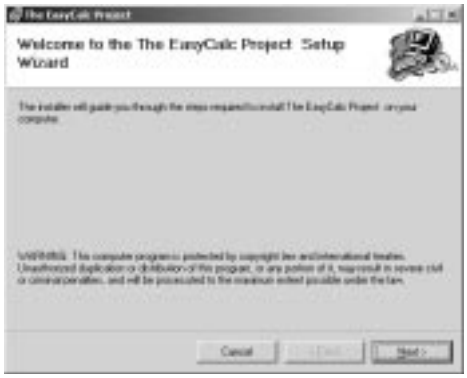
ove slike kada gradite Setup aplikaciju da biste bili sigurni da se prikazuju prave poruke za vreme instalacije na ciljnoj mašini.

1. Ovaj dijalog se pojavljuje dok se Windows instaler pokreće.

2. Ekran dobrodošlice čarobnjaka će voditi korisnika kroz proceduru instalacije. Poruke na ovom ekranu su svojstva CopyrightWarning i WelcomeText okvira za dijalog Welcome u User Interface Editoru.

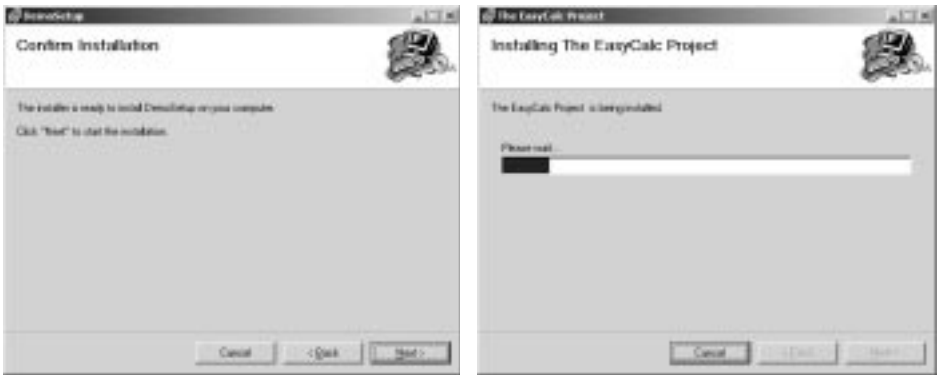
3. Ovaj ekran omogućava korisniku da promeni podrazumevanu putanju aplikacije koju treba instalirati. Primetićete kako je formirana podrazumevana putanja. Možete da kontrolišete podrazumevanu putanju za instalaciju podešavanjem odgovarajućeg svojstva Setup projekta. Instaler će kreirati fasciklu ispod `Program Files` fascikle, nazvanu po Manufacturer i Product Name svojstvima projekta Setup.



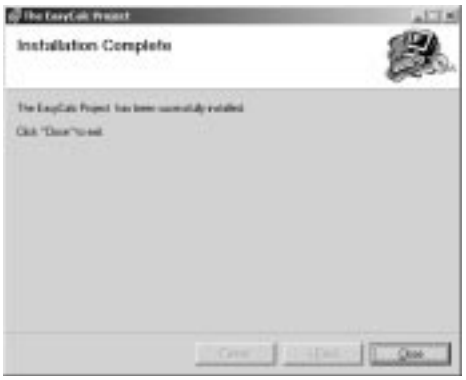


4. Ovaj ekran pita korisnika da potvrdi instalaciju - što se kasnije takođe može otkazati.

5. Aplikacija je instalirana i ovaj ekran prikazuje indikator napredovanja. Korisnik može da prekine instalaciju klikom na Cancel dugme.



6. Poslednji ekran instalera potvrđuje uspešnu instalaciju aplikacije. Kliknite Close da biste završili program. Ako je postojao program pri instalaciji aplikacije, biće prikazan opis problema na ovom poslednjem ekranu. U ovom slučaju sve komponente instalirane u procesu biće, takođe, automatski isključene.



