POGLAVLJE 1

Priprema, pozor, kreni!

Ova knjiga se prvenstveno odnosi na programski jezik Visual Basic. Iz nje ćete naučiti kako se postaje iskusan Visual Basic programer. Ukoliko je proučite od korica do korica, nećete postati prvoklasan programer, već programer koji zna šta treba da uradi kada pravi robustne, stabilne i održive Visual Basic aplikacije.

U ovom poglavlju ćete prvo nabaviti alatke koje su potrebne za Visual Basic aplikacije i isprobaćete ih. Usput ćete napraviti nekoliko Visual Basic aplikacija.

Preuzimanje i instaliranje alatki

Kada počinjete da koristite Visual Basic 2008, sigurno želite da napišete programski kod koji nečemu služi. To osećanje je slično onom kada dobijete vozačku dozvolu i jedva čekate da sednete u automobil, ne razmišljajući gde zapravo treba da idete - želite samo da vozite. Dobra strana .NET-a je da programski kod možete da pišete čim instalirate .NET okruženje (.NET SDK) ili Visual Studio okruženje za pravljenje programa (IDE). Preuzimanje i instaliranje odgovarajućeg okruženja je veoma važan posao koji utiče na komforno pisanje programskog koda.

NAPOMENA

Verzija softvera, opis proizvoda i tehnologije Vas mogu zbuniti. Pošto „Microsoftove" tehnologije koristim više od 10 godina, mogu slobodno da kažem da imenovanje tehnologija ili proizvoda nikada nije bila jača strana „Microsofta". Tehnologije su bile (uglavnom) izuzetne, ali klasifikacija i identifikacija proizvoda nisu bile na tom nivou. Prema tome, ova knjiga se odnosi na programski jezik Visual Basic 2008 koji se koristi za pisanje aplikacija za okruženje .NET Framework. U Visual Basicu 2008 se koriste okruženja .NET 3.0 i .NET 3.5. Okruženje .NET 3.0 obuhvata najvažnije osobine, dok okruženje .NET 3.5 obuhvata dodatne osobine.

Za primere u knjizi Vam je potreban Visual Basic 2008 Express Edition. Ova verzija programskog jezika je besplatna i sadrži sve što je neophodno za učenje programskog jezika Visual Basic 2008. „Microsoft" je ostale verzije Express Edition IDE-a napravio za druge programske jezike (C# i C++) ili IDE, kao što je slučaj sa verzijom Visual Web Developer Express, sadrži specifične osobine koje su previše restriktivne za naše potrebe.

„Microsoft“ nudi kompletne verzije Visual Studio IDE-a, kao što su Standard, Professional i Team. Svaka od njih obuhvata drugačije osobine i cena im je drugačija. Više informacija ćete pronaći na web sajtu Microsoft Visual Studio (`http://msdn2.microsoft.com/en-us/vstudio/default.aspx`). Ukoliko ste već kupili Microsoft Visual Studio 2008 Professional, možete ga koristiti za primere koje ćete pronaći u knjizi. Ta verzija može sve što i Visual Basic 2008 Express, a pruža i mnogo više mogućnosti.

NAPOMENA

Ja koristim Visual Studio Standard ili Professional, zajedno sa drugim alatkama, kao što su X-develop i JustCode!, koje je napravio Omnicore (`http://www.omnicore.com`), TestDriven.NET (`http://www.test-driven.net`) i NUnit (`http://www.nunit.org`). Verzije Visual Studia su veoma dobri proizvodi, ali postoje i drugi. Da biste bili dobar programer, morate znati koje alatke treba da upotrebite.

Instaliranje i preuzimanje verzije Visual Basic Express sa „Microsoftovog“ web sajta podrazumevaju prenošenje velikih datoteka. Ukoliko nemate brzu vezu sa Internetom, savetujem Vam da IDE instalirate sa CD-a.

Preuzimanje verzije Visual Basic Express

Na sledeći način ćete sa „Microsoftovog“ web sajta preuzeti Visual Basic Express. Postupak preuzimanja će se možda razlikovati u vreme kada budete čitali knjigu, ali će biti dovoljno sličan da možete da pronađete i preuzmete IDE pakete.

1. Posetite adresu `http://www.microsoft.com/express`.
2. Kliknite link Download Now!.
3. Na stranici pronađite odeljak Visual Basic 2008 Express Edition (pogledajte sliku 1-1).
4. Kliknite link Download.
5. Biće prikazan okvir za dijalog u kojem treba da odaberete mesto na koje ćete zapisati datoteku koju preuzimate. Datoteka koju preuzimate je datoteka pomoću koje će započeti postupak instaliranja Visual Basic Express IDE-a. Zapišite je na radnu površinu.

Prethodni postupak ćete obaviti veoma brzo – verovatno za nekoliko minuta. On nije zamena za preuzimanje kompletne Visual Basic Express aplikacije. Prilikom instaliranja ćete preuzeti veći deo IDE-a.

SLIKA 1-1 *Odaberite Visual Basic 2008 Express Edition.*

Instaliranje verzije Visual Basic 2008 Express

Pošto preuzmete datoteku, možete započeti instaliranje verzije Visual Basic Express. Tokom ovog postupka se svi delovi IDE-a (oko 300 MB) preuzimaju i instaliraju. Da biste instalirali verziju Visaual Basic Express, treba da uradite sledeće:

1. Dva puta kliknite datoteku vbsetup.exe, koja se nalazi na radnoj površini. Sačekajte da program učita sve neophodne komponente.
2. U prvom prozoru kliknite Next.
3. Biće prikazivan niz okvira za dijalog. Prihvatite ponuđene opcije i kliknite Next da biste nastavili kroz program. U poslednjem okviru za dijalog kliknite Install.
4. Možda ćete, pošto se sve komponente preuzmu i instaliraju, morati da restartujete računar.

Pošto instalirate Visual Basic Express, možete ga pokrenuti iz menija Start.

Izbor tipa aplikacije

Pošto pokrenete Visual Basic Express, možete da napišete svoju prvu .NET aplikaciju. Međutim, prvo morate da odaberete tip aplikacije koji ćete praviti. Uopšteno govoreći, u .NET-u možete da napravite tri osnovna tipa programa:

- *konzolnu aplikaciju (engl. console application)*, koja se izvršava sa komandnog prompta i nema grafički korisnički interfejs (GUI)
- *Windows aplikaciju* koja se izvršava na radnoj površini i ima GUI
- *biblioteku klasa* koja sadrži funkcije koje se mogu koristiti u konzolnim i Windows aplikacijama (ovaj tip programa se ne može samostalno izvršavati)

Pošto sada znate kakve tipove programa možete napraviti, u ovom poglavlju ćete ih napraviti. To će biti varijacije primera „Hello, World" koji na ekranu prikazuje reči „hello, world". Ovakvi programi se već decenijama koriste za demonstriranje programskih jezika.

Pravljenje projekata i rešenja

Kada koristite neki od Visual Studio proizvoda, bez obzira koje programe pravite, Vi pravite projekte i rešenja:

- *Projekat j*e klasifikacija koja se koristi za opisivanje tipa .NET aplikacije.
- *Rešenje* je klasifikacija koja se koristi za opisivanje više .NET aplikacija koje su najverovatnije povezane.

Zamislite da pravite automobil. Projekat bi onda bio volan, motor ili karoserija. Spajanjem svih projekata u jednu celinu dobilo bi se kompletno rešenje, koje se naziva automobil.

Rešenje sadrži projekte. *Rešenje ć*e u primerima ovog poglavlja sadržati tri projekta koja predstavljaju tri različita tipa programa.

Kada koristite Visual Basic Express, pravljenje projekta implicira pravljenje rešenja, jer pravljenje praznog rešenja bez projekta nema mnogo smisla. To je kao pravljenje automobila bez delova. Kada u ovoj knjizi upotrebljavam termine „projekat" ili „aplikacija", iz perspektive organizacije radnog prostora oni imaju isto značenje. Rešenje se eksplicitno odnosi na jedan ili više projekata ili aplikacija.

