

Stare COM kontrole

Većina VB6 programera razvila je biblioteku COM kontrola i s pravom su zabrinuti da će prelaskom na .NET platformu taj rad biti izgubljen.

Microsoft se obvezao da će se te stare komponente moći koristiti i u .NET aplikacijama i (možda manje važno) da će se .NET komponente lako pozivati iz COM okruženja.

U ovom ćete poglavlju dodati relativno jednostavnu COM kontrolu u NorthWind-Windows aplikaciju.

Uvoz ActiveX kontrola

ActiveX kontrole su COM komponente koje možete dodati na obrazac. Mogu i ne moraju imati korisničko sučelje. Kad je Microsoft razvio OCX standard, koji je programerima omogućio razvoj ActiveX kontrola u Visual Basicu i njihovu upotrebu s jezikom C++ (i obratno), započela je revolucija ActiveX kontrola. Tijekom prošlog desetljeća stvoreno je, prodano i upotrijebljeno na tisuće takvih kontrola. Male su, jednostavne za rad i učinkovit primjer mogućnosti ponovne upotrebe.

Uvoz ActiveX kontrola u .NET je iznenađujuće jednostavan, ako se uzme u obzir koliko se COM objekti razlikuju od .NET objekata. Visual Studio 2005 može automatski uvesti ActiveX kontrole.



Kao alternativu Visual Studiju, Microsoft je razvio pomoćni program za naredbeni red, AxImp, koji stvara sklopove potrebne za upotrebu ActiveX kontrola u .NET aplikaciji.

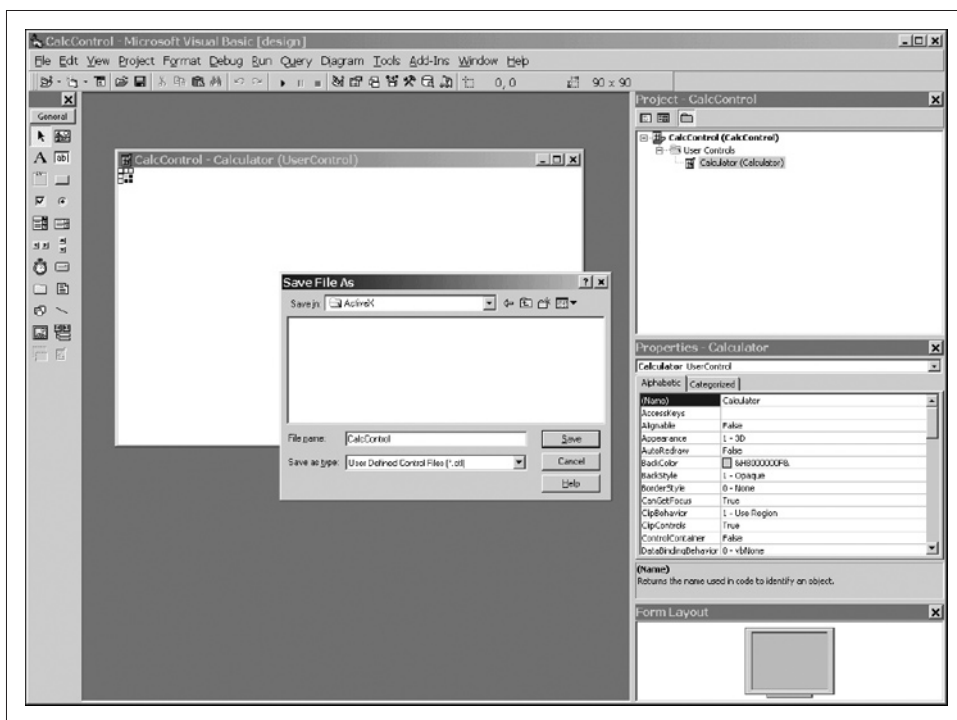
Stvaranje ActiveX kontrole

Za demonstraciju mogućnosti upotrebe klasičnih ActiveX kontrola u .NET aplikaciji, najprije ćete razviti jednostavan kalkulator s četiri funkcije kao ActiveX kontrolu. Kontrolu ćete razviti na jeziku VB6, a zatim je uvesti u Windows Forms aplikaciju.



