Deo I

Uvod u razvoj Windows 8 aplikacija

POGLAVLJE 1: Kratak istorijat razvoja Windows aplikacija

POGLAVLJE 2: Korišćenje Windows 8 operativnog sistema

POGLAVLJE 3: Windows 8 arhitektura sa stanovišta programera

POGLAVLJE 4: Upoznavanje razvojnog okruženja

1

Kratak istorijat razvoja Windows aplikacija

U OVOM POGLAVLJU ĆETE NAUČITI:

- **Nastanak i evolucija Windows operativnog sistema u prethodnih 27 godina**
- **Različiti alati i tehnologije koji su karakterisali Windows razvoj**
- **Promena načina razmišljanja u Windows 8 operativnom sistemu i korišćenje nove paradigme prilikom razvoja aplikacija**
- **Upoznavanje Windows 8 stila aplikacija**

Windows 8 predstavlja najveću promenu u celokupnom razvoju ovog operativnog sistema. To nije samo nova verzija, kojom se proširuju stare funkcije i dodaju određene zahtevane i popularne funkcije. Kako ističu nadležni u kompaniji „Microsoft“, sam operativni sistem je „zamišljen od početka“. Bez poznavanja istorijata Windows 8 operativnog sistema, veoma je teško razumeti promene koje se uvode.

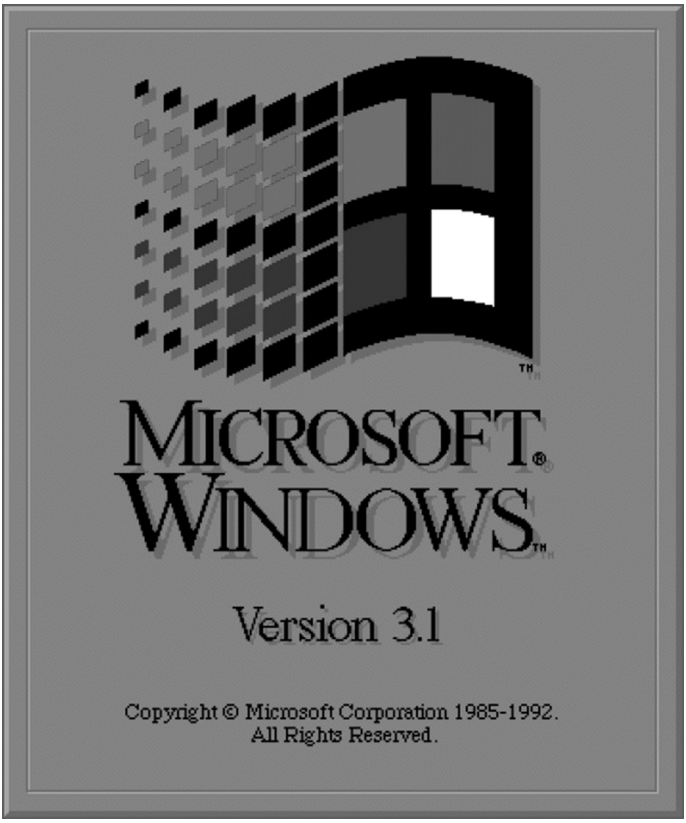
U ovom poglavlju ćete saznati kako je ovaj operativni sistem evoluirano u prethodnih 27 godina, a nakon toga ćete naučiti i nešto o razvojnim tehnologijama i alatima koji su se razvijali zajedno sa Windows operativnim sistemom.

ISTORIJA WINDOWS OPERATIVNOG SISTEMA

Kada je kompanija „Microsoft“ objavila prvu verziju Windows operativnog sistema, 20. novembra 1985. godine, on je bio zasnovan na korišćenju grafičkog korisničkog interfejsa (GUI). „Microsoft“ je kreirao Windows kao dodatnu komponentu MS-DOS operativnog sistema, a sam Windows je u potpunosti promenio izgled personalnih računara. Prva verzija Windows operativnog sistema je koristila veoma jednostavnu grafiku i bila je više dodatni front end za MS-DOS, a ne pravi operativni sistem.

Od Windows 3.1 verzije do korišćenja 32 bita

Prošlo je skoro sedam godina pre nego što je u martu 1992. godine objavljen Windows 3.1. Ovaj 16-bitni operativni sistem je omogućavao multitasking – u okruženju u kome korisnici nisu imali priliku da ga koriste. Nova verzija Windows operativnog sistema je imala drajvere virtuelnih uređaja, koji su mogli biti deljeni između DOS aplikacija. Zbog svog zaštićenog režima, Windows 3.1 je imao mogućnost adresiranja nekoliko megabajta memorije – podrazumevani režim koji je koristila 8086 familija procesora (CPU) omogućavao je samo 640 KB – bez potrebe za korišćenjem softvera za upravljanje memorijom. Oni koji su koristili računare u to doba verovatno se sećaju uvodnog ekrana ovog operativnog sistema, koji je prikazan na slici 1-1.