Naš plan rada, kada govorimo o projektima i rešenjima, u ovom poglavlju će biti ovakav:

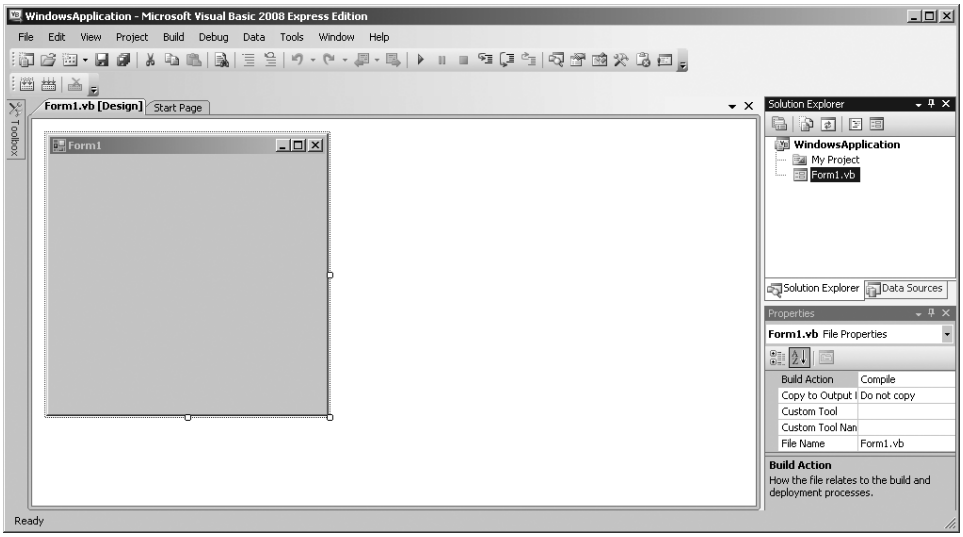
- Napravićemo .NET rešenje, tako što ćemo napraviti `Windows aplikaciju` WindowsApplication (pravljenjem aplikacije ćemo napraviti rešenje).
- Rešenju ćemo dodati konzolnu aplikaciju `ConsoleApplication`.
- Rešenju ćemo dodati biblioteku klasa `ClassLibrary`.

Pravljenje Windows aplikacije

Odmah ćemo početi pravljenje Windows aplikacije. Pošto se Visual Basic Express izvršava, treba da uradite sledeće da biste napravili Windows aplikaciju:

1. Odaberite File ⇨ New Project.
2. Odaberite ikonu Windows Forms Application. Njom je predstavljen projekat koji se zasniva na pripremljenom šablonu Windows Forms Application.
3. Promenite naziv projekta u `WindowsApplication`.
4. Kliknite OK.

Na ovaj način ćete istovremeno napraviti novi projekat i rešenje: u Visual Basic Expressu se prikazuje samo kompletan projekat (pogledajte sliku 1-2).



SLIKA 1-2 *Visual Basic Express IDE u kojem je prikazan projekat WindowsApplication*

Prikazivanje programskog koda

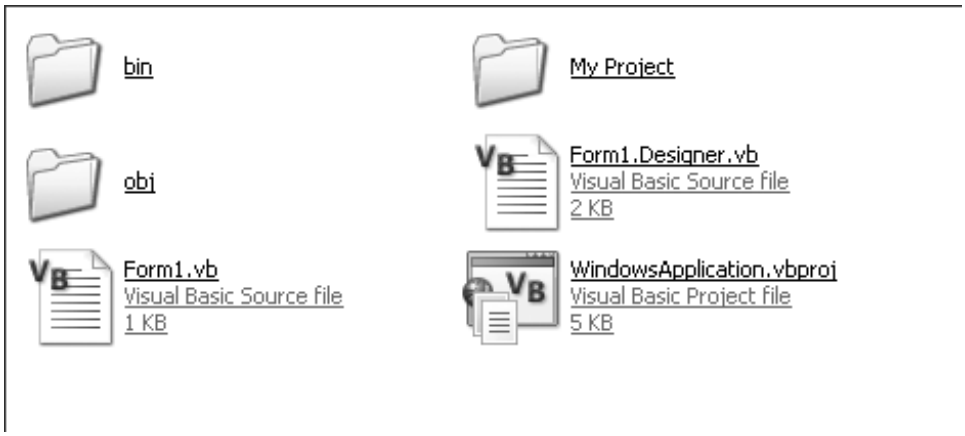
Kada pravite novu aplikaciju, Visual Basic Express automatski generiše programski kod. Desnim tasterom miša kliknite element `Form1.vb`, koji se nalazi u Solution Exploreru, i u kontekstnom meniju odaberite View Code. Sledeći programski kod će biti prikazan levo od Solution Explorera.

NAPOMENA

Da biste naizmenično prikazivali korisnički interfejs i generisani programski kod u Solution Exploreru, desnim tasterom miša kliknite `Form1.vb`. U podmeniju ćete videti opcije View Code (prikazivanje programskog koda) i View Designer (prikazivanje korisničkog interfejsa).

```
Public Class Form1
End Class
```

Generisani programski kod u Visual Basicu je „ogoljen", jer je Visual Basic alatka koja se nekada nazivala okruženje za brzo pravljenje aplikacija (RAD). Osnovna ideja u Visual Basicu je da se aplikacije prave što brže, tako da ezoterične mogućnosti programskog jezika ne predstavljaju problem. Takvo nasleđe ima dobre i loše strane. Na slici 1-2 vidite jednostavan projekat koji ima jednu datoteku, ali na hard disku postoji još jedna datoteka koja je prikazana na slici 1-3 (nju ćete prikazati tako što ćete u Solution Exploreru kliknuti ikonu Show All Files i otvoriti čvor `Form1.vb`).



SLIKA 1-3 *Sve datoteke koje sačinjavaju projekat WindowsApplication*

Datoteka `Form1.Designer.vb` je u prethodnim verzijama bila u binarnom obliku i njen sadržaj nije mogao da se menja. Sada je to tekstualna datoteka čiji sadržaj možete da promenite, ali to ne treba da činite, jer je za njen sadržaj odgovoran IDE. U datoteci `Form1.Designer.vb` se nalaze informacije koje su potrebne za pravljenje elementa Form1 (pogledajte sliku 1-2). `From1`, kao ni tekstualna datoteka, za sada ne sadrži ništa vredno pažnje. Međutim, ukoliko na formular postavite komandno dugme ili tekstualno polje, odgovarajuće informacije će biti upisane u tekstualnu datoteku `Form1.Designer.vb`.

Visual Basic je kompletan programski jezik koji korene vuče iz RAD modela. Na primer, pomoću sledećeg programskog koda se pravi korisnički tip podataka (u ovoj knjizi ćete naučiti kako se prave korisnički tipovi podataka):

```
Public Class Example
Public Sub Empty()
End Sub
End Class
```

Treba obratiti pažnju na sledeće elemente:

Class - organizaciona jedinica pomoću koje se grupišu povezani elementi. Ovakvo grupisanje je preciznije od grupisanja u rešenje ili projekat. Prisetimo se analogije sa automobilom. Ukoliko je motor automobila projekat, onda klasa može biti karburator. Drugačije rečeno, projekti su sastavljeni od klasa.

Sub - skup instrukcija pomoću kojih se obavlja posao. Podrutina (engl. sub; skraćeno od subroutine - prim. prev.), koja se često naziva *metod* (engl. *method*), predstavlja isto što i funkcija u mnogim drugim programskim jezicima. Metod `Empty()` se može pozvati iz bilo kojeg dela programskog koda da bi bila obavljena neka akcija.

Zapisivanje projekta

Pošto promenite naziv projekta, poželjno je da zapišete izmene. Da biste zapisali projekat, treba da uradite sledeće:

1. U Solution Exploreru obeležite naziv projekta.

2. Odaberite File ⇨ Save WindowsApplication.
3. Primetićete da Visual Basic Express „želi" da sačuva projekat koristeći naziv `WindowsApplication`, što nije najbolje rešenje (za ovo rešenje ćete napraviti tri projekta, od kojih je jedan aplikacija tipa `Windows Forms`). Da biste zapisali rešenje i promenili mu naziv, naziv `WindowsApplication` treba da promenite u `ThreeExamples` (nemojte menjati naziv projekta `WindowsApplication`). Upamtite mesto na koje zapisujete Visual Basic Express projekte, jer će Vam s vremena na vreme ta informacija biti potrebna.
4. Kliknite komandno dugme Save.

Pošto zapišete rešenje i projekat, u donjem levom uglu prozora će na statusnoj liniji biti prikazana poruka „Item(s) Saved".