Ako nemate VB6, gotov projekt možete preuzeti sa stranica tvrtke O'Reilly ili s adrese <http://www.LibertyAssociates.com> (pritisnite Books, dođite do naslova ove knjige i pritisnite hipervezu za izvorni kod). Kada budete imali kontrolu, možete pokrenuti Regsvr32 kako biste ju registrirali.

Da biste stvorili kontrolu, pokrenite VB6 i načinite novi projekt te odaberite ActiveX Control kao vrstu projekta. Obrazac projekta načinite što je moguće manjim, jer ova kontrola neće imati korisničko sučelje. Desnom tipkom miša pritisnite UserControl1 i odaberite Properties. Promijenite naziv kontrole u Calculator u prozoru Properties. Pritisnite Project u prozoru s projektima te ga u prozoru Properties preimenujte u CalcControl. Spremite projekt i datoteku te i jedno i drugo nazovite CalcControl, kao na slici 7-1.



Slika 7-1. Stvaranje VB6 kontrole

Sada možete dodati četiri funkcije kalkulatora tako što ćete desnom tipkom miša pritisnuti obrazac CalcControl, odabrati View Code s padajućeg izbornika te upisati kod iz primjera 7-1.

Primjer 7-1. Implementacija ActiveX kontrole u VB6

```
Public Function _  
Add(left As Double, right As Double) _  
As Double
```

Primjer 7-1. Implementacija ActiveX kontrole u VB6

```
Add = left + right
End Function

Public Function _
Subtract(left As Double, right As Double) _
As Double
    Subtract = left - right
End Function

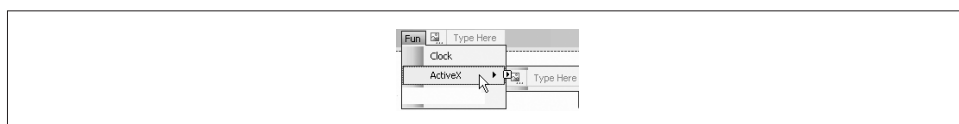
Public Function _
Multiply(left As Double, right As Double) _
As Double
    Multiply = left * right
End Function

Public Function _
Divide(left As Double, right As Double) _
As Double
    Divide = left / right
End Function
```

I to je cjelokupni kod kontrole. Želite li ga testirati u VB6 okruženju prije nego što ga uvezete u .NET, prevedite kontrolu u datoteku *CalcControl.ocx* tako što ćete odabrati File → Make CalcControl.ocx. Ili, možete odvući kontrolu na .NET obrazac (što ćemo objasniti kasnije), a .NET će izgraditi i registrirati kontrolu umjesto vas.

Uvoz kontrole u .NET okruženje

Dodajte novi obrazac u NorthWindWindows aplikaciju i nazovite ga *frmActiveX*. Modificirajte izbornik na uvodnoj stranici tako da Clock i ActiveX budu podizbornici novog izbornika Fun, kao na slici 7-2.