Verifikacija instalacije

Vi već znate tipove promena koje je izvršio instalacioni program. Ako otvorite Programs meni (Start➥Programs), videćete da je dodata nova stavka - Demo Calculators. Ako je izaberete mišem, otvoriće se podmeni, kao što je prikazano na slici 2.21. Možete da izaberete bilo koju od tri komande (Calculators, Calculator, ili LoanCalculator) da biste pokrenuli odgovarajuću aplikaciju.

SAVET

Sve tri stavke u Demo Calculators podmeniju imaju podrazumevanu ikonu aplikacije. Treba da promenite podrazumevane ikone Vaših aplikacija da bi dobile profesionalniji izgled. ■

SLIKA 2.21 *Nove stavke koje je Programs meniju dodao Windows instaler*

Windows instaler je kreirao i instalirao program za deinstaliranje aplikacije sa ciljnog računara. Otvorite Control panel i duplo kliknite Add/Remove Programs ikonu. Okvir za dijalog koji se pojavljuje sadrži stavku za svaki program koji možete da uklonite sa Vašeg računara. Novoinstalirana aplikacija je stavka The EasyCalc Project, kao što je prikazano na slici 2.22. Kliknite njeno Remove dugme da biste deinstalirali aplikaciju, ili Change dugme da biste popravili postojeću instalaciju.

SLIKA 2.22 *Koristite Add/Remove Programs pomoćni program za uklanjanje, ili popravku aplikacije instalirane Windows instalerom.*

Što se tiče lokacije izvršnih fajlova i njihovih fajlova za podršku, oni su u `EasyCalc Project` fascikli ispod `\ProgramFiles\CompanyName` fascikle. Ako isti klijent instalira još neku Vašu aplikaciju - na primer, ProCalc Project, ona će takođe biti instalirana u sopstvenoj fascikli ispod `\ProgramFiles\CompanyName`. Samo treba da proverite da li svi Setup projekti imaju istu vrednost za Manufacturer svojstvo i fajlovi za podršku neće biti instalirani u više fascikli.

Zaključak

U ovom poglavlju su Vam predstavljeni koncepti *rešenja* i *projekti*. Naučili ste kako da izgradite jednostavno rešenje sa jednim projektom, kao i rešenje sa više projekata. Koristite rešenja da biste kombinovali više povezanih projekata u jednu jedinicu, tako da Vaši projekti mogu da dele komponente. Svaki projekat u rešenju održava svoju individualnost i možete ili da editujete projekat u rešenju, ili da ga otvorite kao projekat i editujete nezavisno od drugih projekata u rešenju.

Takođe ste učili kako da razvijate web aplikacije. Sa VB.NET-om, razvoj tih aplikacija je lak kao i razvoj Windows aplikacija. Za nekoliko godina treba da budete sposobni da dizajnirate jedan interfejs koji može da bude korišćen za oba tipa projekta (čak i ako to znači da neće biti drugih aplikacija sem web aplikacija). Korisnički interfejs web aplikacija i Windows aplikacija možda je različit, ali je kod iza oba tipa projekta čist Visual Basic.

Pošto ste razvili aplikaciju, moraćete da je distribuirate. Distribuiranje Windows aplikacije nije lak proces, ali izgradnja Setup programa za Vašu aplikaciju sa VB.NET-om jeste. Sve što treba da uradite je da dodate Setup projekat rešenju koje sadrži projekat, ili projekte koje želite da distribuirate. Najjednostavniji tip Setup programa ne zahteva nikakav kod i možete kreirati Windows instaler samo podešavanjem nekoliko svojstava. Izlaz Setup projekta je fajl sa ekstenzijom .msi, koji možete da kopirate na drugi računar. Kada se izvrši na ciljnom računaru, MSI fajl će instalirati aplikaciju, kreirati skraćenicu prema aplikaciji u Programs meniju korisnika i, čak, kreirati ulaz u Add/Remove Programs za popravku, ili deinstalaciju aplikacije.

Za sada, imate dobru predstavu o okruženju i kako su izgrađene Windows aplikacije. U sledeća dva poglavlja učićete o samom jeziku.