Kad god budete kasnije želeli da zapišete rešenje ili projekat, možete upotrebiti tastaturnu prečicu Ctrl+S.

NAPOMENA

Ukoliko niste zapisali izmene i rešite da zatvorite Visual Basic Express, morate odgovoriti na pitanje da li želite da zapišete izmene ili da odbacite projekat ili rešenje.

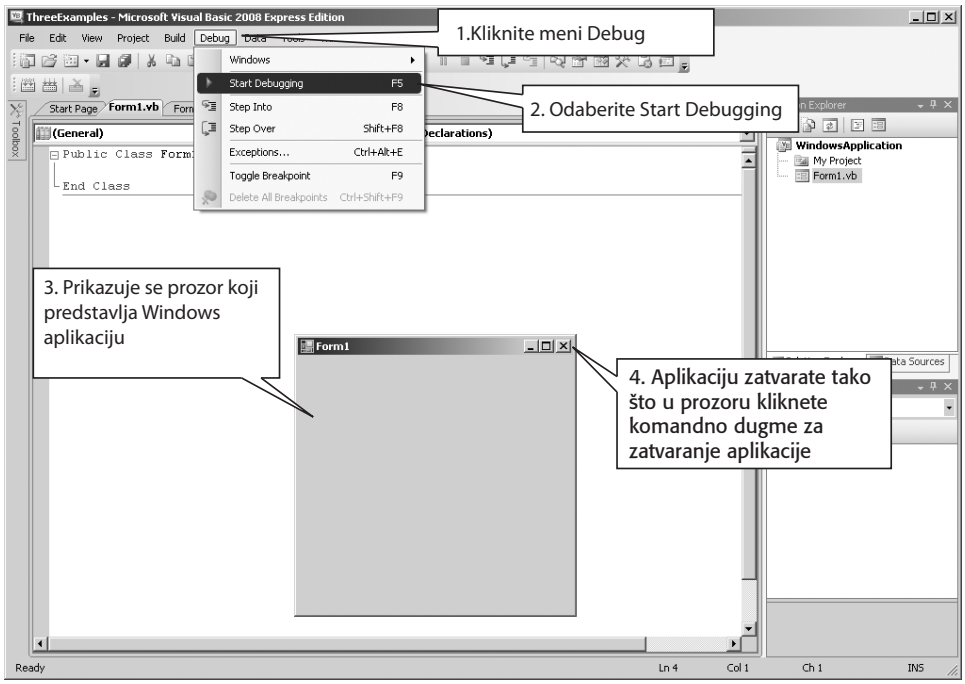
Rešenje koje ste prethodno zapisali ćete otvoriti tako što ćete odabrati File ⇨ Open Project i pronaći datoteku koja sadrži to rešenje. Možete ga odabrati u prozoru Recent Projects, koji se prikazuje kada pokrenete Visual Basic Express. Prozor Recent Projects uvek možete otvoriti koristeći karticu Start Page, koja se nalazi u osnovnom prozoru Visual Basic Expressa (i u meniju File se takođe nalazi spisak projekata koje ste u poslednje otvarali).

Pokretanje Windows aplikacije

Programski kod koji generiše Visual Basic Express je aplikacija koja sadrži prazan prozor. Programski kod je početna tačka od koje možete da dodajete još programskog koda, da taj programski kod debagirate i da pokrenete aplikaciju.

Da biste pokrenuli aplikaciju, odaberite Debug ⇨ Start Debuging. Aplikaciju na drugi način možete pokrenuti tako što ćete pritisnuti taster F5. Videćete prozor koji predstavlja aplikaciju `WindowsApplication`. Aplikaciju ćete zatvoriti tako što ćete u prozoru kliknuti komandno dugme za zatvaranje aplikacije. Na slici 1-4 je prikazan ovaj postupak (debagiranje je objašnjeno u Poglavlju 5).

Pokretanje aplikacije omogućava da vidite šta ona radi. Pokretanje aplikacije iz IDE-a je isto kao kada bi korisnik aplikaciju pokrenuo sa radne površine. U ovom primeru `WindowsApplication` prikazuje prazan prozor, u kojem nema kontrola i iz kojeg se ništa ne može uraditi. Programski kod služi za prikazivanje praznog prozora i komandnog dumeta pomoću kojeg se prekida izvršavanje aplikacije (i komandnih dugmadi za promenu veličine prozora – za prikazivanje prozora preko celog ekrana i za prikazivanje aplikacije na paleti aktivnih poslova). Zatvorite aplikaciju.



SLIKA 1-4 *Pokretanje aplikacije*

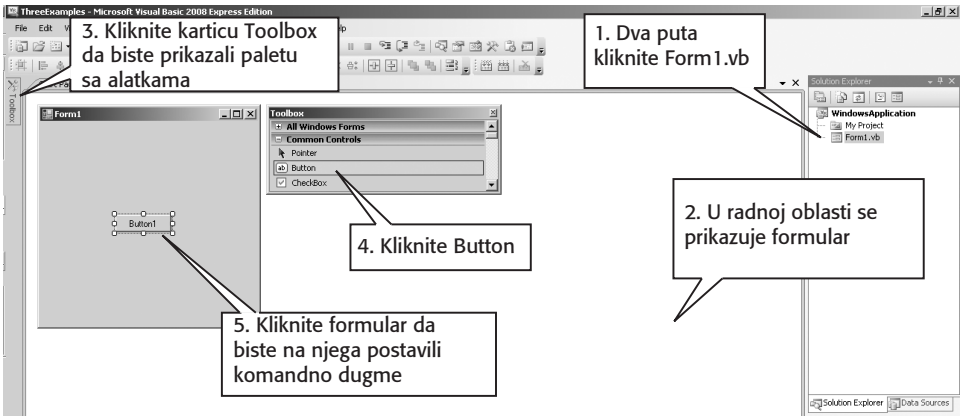
Niste napisali ni jedan programski red, ali ste, ipak, napravili aplikaciju i nešto se dogodilo, zahvaljujući tome što Visual Basic Express generiše programski kod koji se odmah može koristiti.

Napravili ste aplikaciju, prikazali njen programski kod i pokrenuli ste je. Sve to ste uradili u komfornom okruženju Visual Basic Express, koje za Vas obavlja sav posao. Visual Basic Express istovremno ima dobrih i loših strana. Dobra strana je što od Vas sakriva složene detalje, a loša je što su ti detalji sakriveni. Zamislite da ste automehaničar. Dobro je što proizvođači automobila prave instrumet-table sa lampicama koje se pale kada nešto nije u redu. Međutim, bilo bi jako loše da se automehaničar oslanja samo na lampice prilikom popravljanja automobila.

Neka Vas Widnows aplikacija pozdravi

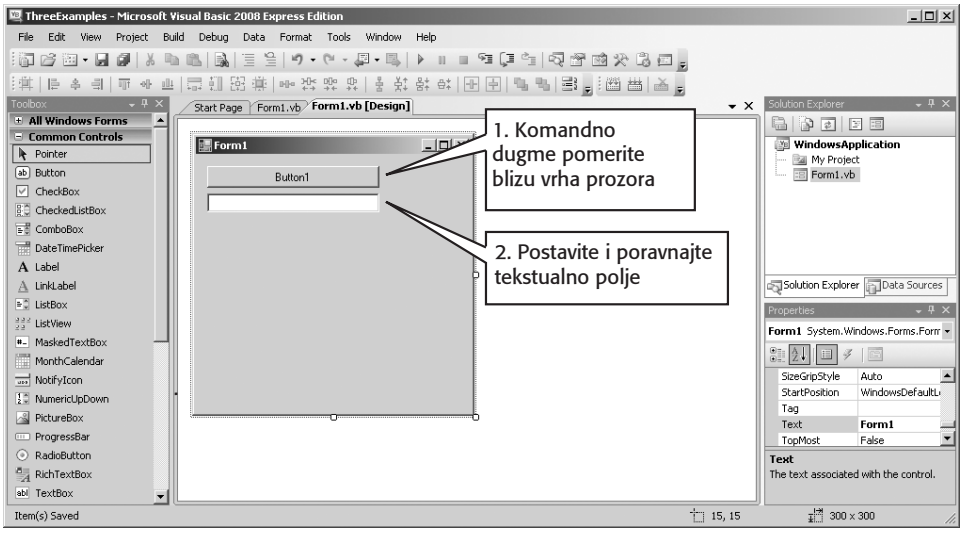
Windows aplikacija za sada ne „radi" ništa, osim što prikazuje prazan prozor koji možete da zatvorite. Da bi aplikacija nešto „radila", morate joj dodati elemente korisničkog interfejsa ili napisati programski kod. Pisanje programskog koda bez dodavanja elemenata korisničkog interfejsa će omogućiti da aplikacija nešto „radi", ali to nije mnogo zanimljivo. Dakle, postavićete komadno dugme pomoću kojeg se, kada se klikne, u tekstualnom polju prikazuje „hello, world".