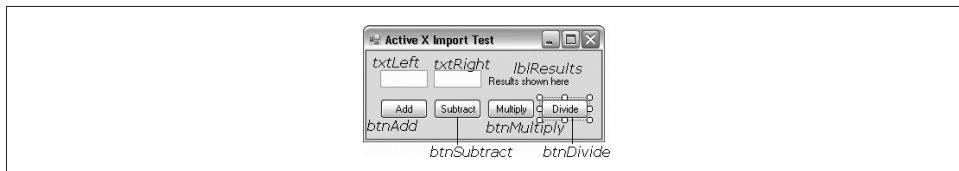


Slika 7-2. Dodavanje podizbornika ActiveX

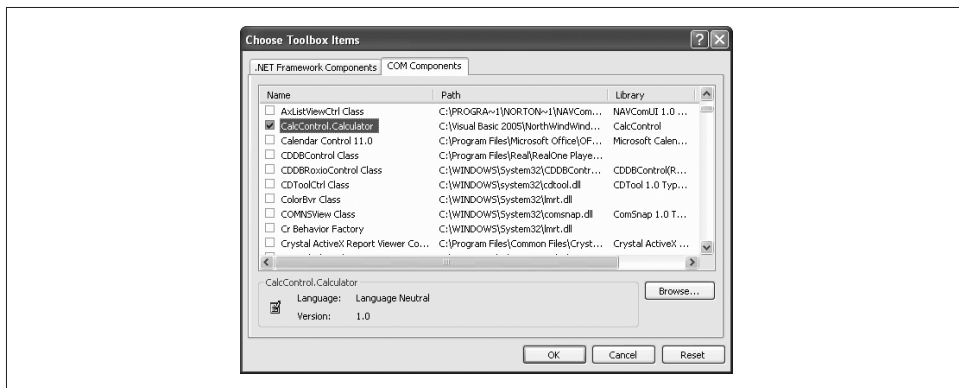
Vratite se u *frmActiveX* i dodajte potrebne kontrole kako biste testirali ActiveX kontrolu (dva polja s tekстом, četiri gumba i natpis), kao na slici 7-3.

Uvoz kontrole

Da biste uvezli kontrolu odaberite Tools → Choose Toolbox Items.... Kad se otvori dijaloški okvir Choose Toolbox Items, odaberite karticu COM Components te pronađite objekt *CalcControl.Calculator* koji ste upravo registrirali (slika 7-4).

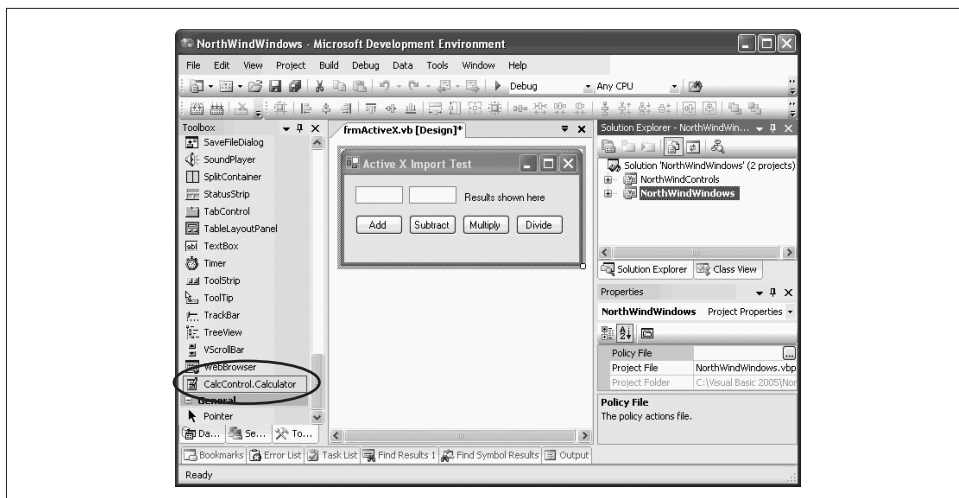


Slika 7-3. Testni obrazac za ActiveX kontrolu



Slika 7-4. Dijaloški okvir Choose Toolbox Items

Budući da je objekt CalcControl registriran na računalo s .NET platformom, Visual Studio 2005 Customize Toolbox ga može pronaći. Odabirom kontrolu u ovom dijaloškom okviru, ona se uvozi u aplikaciju. Visual Studio se brine o pojedinostima uvoza, uključujući i dodavanje kontrolu na paletu alata (slika 7-5).



Slika 7-5. Kontrola CalcControl na paleti alata

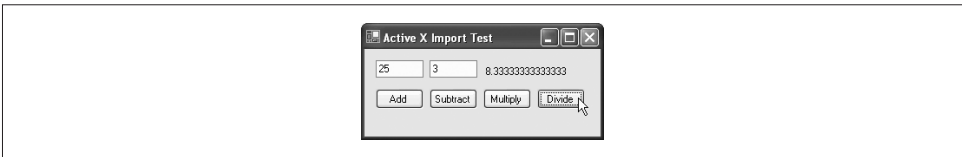
Sada ovu kontrolu možete odvući na Windows obrazac i iskoristiti njezine funkcije. Dodajte metode za obradu događaja za sva četiri gumba. One će delegirati svoj posao ActiveX kontroli koju ste napisali u VB6 i uvezli u .NET.

Kod metoda za obradu događaja prikazan je u primjeru 7-2.

Primjer 7-2. Implementiranje metoda za obradu događaja koje koriste CalcControl ActiveX kontrolu

```
Private Sub btnAdd_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAdd.Click  
    Dim left As Double = Double.Parse(txtLeft.Text)  
    Dim right As Double = Double.Parse(txtRight.Text)  
    lblResults.Text = Me.AxCalculator1.Add(left, right)  
End Sub  
  
Private Sub btnSubtract_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnSubtract.Click  
    Dim left As Double = Double.Parse(txtLeft.Text)  
    Dim right As Double = Double.Parse(txtRight.Text)  
    lblResults.Text = Me.AxCalculator1.Subtract(left, right)  
End Sub  
  
Private Sub btnMultiply_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnMultiply.Click  
    Dim left As Double = Double.Parse(txtLeft.Text)  
    Dim right As Double = Double.Parse(txtRight.Text)  
    lblResults.Text = Me.AxCalculator1.Multiply(left, right)  
  
End Sub  
  
Private Sub btnDivide_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnDivide.Click  
    Dim left As Double = Double.Parse(txtLeft.Text)  
    Dim right As Double = Double.Parse(txtRight.Text)  
    lblResults.Text = Me.AxCalculator1.Divide(left, right)  
End Sub
```

Svaka od implementiranih metoda uzima vrijednosti iz polja za tekst, pretvara ih u Double koristeći zajedničku metodu Double.Parse i proslijeđuje metodama kalkulatora. Rezultati se pretvaraju u niz znakova i umeću u natpis, kao na slici 7-6.



Slika 7-6. Funkcionalna ActiveX kontrola

Uvoz COM komponenti

Mnoge COM komponente koje pojedine tvrtke razvijaju nisu ActiveX kontrole. Radi se o standardnim COM dinamički povezanim bibliotekama (DLL datotekama). Da biste vidjeli kako ih koristiti sa .NET platformom, vratite se u VB6 i načinite COM poslovni objekt koji će se ponašati jednako kao i komponenta iz prethodnog odjeljka.



Još jednom, ako nemate VB6, DLL možete preuzeti na ranije opisani način.

Prvi korak je načiniti novi ActiveX DLL projekt. VB6 tako stvara standardne COM DLL datoteke. Klasu nazovite `ComCalc` a projekt `ComCalculator`. Spremite datoteku i projekt. Kopirajte metodu iz primjera 7-1 (još jednom je prikazana u primjeru 7-3) u prozor s kodom.

Primjer 7-3. Metode za VB6 COM DLL ComCalc

```
Public Function _
Add(left As Double, right As Double) _
As Double
    Add = left + right
End Function

Public Function _
Subtract(left As Double, right As Double) _
As Double
    Subtract = left - right
End Function

Public Function _
Multiply(left As Double, right As Double) _
As Double
    Multiply = left * right
End Function

Public Function _
Divide(left As Double, right As Double) _
As Double
    Divide = left / right
End Function
```

Integriranje COM DLL datoteke u .NET

Načinite novi obrazac pod nazivom `frmCOMDLL`. Dodajte stavku u izbornik `Fun` na uvodnom obrascu koja poziva taj obrazac.

Kad ste načinili `ComCalc` DLL datoteku možete ju uvesti u .NET. No, prije uvoženja morate se odlučiti za rano ili kasno povezivanje. Kad klijent poziva metodu na poslužitelju, potrebno je otkriti adresu poslužiteljeve metode u memoriji. Taj se postupak naziva *povezivanje* (engl. *binding*).