Na formular prvo treba da postavite kontrolu Button. U Solution Exploreru dva puta kliknite `Form1.vb` da biste prikazali prazan formular. Potom, kliknite karticu Toolbox da biste prikazali kontrole i otvorite karticu Common Controls (kliknite ikonu u obliku čiode da bi kartica Toolbox ostala otvorena, ukoliko to želite). Kliknite Button i formular (da biste na njega postavili kontrolu). Ovaj postupak je prikazan na slici 1-5.



SLIKA 1-5 *Postavljanje komandnog dugmeta na formular*

Potom, na formular postavite kontrolu TextBox, ponavljajući isti osnovni postupak. Poravnajte komandno dugme i tekstualno polje, kao na slici 1-6. Kontrolu ćete pomeriti tako što ćete upotrebiti ručice koje će biti prikazane pošto je kliknete. Kada je pomerate, Visual Basic Express će ivicu kontrole poravnati sa ivicama susednih kontrola, što će omogućiti njihovo precizno poravnavanje.



SLIKA 1-6 *Komandno dugme i tekstualno polje su poravnati.*

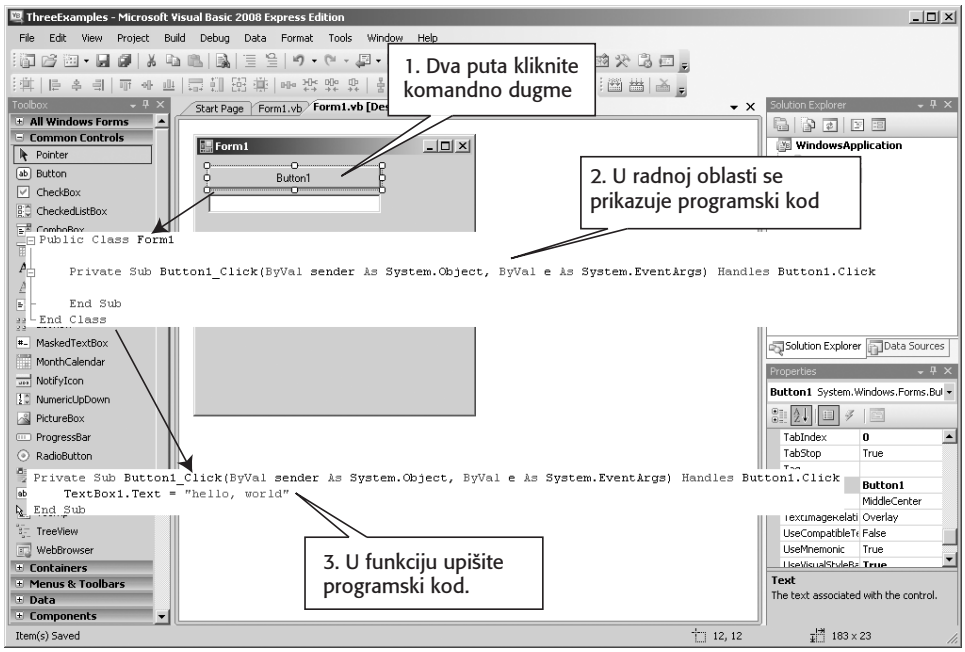
Ukoliko sada pokrenete aplikaciju `WindowsApplication` tako što ćete pritisnuti kombinaciju tastera Ctrl+F5 (pomoću kombinacije tastera Ctrl+F5 se bez debagiranja pokreće aplikacija), biće prikazan prozor sa komandnim dugmetom i tekstualnim poljem. Komandno dugme možete da kliknete i da u tekstualno polje upišete tekst ili da ga u njemu obrišete. Međutim, bilo šta da uradite to neće proizvesti nikakav rezultat, jer niste napisali programski kod za kontrole.

Da bi aplikacija mogla nešto da „uradi", morate razmišljati o *događajima*. Na primer, ukoliko imate garažu sa vratima koja se automatski otvaraju, očekujete da će se, kada na daljinskom upravljaču pritisnete dugme, vrata otvoriti ukoliko su bila zatvorena ili zatvoriti ukoliko su bila otvorena. Proizvođač automatskih vrata je pritiskanje dugmeta na daljinskom upravljaču (događaj) pridružio akciji otvaranja ili zatvaranja vrata garaže. U aplikaciji `WindowsApplication` će akcija prikazivanja teksta u tekstualnom polju biti pridružena događaju kada korisnik klikne komandno dugme.

U Visual Basic Expressu označite komandno dugme koje se nalazi na formularu i dva puta ga kliknite. U radnoj oblasti će biti programski kod, a kursor će se nalaziti u funkciji `Button1_Click()`. U funkciju unesite sledeći programski kod:

```
TextBox1.text = "hello, world"
```

Na slici 1-7 vidite postupak pridruživanja događaja i akcije.



SLIKA 1-7 *Pridruživanje događaja klika mišem komandnog dugmeta i akcije ispisivanja teksta u tekstulanom polju*

Upamtite da je `TextBox1` naziv tekstualnog polja koji ste postavili na formular. Naziv je generisao Visual Basic Express, kao što je podrazumevani naziv generisao za komandno dugme. Podrazumevane nazive možete da promenite (u prozoru Properties svake od kontrola), ali ih u ovom primeru nismo menjali.

Dodavanje akcije za događaj je veoma jednostavan postupak ukoliko pratite uputstva koja su prikazana na slici 1-7. Jednostavnost je posledica prirode Visual Basic Expressa, a ne posledica jednostavnosti događaja ili akcije. Visual Basic Express pretpostavlja da, kada dva puta kliknete kontrolu, želite da izmenite *podrazumevani događaj* (engl. *default event*) kontrole, pa zbog toga automatski generiše programski kod koraka 3 na slici 1-7. Podrazumevani događaj za komandno dugme je događaj kada ga korisnik klikne mišem. Sasvim je logična pretpostavka da je događaj klika mišem podrazumevani događaj. Za ostale kontrole podrazumevani događaji se razlikuju. Na primer, ukoliko dva puta kliknete kontrolu TextBox, generisaćete programski kod za događaj menjanja teksta.

Pokrenite aplikaciju tako što ćete pritisnuti kombinaciju tastera Ctrl+F5, pa kliknite komandno dugme. U tekstualnom polju će biti prikazan tekst „hello, world". Čestitam, upravo ste napravili prvu Visual Basic aplikaciju!

Događaju ste pridružili akciju: kada se klikne komandno dugme, prikazuje se tekst. Pridruživanje akcija događajima je osnova svih Windows aplikacija.

Pisanje komentara u aplikaciji

Pošto je pred nama program koji radi, bilo bi dobro u okviru programskog koda objasniti šta radi. Na taj način nećete, ukoliko kasnije pogledate aplikaciju, doći u situaciju da se pitate kako radi. Zapravo, možda nećete biti zaduženi za održavanje programskog koda, tako da se može reći da je pisanje komentara dobar način da objasnite čemu služi programski kod. Čak i kada znate da ćete zauvek održavati programski kod, odnosite se prema sebi kao da ste stranac. Možda ćete se iznenaditi koliko je vremena potrebno da odgonetnete programski kod koji ste napisali nekoliko meseci ili nekoliko godina ranije.