Pri *ranom povezivanju* (engl. *early binding*) adresa metode na poslužitelju otkriva se kad je projekt klijenta preveden i kad se modulu dodaju metapodaci. Pri *kasnom povezivanju* (engl. *late binding*) otkrivanje adrese se ne događa prije izvođenja koda, kad COM ispituje poslužitelj da li podržava tu metodu.

Rano povezivanje ima mnoge prednosti. Najznačajnija su performanse. Rano povezane metode se pozivaju brže od kasno povezanih. Da bi prevoditelj izveo rano povezivanje, mora ispitati COM objekt. Ako će prevoditelj ispitivati poslužiteljsku biblioteku tipova, ona se najprije mora uvesti u .NET.

Uvoženje biblioteke tipova

VB6 COM DLL sadrži biblioteku tipova, ali se format biblioteke COM tipova ne može koristiti u .NET aplikaciji. Da biste riješili ovaj problem morate uvesti biblioteku COM tipova u sklop. To možete učiniti na dva načina: možete dopustiti razvojnom okruženju uvoženje klase dodavanjem reference na komponentu, ili možete uvesti biblioteku tipova ručno, upotrebom pomoćnog programa *TlbImp.exe*.



TlbImp.exe će napraviti interoperacijski sklop. .NET objekt koji umata COM objekt naziva se *Runtime Callable Wrapper* (RCW). .NET klijent će koristiti RCW za povezivanje s metodama u COM objektu, kao što je prikazano u sljedećem odjeljku.

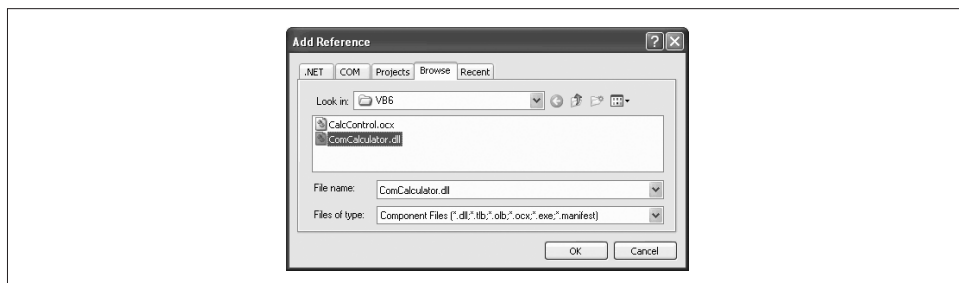
Odaberite karticu COM u dijaloškom okviru Add Reference i zatim registrirani COM objekt, kao na slici 7-7.

Time se poziva program *TlbImp* koji će kopirati rezultirajući RCW u mapu *C:\Documents and Settings\Administrator\Application Data\Microsoft\VisualStudio\RCW*.



Točna mapa u koju će RCW biti smješten može se razlikovati na vašem računalu.

Morat ćete biti pažljivi jer DLL koji se stvara ima isti naziv kao i COM DLL.



Slika 7-7. Dodavanje reference na *ComCalculator.dll*

Stvaranje testnog programa

Vratite se u frmActiveX i odaberite sve kontrole (osim ActiveX kontrole). Zatim ih kopirajte na Clipboard. Prebacite se na frmCOMDLL i kopirajte kontrole sa Clipboarda na novi obrazac. Pomaknite ih na njihova mjesta. Primjetit ćete da su sve zadržale nazive i metode za obradu događaja.

Dodajte klasi frmCOMDLL sljedeću varijablu članicu:

```
Private theCalc As New ComCalculator.ComCalc
```

Pritisnite metodu za obradu događaja i dodajte kod iz primjera 7-4 kojim se pozivaju metode objekta ComCalc.