Komentar u jednom programskom redu se zapisuje na sledeći način:

```
' A single-line comment
```

Kompajler ignoriše sve što se nalazi u istom programskom redu iza apostrofa (´) i ne uključuje u konačnu verziju aplikacije. Hajde sada da dokumentujemo našu Windows aplikaciju:

```
' Kada korisnik klikne komandno dugme u tekstualnom polju, prikazujemo tekst
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        TextBox1.Text = "hello, world"
    End Sub
```

U programskom jeziku Visual Basic se instrukcije zapisuju u jednom programskom redu. To znači da se iskaz mora nalaziti u jednom programskom redu. Pogledajte primer takvog iskaza:

```
        TextBox1.Text = "hello, world"
```

Prethodni programski red je jedan iskaz, jer se tumači kao dodeljivanje vrednosti promenljivoj pomoću dela programskog koda. Isti iskaz ne možete napisati na sledeći način:

```
        TextBox1.Text =
                "hello, world"
```

Kada se iskaz podeli u dva programska reda, Visual Basic kompajler ta dva reda vidi kao dva iskaza. Pošto ta dva iskaza nisu kompletna, prilikom kompajliranja će biti prijavljena greška.

Ukoliko jedan iskaz morate da upišete u dva reda, to morate da naznačite kompajleru, tako što ćete na kraj programskog reda koji nastavljate u sledećem programskom redu upisati znak za nastavljanje programskog reda – podvlaku (_), kao u sledećem primeru:

```
TextBox1.Text = _
        "hello, world"
```

Kretanje kroz korisničke kontrole rešenja

Kada pišete programski kod, osnovno sredstvo za kretanje je Solution Explorer – on predstavlja tri kontrole koje sadrže reference na rešenja i projekte. Solution Explorer možete zamisliti kao nadzornu tablu programera koja se koristi za fino podešavanje izvršavanja i pravljenja .NET aplikacije.

Savetujem Vam da istražite Solution Explorer. Desnim tasterom miša kliknite njegove razne elemente. Kontekstni meniji omogućavaju da brzo podesite neke od aspekata rešenja ili projekta. Međutim, ni u jednom okviru za dijalog nemojte kliknuti OK; kliknite Cancel da ne biste zapisali izmene koje napravite.

Desno od Solution Explorera vidite radnu oblast. U njoj pišete programski kod ili menjate izgled korisničkog interfejsa. U radnoj oblasti se prikazuje samo jedna vrsta informacija - programski kod, korisnički interfejs ili projekat. Kao što ste već videli, pošto u Solution Exploreru dva puta kliknete element `Form1.vb`, u radnoj oblasti se prikazuje formular koji odgovara datoteci `Form1.vb`.

Pošto ste naučili kako se koristi IDE, nastavićemo sa primerima. Sada ćete napraviti konzolnu aplikaciju.

Pravljenje konzolne aplikacije

Konzolna aplikacija je tekstualna aplikacija. To znači da se, umesto grafičkog korisničkog interfejsa (GUI), na komandnoj liniji prikazuje interfejs.

Konzole postoje veoma dugo, jer je to bio prvi način za interakciju sa računarom. One ne olakšavaju rad korisnika i zamaraju kada treba obaviti komplikovane operacije, ali za njih neki ljudi tvrde da predstavljaju sve što je potrebno za rad na računaru (više informacija o konzolama ćete pronaći na adresi `http://en.wikipedia.org/wiki/Command_line_interface`).

Pisanje u konzoli je moguće samo ako aplikacija koja se izvršava ima konzolu. Da biste u Windowsu otvorili konzolu, odaberite Start ⇨ Run i u okvir za dijalog unesite komandu cmd. Kada testirate konzolnu aplikaciju, Visual Basic automatski otvara konzolu.

U Visual Basic Expressu možete da napravite i kompajlirate konzolne aplikacije i da upravljate njima.

Dodavanje konzolne aplikacije u rešenje

Sada ćete napraviti konzolnu aplikaciju koja tekst „hello, world“ prikazuje u konzoli. Na sledeći način ćete novi projekat dodati u rešenje ThreeExamples:

1. Odaberite File ⇨ Add ⇨ New Project.
2. Odaberite lokaciju koju ste odabrali za aplikaciju `WindowsApplication`.
3. Odaberite Console Application i zadajte naziv `ConsoleApplication`.

U Solution Exploreru će biti prikazani novi projekat i rešenje. U radnoj oblasti se prikazuje programski kod.

Obratite pažnju koliko je konzolna aplikacija jednostavna. Sadrži jednu jednostavnu datoteku Module1.vb sa programskim kodom. U konzolnim aplikacijama obično ne postoji posebno grupisanje i ne postoje događaji.

Konzolna aplikacija koja Vas pozdravlja

Da bi konzolna aplikacije nešto „radila", u metod Main() morate da upišete programski kod, kao u sledećem primeru:

```
Module Module1

    Sub Main()
        Console.WriteLine("hello, world")
        Console.ReadKey()
    End Sub

End Module
```

Programski red prikazan masnim slovima ispisuje tekst „hello, world" u konzolu.

Ukoliko ste konzolnu aplikaciju pokušali da pokrenete tako što ste pritisnuli kombinaciju tastera Ctrl+F5, umesto konzolne aplikacije ste pokrenuli Windows aplikaciju (aplikaciju `WindowsApplication`). Hajde da to promenimo.

Zadavanje polaznog projekta

Da biste pokrenuli konzolnu aplikaciju, morate je zadati kao polazni (početni) projekat. Da li ste primetili da je aplikacija `WindowsApplication` u Solution Exploreru prikazana masnim slovima? To znači da je aplikacija `WindowsApplication` polazni (početni) projekat. Kada pokrenete ili debagirate aplikaciju, izvršava se ili debagira polazni (početni) projekat.

Da bi aplikacija `ConsoleApplication` postala polazni projekat, desnim tasterom miša kliknite projekat `ConsoleApplication` i u kontekstnom meniju odaberite Set As StartUp Project. Aplikacija `ConsoleApplication` će sada biti prikazana masnim slovima, što znači da je to polazni projekat rešenja ThreeExamples.

Pokretanje konzolnog projekta

Pošto je aplikacija `ConsoleApplication` postala polazni projekat, možete je pokrenuti tako što ćete pritisnuti kombinaciju tastera Ctrl+F5. Rezultat je:

```
hello, world
```

Izvršavanje konzolne aplikacije ne proizvodi prozor, kao što je bio slučaj sa Windows aplikacijom. Umesto toga se prilikom pokretanja aplikacije `ConsoleApplication` prikazuje komandni prompt. Izvršavanjem aplikacije `ConsoleApplication` se prikazuje tekst „hello, world". Prozor komandnog prompta ćete zatvoriti tako što ćete pritisnuti bilo koji taster. Visual Basic Express je automatski generisao programski kod pomoću kojeg se dobija tekst i izvršava akcija.

U opštem slučaju, mogućnosti konzolnih aplikacija su ograničene, ali takve aplikacije omogućavaju lako obavljanje specifičnih poslova. Pređimo sada na sledeći primer.

Pravljenje biblioteke klasa

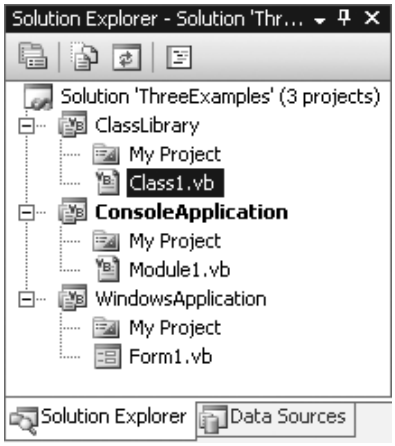
Treći primer u ovom poglavlju nije .NET aplikacija; to je programski kod koji se može više puta koristiti, a koji se obično naziva *biblioteka klasa* (engl. *class library*). Windows aplikacije i konzolne aplikacije su programi koje možete izvršiti sa komandnog prompta ili iz Windows Explorera. Korisnik ne može pokrenuti i izvršiti biblioteku klasa. Biblioteka klasa se koristi u Windows aplikaciji ili konzolnoj aplikaciji. U nju se zapisuje programski kod koji se može koristiti u više aplikacija.

Dodavanje biblioteke klasa u rešenje

Sada ćete napraviti biblioteku klasa koja se može koristiti u Windows aplikaciji ili konzolnoj aplikaciji. Na sledeći način ćete novi projekat dodati u rešenje `ThreeExamples`:

1. U Solution Exploreru desnim tasterom miša kliknite naziv rešenja; dakle, kliknite `ThreeExamples`.
2. Odaberite Add ➪ New Project.
3. Odaberite Class Library i zadajate naziv `ClassLibrary`.

Rezultujući projekat rešenja treba da izgleda kao na slici 1-8.



SLIKA 1-8 *Struktura rešenja koje sadrži tri projekta*

Projekat `ClassLibrary` ima samo datoteku `Class1.vb`, u kojoj nema programskog koda.

Premeštanje funkcionalnosti

Sada ćete programski kod pomoću kojeg se ispisuje tekst „hello, world" iz aplikacije `ConsoleApplication` prebaciti u biblioteku `ClassLibrary`. Sledeći programski kod unesite u datoteku `Class1.vb` (programski kod koji je prikazan masnim slovima):

```
Public Class Class1
    Public Shared Sub HelloWorld()
      Console.WriteLine("hello, world")
    End Sub
End Class
```

U programskom kodu se nalazi metod `HelloWorld()`, koji, kada se pozove, ispisuje tekst „hello, world". Kao što sam već istakao, metod je skup instrukcija pomoću kojih se obavlja posao. Metode ću detaljnije objasniti u Poglavlju 2.

Da bi aplikacije mogle da koriste programski kod koji se nalazi u biblioteci klasa, morate omogućiti da projekti „znaju" jedan za drugog. To ćete postići korišćenjem veza (referenci).

Definisanje veza

Da bismo u jednom projektu mogli da koristimo definicije iz drugog projekta, moramo definisati *vezu* (referencu). Svrha veza (referenci) jeste da se ukaže na to da se u projektu mogu koristiti delovi programskog koda koji se nalaze na drugom mestu.

NAPOMENA

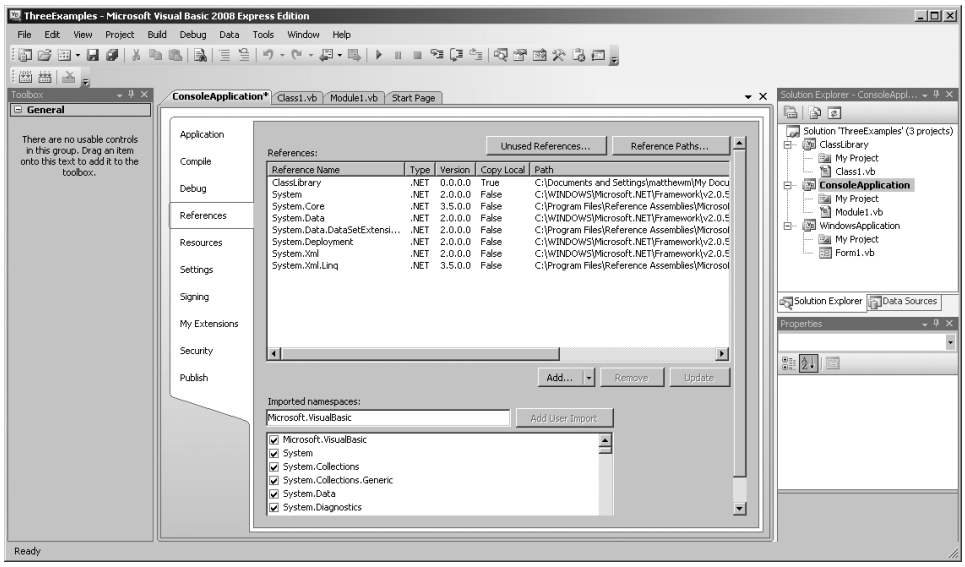
U projektu se mogu koristiti delovi programskog koda koji su deklarisani tako da su tipa public (javni delovi programskog koda). Javni delovi programskog koda, ili ono što Visual Basic programeri nazivaju javni opseg (engl. public scope), predstavljaju delove programskog koda koji se deklarišu pomoći rezervisane reči `Public`. U ovoj knjizi ćete naučiti šta su javni i drugi opsezi.

Da biste u aplikaciji `ConsoleApplication` mogli da koristite programski kod koji se nalazi u biblioteci `ClassLibrary`, morate na sledeći način napraviti vezu (referencu):

1. U Solution Exploreru kliknite `ConsoleApplication`.
2. Kliknite desnim tasterom miša i u kontekstnom meniju odaberite Add Reference.
3. Kliknite karticu Projects.
4. Odaberite `ClassLibrary`, pa kliknite OK. Na ovaj način ćete biblioteku `ClassLibrary` dodati u veze (reference) aplikacije `ConsoleApplication`.

Pošto pridružite vezu (referencu), aplikacija `ConsoleApplication` može da koristi programski kod koji se nalazi u biblioteci `ClassLibrary`.

U podešavanjima projekta ćete pronaći veze (reference) koje su definisane za aplikaciju ili biblioteku klasa. Podešavanja projekta ćete prikazati tako što ćete u Solution Exploreru desnim tasterom miša kliknuti naziv projekta (u ovom slučaju projekta `ConsoleApplication`) i u kontekstnom meniju odabrati Properties. U prozoru Properties odaberite karticu References (pogledajte sliku 1-9).



SLIKA 1-9 *Veze (reference) koje su definisane za Visual Basic projekat*

Pozivanje programskog koda iz biblioteke klasa

Sada treba da izmenite aplikaciju `ConsoleApplication` tako da poziva funkciju koja se nalazi u biblioteci `ClassLibrary`. Datoteku `Module1.vb` izmenite na sledeći način:

```
Module Module1

    Sub Main()
      Console.WriteLine("hello, world")
      ClassLibrary.Class1.HelloWorld()
      Console.ReadKey()
    End Sub

End Module
```

Pokrenite aplikaciju `ConsoleApplication`, tako što ćete pritisnuti kombinaciju tastera Ctrl+F5. Trebalo bi da se prikaže prozor sa komandnim promptom u kojem je dva puta ispisan tekst „hello, world“. Prvi tekst „hello, world“ se dobija izvršavanjem programskog reda `Console.WriteLine("hello, world")`. Pozivanjem funkcije `ClassLibrary.Class1.HelloWorld()` se drugi put ispisuje tekst “hello, world”.

SKRAĆENO KORIŠĆENJE VEZA (REFERENCI)

`ClassLibrary.Class1.HelloWorld()` je duži način za korišćenje veze (reference). Da smo hteli na isti način da pozovemo metod `Console.WriteLine()`, onda bismo napisali `System.Console.WriteLine()`, jer je metod `Console.WriteLine()` definisan korišćenjem veze (reference) System. Međutim, Visual Basic Express automatski definiše vezu (referencu) System, što znači da ne moramo koristiti duži način zapisivanja veze (reference).

Da bismo mogli da koristimo skraćeni način pozivanja metoda iz biblioteke `ClassLibrary`, moramo na početku datoteke `Module1.vb` aplikacije `ConsoleApplication` da napišemo programski red Imports i da promenimo pozivanje metoda `HelloWorld()` koji se nalazi u klasi `Class1`:

```
    Imports ClassLibrary
    Module Module1

        Sub Main()
            Console.WriteLine("hello, world")
            Class1.HelloWorld()
            Console.ReadKey()
        End Sub

End Module
```

Međutim, skraćeni način zapisivanja ima nedostatak. Šta bi se dogodilo kada bismo napravili veliki broj veza (referenci) koje sadrže klasu `Class1`? U tom slučaju Visual Basic Express ne bi znao koju klasu da upotrebi, osim kada bismo upotrebili duži način zapisivanja veza (referenci). Doduše, malo je verovatno da će postojati više klasa `Class1`, ali treba istaći da se čak i smisleni nazivi mogu ponoviti u skupu veza (referenci). A ako se nečiji programski kod koristi kao veza (referenca), verovatnoća da postoje duplirani nazivi je veća. Prema tome, u ovom slučaju je bolje da koristite duži način zapisivanja veza (referenci).

Korišćenje promenljivih i konstanti

Jedan od osnovnih koncepata u Visual Basic programu jeste korišćenje promenljivih. Promenljivu zamislite kao deo memorije u koji možete zapisati podatak koji ćete kasnije koristiti. Na taj način veoma lako možete koristiti podatak u okviru programa.