Primjer 7-4. Upotreba ComCalculator DLL datoteke

```
Public Class frmCOMDLL
    Private theCalc As New ComCalculator.ComCalc

    Private Sub btnAdd_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnAdd.Click
        Me.lblResults.Text = theCalc.Add(Double.Parse(txtLeft.Text), _
            Double.Parse(txtRight.Text))
    End Sub

    Private Sub btnSubtract_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnSubtract.Click
        Me.lblResults.Text = theCalc.Subtract(Double.Parse(txtLeft.Text), _
            Double.Parse(txtRight.Text))
    End Sub

    Private Sub btnMultiply_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnMultiply.Click
        Me.lblResults.Text = theCalc.Multiply(Double.Parse(txtLeft.Text), _
            Double.Parse(txtRight.Text))
    End Sub

    Private Sub btnDivide_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnDivide.Click
        Me.lblResults.Text = theCalc.Divide(Double.Parse(txtLeft.Text), _
            Double.Parse(txtRight.Text))
    End Sub
End Class
```

Kasno povezivanje

Da biste pozvali COM objekt s kasnim povezivanjem u Visual Basicu 2005 morate koristiti *refleksiju* (pogledajte izdvojeni odlomak „Refleksija“), premda je to neobičan zahtjev i stoga napredna tema (to jest, slobodno preskočite ovaj odlomak, neću se uvrijediti).

Refleksija

Refleksija je način na koji program promatra samog sebe ili metapodatke nekog drugog programa. Refleksija se općenito koristi za četiri zadatka:

Pregled metapodataka

Metapodaci su podaci koji su dio programa, ali se ne izvode. Metapodaci mogu uključivati informacije o verziji, posebnim atributima koji alatima kao što je Visual Studio 2005 govore kako prikazati kontrolu i tako dalje. U COM okruženju metapodaci se čuvaju u biblioteci tipova, dok se u .NET okruženju oni čuvaju u samim programima.

Otkrivanje tipova

Omogućava ispitivanje tipova u nekom sklopu te interakciju s njima ili njihovo instanciranje. To može biti korisno kada korisnicima želite dopustiti interakciju s programom koristeći skriptni jezik, primjerice JavaScript, ili neki skriptni jezik koji ćete sami razviti.

Kasno povezivanje s metodama i svojstvima

Omogućava programerima pozivanje svojstava i metoda za objekte koji su dinamički instancirani na temelju otkrivanja tipova. To je poznato i kao *dinamičko pozivanje* (engl. *dynamic invocation*). Za to ćemo u ovom poglavlju koristiti refleksiju.

Stvaranje tipova pri izvođenju

Naziva se i refleksijsko emitiranje (engl. *reflection-emit*) – tajnovita i napredna upotreba refleksije koja omogućava dinamičko stvaranje i pokretanje programa.

Da biste vidjeli kako koristiti kasno povezivanje, uklonite referencu prema uvezenoj `com` biblioteci. Četiri metode za obradu događaja gumba sada morate ponovno napisati. Više ne možete instancirati `ComCalculator.comCalc` objekt pa njegove metode morate pozivati dinamički.

Budući da sve četiri metode za obradu događaja moraju replicirati refleksiju nad objektom, razlikujući se samo po metodi koju pozivaju, zajednički kod ćete faktorirati u privatnu pomoćnu metodu `DoInvoke`. Svaka metoda za obradu događaja pritiska na gumb poziva ovu metodu s nazivom odgovarajuće ciljne metode (`Add`, `Subtract`, `Multiply` ili `Divide`), kao u primjeru 7-5.

Primjer 7-5. Kod za kasno povezivanje metoda za obradu događaja pritiska na gumb Add, Subtract, Multiply i Divide

```
Private Sub btnAdd_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnAdd.Click  
    DoInvoke("Add")  
End Sub  
  
Private Sub btnSubtract_Click( _  
ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btnSubtract.Click
```

Primjer 7-5. Kod za kasno povezivanje metoda za obradu događaja pritiska na gumb Add, Subtract, Multiply i Divide (nastavak)

```
DoInvoke("Subtract")
End Sub

Private Sub btnMultiply_Click( _
ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnMultiply.Click
DoInvoke("Multiply")
End Sub

Private Sub btnDivide_Click( _
ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles btnDivide.Click
DoInvoke("Divide")
End Sub
```