U projektu `ClassLibrary` bi bilo mnogo bolje kada bismo mogli da definišemo poruku koja bi se prikazivala na početku metoda. Onda bismo, ukoliko treba da promenimo poruku, mogli to mnogo lakše da uradimo. Dakle, ukoliko odlučimo da pre pozivanja metoda `Console.WriteLine()` napišemo programski kod, moramo da pronađemo poruku koju želimo da izmenimo. U ovakvom slučaju treba da koristimo promenljivu, jer možemo definisati podatak (poruku koju treba prikazati) i kasnije ga koristiti u programu.

```
Public Class Class1
    Public Shared Sub HelloWorld()
        Dim message As String = "hello, world"
        Console.WriteLine(message)
    End Sub
End Class
```

U prethodnom programskom kodu smo definisali promenljivu `message` tipa `String` (tip `String` je duži tekst). Kada želimo da sadržaj promenljive upotrebimo u programskom kodu, treba da pozovemo promenljivu message. U primeru smo sadržaj promenljive smestili u poziv metoda `Console.WriteLine()` koji se izvršava kao prethodnom programskom kodu. Međutim, ovoga puta poruka se prikazuje pomoću posebnog iskaza.

Ovo je od velike pomoći, ali to nije sve što možemo da uradimo sa promenljivim. Za promenljive se vezuje pojam *opsega* (engl. *scope*). Opseg promenljive `message` je metod, što znači da se može koristiti samo u metodu u kojem je definisana. Proučite sledeći programski kod:

```
Public Shared Sub HelloWorld()
    Dim message As String = "hello, world"
    Console.WriteLine(message)
End Sub

Public Shared Sub DisplayeMessageText()
    Console.WriteLine("hello, world")
    Console.WriteLine(message)
End Sub
```

Pomoću metoda `DisplayMessageText()` se prikazuju dva reda teksta koja nas obaveštavaju kako treba da izgleda tekst poruke. Međutim, prethodni programski kod neće biti kompajliran, jer kompajler zna da se promenljiva `message` ne može koristiti u metodu `DisplayMessageText()` pošto je izvan opsega.

Da bismo rešili problem, promenljivoj `message` treba da dodelimo opseg na nivou klase, tako što ćemo je premestiti na početak definicije klase (pošto se koristi u metodima tipa `Shared`, promenljiva takođe treba da bude tipa `Shared`):

```
Public Class Class1
    Shared Dim message As String = "hello, world"
    Public Shared Sub HelloWorld()
        Console.WriteLine(message)
    End Sub

    Public Shared Sub DisplayeMessageText()
        Console.WriteLine("hello, world")
        Console.WriteLine(message)
    End Sub
End Class
```

Sada se promenljiva `message` može koristiti u svim metodima klase `Class1`. U ovoj knjizi ćete još mnogo štošta naučiti o opsezima na nivou metoda i klase i o rezervisanim rečima `Public` i `Shared`.

Korišćenje promenljive u metodima klase se može pokazati korisnim, iako nekada nije uputno - razlog jer činjenica da metodi mogu, prilikom obavljanja poslova, promeniti vrednost promenljive, što može dovesti do neočekivanih rezultata. Vrednost možemo učiniti nepromenljivom ukoliko umesto promenljive koristimo konstantu. Konstanta se definiše pomoću rezervisane reči const:

```
Public Class Class1
    Const MESSAGE As String = "hello, world"
    Public Shared Sub HelloWorld()
        Console.WriteLine(MESSAGE)
    End Sub

    Public Shared Sub DisplayeMessageText()
        Console.WriteLine("hello, world")
```

```
        Console.WriteLine(MESSAGE)
    End Sub
End Class
```

Constant names are usually all uppercase. The contents of a constant cannot be changed at any point. The following would not compile, for instance.

```
Public Class Class1
    Const MESSAGE As String = "hello, world"

    Public Shared Sub DisplayeMessageText()
        MESSAGE = "another text that cannot be assigned"
        Console.WriteLine(MESSAGE)
    End Sub
End Class
```

Pošto ste videli primere iz ovog poglavlja, hajde da vidimo kako se Visual Basic programski kod u Visual Basic Expressu pretvara u program koji se može koristiti u operativnim sistemima kao što je Windows.

Razumevanje rada okruženja .NET Framework

Kada pišete Visual Basic programski kod, zapisujete instrukcije koje će program izvršiti. Instrukcije su definisane pomoću programskog jezika Visual Basic, koji je za Vas koristan, ali ne i za računar. Računar ne može da „razume" delove teksta, već nule i jedinice. Da bi se instrukcije unele u računar, programeri su napravili mehanizam instrukcija višeg nivoa koji instrukcije pretvara u nešto što računar može da razume. Pomoćni program koji obavlja taj posao se naziva *kompajler* (engl. *compiler*).

.NET, za razliku od tradicionalni programskih jezika, kao što su C++ ili C, generiše binarni međujezik, koji se naziva opšti međujezik (engl. Common Intermediate Language - CIL). Potom, okruženje .NET Framework pretvara CIL u binarne instrukcije koje zahteva procesor računara.

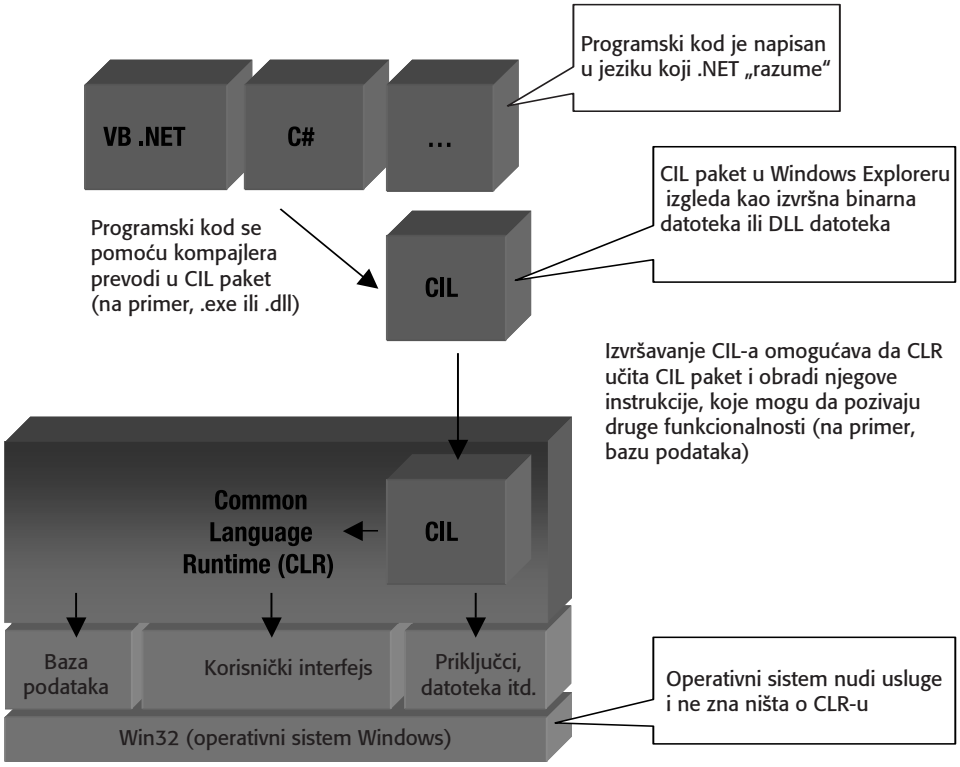
Možda mislite da je prevođenje programskog koda u opšti međujezik neefikasno rešenje, ali je to, zapravo, dobar pristup. Poslužiću se analogijom. Postoje psi koji brzo uče komande i psi kojima je potrebno više vremena da ih nauče. Na primer, nemački ovčari brzo uče i ne moraju mnogo puta da ponavljaju lekciju. Sa druge strane, sa bulmastifima morate biti strpljivi, jer su tvrdoglavi. Zamislite da ste instruktor koji je osmislio uputstva na osnovu kojih se dresira bulmastif. Ukoliko se ista uputstva upotrebe za nemačkog ovčara, psu će biti dosadno i verovatno neće naučiti ono što Vi želite.

Problem sa uputsvima je što su osmišljena za jednog psa. Ukoliko želite da dresirate oba psa, potrebna su dva skupa uputstava. Problem se rešava pravljenjem opštih uputstava sa dodatnim smernicama (na primer, „Ukoliko je pas tvrdoglav, ponovi vežbu.").

Ukoliko ovo prevedemo na jezik računara, dva skupa uputstava su namenjena različitim procesorima ili procesorima koji se koriste u posebnim situacijama. Na primer, postoje serverski računari i klijentski računari. Svaki tip računara ima drugačije zahteve. Serverski računar mora što brže da obradi podatke, dok klijentski računar mora što brže da ih prikaže. Za svaku namenu postoje kompajleri, ali nije efikasno da programer pomoću različitih kompajlera pravi distribucije ili

zadaje podešavanja. Rešenje je pravljenje opšteg skupa instrukcija uz koje su pridružene napomene. Okruženje .NET Framework potom izvršava te instrukcije koristeći napomene.

Okruženje .NET Framework kompajlira programski kod u instrukcije (CIL), koje se, potom, pretvaraju u instrukcije za procesor, pri čemu se koriste napomene koje su ugrađene u .NET Framework. .NET arhitektura je prikazana na slici 1-10.



SLIKA 1-10 *.NET arhitektura*

Na slici 1-10 vidite da je Visual Basic Express zadužen za pretvaranje Visual Basic programskog koda u CIL paket. CIL paket je binarna datoteka koja, kada se izvršava, zahteva opšti izvršni jezik (CLR). Ukoliko CLR nije instaliran, ne možete izvršavati CIL paket. Kada ste instalirali Visual Basic Express, automatski ste kao poseban paket instalirali CLR. Visual Basic Express je aplikacija koja omogućava da pravite programe za CLR, ali je i aplikacija koja koristi CLR.

CLR može instrukcije iz CIL paketa da prevede u nešto što operativni sistem i procesor mogu da „razumeju". Ukoliko pogledate sintaksu .NET programskih jezika, kao što su Visual Basic, C# i Eiffel.NET, videćete da oni nisu slični. Ipak, CLR može da obradi CIL pakete koje generišu ti programski jezici, jer .NET kompajler, bez obzira na programski jezik, generiše opšti skup CLR instrukcija.

Kada koristite okruženje .NET Framework, programski kod pišete za CLR, a sve što radite CLR mora da „razume". Uopšteno govoreći, to ne predstavlja problem ukoliko programski kod pišete koristeći programski jezik Visual Basic. Prednosti pisanja programskog koda za CLR su sledeće:

Memorija i sakupljanje „otpadaka": Programi koriste resurse kao što su memorija, datoteke i tako dalje. U tradicionalnim programskim jezicima, kao što su C i C++, očekuje se da otvorite i zatvorite datoteku i da rezervišete i oslobodite memoriju. U .NET-u ne morate da brinete o zatvaranju datoteka ili oslobađanju memorije. CLR „zna" kada se datoteka ili memorija ne koriste, pa automatski zatvara datoteku, odnosno oslobađa memoriju.

NAPOMENA

Neki programeri smatraju da CLR propagira aljkavo programiranje, pošto programer za sobom ne mora da „posprema". Međutim, u praksi se pokazalo da je prilikom programiranja svake složene aplikacije pronalaženje memorije koju treba osloboditi traćenje vremena i resursa.

Podešavanje optimizacije: Neki programi moraju da obrađuju velike količine podataka (na primer, podatke koji se nalaze u bazi podataka) ili da prikazuju složeni korisnički interfejs. Performasne takvih aplikacija zavise od različitih delova programskog koda. CLR može da optimizuje CIL paket i donese odluku tako da ga izvrši što brže i što efikasnije.

Sistem opštih tipova (Common Type System – CTS): String u programskom jeziku Visual Basic je isto što i string u programskom jeziku C#. To obezbeđuje da ne dođe do pogrešnog tumačenja podataka kada CIL paket koji je generisao Visual Basic komunicira sa CIL paketom koji je generisao programski jezik C#.

Bezbedan programski kod: Kada pravite programe koji komuniciraju sa datotekama ili memorijom, postoji mogućnost da programska greška stvori bezbednosne probleme. Hakeri će iskoristiti bezbednosni problem da bi mogli da izvršavaju svoje programe i eventualno prouzrokuju finansijsku katastrofu. CLR ne može da ispravi greške koje postoje u aplikacijama, ali može da zaustavi program koji izaziva grešku nepravilnim korišćenjem datoteke ili memorije.

Prednost CLR-a je što programerima omogućava da pažnju usmere na probleme u vezi sa „radom" aplikacije, jer ne moraju da brinu o problemima koji se odnose na infrastrukturu. CLR omogućava da pažnju usmerimo na programski kod aplikacije pomoću kojeg se čita i obrađuje sadržaj datoteke. Kada CLR ne bi postojao, onda bismo morali da napišemo programski kod pomoću kojeg se koristi sadržaj datoteka i programski kod pomoću kojeg se datoteka otvara, čita i zatvara.

Zapamtite sledeće

Ovo je poglavje u kojem ste u IDE-u počeli da koristite programski jezik Visual Basic. Evo šta morate da zapamtite:

- Postoje tri osnovna tipa Visual Basic programa: Windows aplikacije, konzolne aplikacije i biblioteke klasa.
- Windows aplikacija ima korisnički interfejs i „radi" kao i svaka druga Windows aplikacija (na primer, aplikacije Notepad i Calculator). U Windows aplikacijama povezujete događaje i akcije.

- Konzolna aplikacija je jednostavnija od Windows aplikacije i nema događaje. Koristi se za obradu podataka. Konzolna aplikacija generiše i prihvata podatke sa komandnog prompta.
- IDE koristite za pisanje programskog koda, debagiranje i izvršavanje aplikacije.
- IDE, između ostalog, koristite za uređivanje programskog koda u projekte i rešenja.
- Tastaturne prečice u IDE-u olakšavaju obavljanje operacija koje se često ponavljaju. Na primer, u Visual Basic Expressu tastaturna prečica Ctrl+S služi da na disk zapišemo sve što smo radili, a tastaturna prečica Ctrl+F5 da bez debagiranja pokrenemo aplikaciju.
- U Visual Basic Express projektima postoje prazne datoteke i posebna grupisanja. Kada koristite grupisanja, morate razumeti kako ona funkcionišu i izmeniti samo one datoteke koje ste namerili da izmenite.

Šta treba da uradite?

Evo pitanja koja se odnose na ono što ste naučili u ovom poglavlju. Odgovorima na pitanja ćete lakše početi da pravite projekte u IDE-u.

NAPOMENA

Odgovore/rešenja na pitanja/vežbe na kraju svakog poglavlja možete preuzeti zajedno sa programskim kodom sa web sajta izdavačke kuće „Apress" iz odeljka Source Code/Download (http://www.apress.com). Osim toga, možete sa mnom kontaktirati na adresi christiangrossŽgmail.com.

1. Rešenja i projekti se u IDE-u koriste za klasifikovanje međusobno povezanih delova programskog koda. Tu vezu sam opisao pomoću analogije sa automobilima i delovima automobila. Da li biste ikada napravili rešenje koje sadrži programski kod koji nije međusobno povezan? Na primer, da li biste napravili rešenje za avion koje sadrži delove za automobil?
2. Projekti se zasnivaju na šablonima koje je napravio „Microsoft". Da li možete da zamislite situaciju za koju biste napravili šablon i dodali ga u Visual Basic Express?
3. Svaki deo stabla u Solution Exploreru predstavlja jedan element (na primer, datoteku, korisnički interfejs i tako dalje). Ukoliko biste dva puta kliknuli .vb datoteku, upravljali biste Visual Basic datotekom koja sadrži Visual Basic programski kod. Da li jedna Visual Basic datoteka treba da se odnosi na jednu Visual Basic klasu ili imenovani prostor? Ako ne treba, da li biste Visual Basic programski kod organizovali u Visual Basic datoteke?
4. Naučili ste kako .NET aplikacija generiše izvršnu datoteku. Pretpostavimo da ste napravljenu aplikaciju pokrenuli na drugom Windows računaru. Da li će se aplikacija izvršavati? Pretpostavimo da ste izvršnu datoteku prebacili na Macintosh OS X ili Linux računar. Da li možete da pokrenete aplikaciju? Zašto je možete koristiti, odnosno zašto je ne možete koristiti?

5. Ne dopada Vam se naziv elementa `TextBox1` i želite da ga promenite u `TxtOutput`. Kako ćete ga promeniti?
6. Biblioteka `ClassLibrary` je napravljena tako da se pretpostavlja da će njen programski kod pozivati konzolna aplikacija. Da li treba pretpostaviti da će se programski kod biblioteke koristiti u određenom tipu aplikacije? Zašto je prethodna tvrdnja tačna, odnosno, zašto nije tačna?