Prije nego što implementirate DoInvoke, morate dodati dvije varijable članice u klasu frmCOMDLL. Jedna od njih je Type objekt koji će sadržavati informacije o tipu comCalc. Potreban vam je i generički Object koji će predstavljati COM objekt kojeg ćete instancirati.

```
Private comCalcType As Type = Type.GetTypeFromProgID("ComCalculator.ComCalc")
Private comCalcObject As Object
```

Poziv metode GetTypeFromProgID govori .NET kosturu da otvori registriranu COM DLL datoteku i uzme potrebne informacije o tipu za određeni objekt.

Zatim dodajte metodu za obradu događaja Load za frmCOMDLL. U ovoj metodi pozovite CreateInstance da vrati instancu comCalc objekta, kao u primjeru 7-6.

Primjer 7-6. Metoda za obradu događaja Load za COMDLL

```
Private Sub frmCOMDLL_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
comCalcObject = Activator.CreateInstance(comCalcType)
End Sub
```

Sada kada imate metapodatke tipa za COM objekt i kada ste instancirali objekt tog tipa, možete implementirati DoInvoke. Neizravno će pozvati metode COM objekta (Add, Subtract, Multiply i Divide) pozivanjem metode InvokeMember klase Type. Točno to biste i vi učinili da pozivate metode refleksijom klase opisane unutar .NET sklopa.

U metodi DoInvoke najprije načinite polje koje će sadržavati argumente metode:

```
Dim left As Double = Double.Parse(txtLeft.Text)
Dim right As Double = Double.Parse(txtRight.Text)
Dim inputArguments As Object = New Object(1) {left, right}
```



Primijetite da konstruktor za polje objekata uzima gornju granicu (1), što govori da će to polje sadržavati dva objekta.

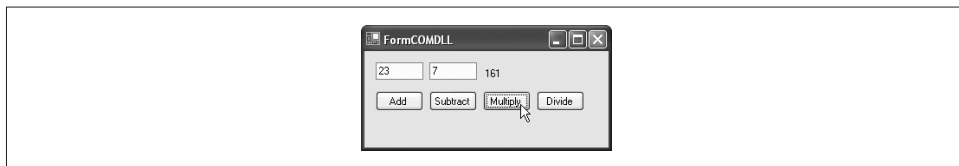
Type.InvokeMember očekuje nekoliko argumenata:

- Metodu koju želite pozvati kao niz znakova (Add, Subtract, Multiply ili Divide)
- Oznaku povezivanja (postavljenu na Reflection.BindingFlags.InvokeMethod)
- Veznik (postavljen na Nothing)
- Objekt kojeg vraća CreateInstance()
- Polje ulaznih argumenata

Rezultati ovog poziva pretvaraju se u Double te pohranjuju u lokalnu varijablu result:

```
result = Double.Parse(comCalcType.InvokeMember( _  
    whichMethod, _  
    Reflection.BindingFlags.InvokeMethod, _  
    Nothing, _  
    comCalcObject, _  
    inputArguments))
```

Zatim taj rezultat možete prikazati u korisničkom sučelju, kao što je prikazano na slici 7-8.



Slika 7-8. Pokretanje s kasnim povezivanjem

Cjelokupna implementacija za DoInvoke prikazana je u primjeru 7-7.

Primjer 7-7. Implementacija metode DoInvoke()

```
Public Sub DoInvoke(ByVal whichMethod As String)  
    Dim left As Double = Double.Parse(txtLeft.Text)  
    Dim right As Double = Double.Parse(txtRight.Text)  
    Dim result As Double = Nothing  
  
    Dim inputArguments As Object = New Object(1) {left, right}  
    result = Double.Parse(comCalcType.InvokeMember( _  
        whichMethod, _  
        Reflection.BindingFlags.InvokeMethod, _  
        Nothing, _  
        comCalcObject, _  
        inputArguments))  
  
    Me.lblResults.Text = result.ToString( )  
  
End Sub
```