SLIKA 1-1 Uvodni ekran Windows 3.1 operativnog sistema

NAPOMENA Multitasking je mogućnost operativnog sistema da istovremeno izvršava veći broj zadataka (koji se nazivaju i procesi). Ovi zadaci dele resurse računara, kao što su CPU i memorija. Operativni sistem prelazi sa jednog na drugi zadatak prilikom njihovog izvršavanja. Svaki zadatak se izvršava u određenom malom vremenskom intervalu, a, zbog brze promene, izgleda kao da se izvršavaju simultano.

Zaštićeni režim je režim funkcionisanja x86-kompatibilnih CPU procesora, koji omogućava specijalne funkcije, kao što su upravljanje virtuelnom memorijom i multitasking.

Objavljen u avgustu 1995. godine, Windows 95 je bio 32-bitni operativni sistem koji je podržavao *preemptive multitasking* – drugim rečima, operativni sistem je imao mogućnost prekidanja izvršavanja zadatka bez bilo kakvog aktivnog učešća samog zadatka. Windows 95 više nije bio dodatak za MS-DOS, već je postao potpuno nezavisan operativni sistem (činjenica o kojoj se već dugo raspravlja). Nakon ove, usledilo je još nekoliko verzija Windows operativnog sistema (konkretno, Windows 98 i Windows Me), pre nego što je u oktobru 2001. godine objavljen Windows XP.

Windows XP i Windows Vista

Windows XP je postao najpopularnija verzija Windows operativnog sistema (njegov čuveni logo je prikazan na slici 1-2). Veoma čudno, ali njegov uspeh (sa stanovišta broja instalacija) je samo delimično posledica novih iskustava (XP predstavlja skraćenicu od eXPerience) koja su korisnici doživeli prilikom njegovog korišćenja. Slično tome, inovacije, kao što su GDI+ grafički podsistem, brza promena korisnika, ClearType renderovanje fontova, 64-bitna podrška i još puno štošta, samo su delimično uticale na uspeh ove verzije Windows operativnog sistema. Zapravo, primarni razlog uspeha XP operativnog sistema je bila nepopularnost Windows Vista verzije – njegovog sledbenika.

SLIKA 1-2 Windows XP logo

Objavljena u novembru 2006. godine, Windows Vista je ponudila sasvim nov dizajn, kao i značajno poboljšanje bezbednosti – nasuprot XP verziji, koja je zahtevala tri servisna paketa da bi bili uklonjeni svi bezbednosni propusti i problemi. Iako je samo ovo trebalo da bude dovoljno da bi se postigla veća popularnost u odnosu na prethodnu verziju, Vista je zahtevala značajno jači hardver da bi se te funkcije na odgovarajući način izvršavale. Najveći broj kompanija koje su utrošile značajna sredstva u IT budžetu na stabilizovanje XP operativnog sistema – nakon pojave Windows XP Service Pack 3 servisnog dodatka (SP3) – jednostavno nisu smatrale da je razumno preći na Vista operativni sistem. Vista je ubrzo postala najkraće korišćeni operativni sistem u Windows familiji.

Windows 7 je bacio u drugi plan fijasko Vista operativnog sistema

Kao što je Steven Sinofsky (predsednik Windows odeljenja u kompaniji „Microsoft“) nekoliko puta priznao, „Microsoft“ je naučio lekciju i pre nego što je pristupio razvoju Windows 7 operativnog sistema koji je objavljen u julu 2009. godine, dve godine i osam meseci nakon Vista operativnog sistema. Windows 7 sadrži značajna poboljšanja performansi u odnosu na Windows XP i Vistu, uključujući vreme startovanja, vreme prekidanja rada, definisanje rasporeda zadataka na CPU jedinicama sa više jezgara (odnosno, na CPU jedinicama sa više procesora), pretraživanje i još mnogo štošta. Korisničko iskustvo je poboljšano, a u Windows 7 operativnom sistemu uvedene su liste za brzi pristup, Aero Pack, Aero Snap i brojni jednostavni dodaci koji poboljšavaju pristupanje prozorima aplikacija i njihovo organizovanje na ekranu.

Windows 7 je definitivno omogućio da se zaboravi neuspeh Vista operativnog sistema. Za Windows tim bi bilo mnogo jednostavnije da je nastavio u pravcu koji je definisan Windows 7 operativnim sistemom, ali je pred sebe postavio najsloženiji mogući zadatak.

Promena paradigme u Windows 8 operativnom sistemu

Iako je Windows operativni sistem nastao u periodu kada su personalni računari postali deo svakodnevnog života ljudi, dugo godina je operativni sistem razvijan uz prevashodno razmatranje poslovnog okruženja i zaposlenih koji koriste određene informacije. Najveći broj funkcija definisan je u operativnom sistemu, tako da korisnici žele (ili su primorani) da ih izvršavaju na način koji je definisan u samom sistemu. Korisnici nisu mogli da izvršavaju bilo koju akciju ukoliko nisu poznavali određene koncepte, kao što su datoteke, direktorijumi, bezbednosne grupe, privilegije, deljeni sadržaj, registar i slično. Ovaj pristup je podrazumevao da dizajneri predvide kako će korisnici interagovati sa sistemom.

NAPOMENA Zbog aplikacija koje troše raspoložive resurse, Windows operativni sistem mora da se periodično održava uklanjanjem nepotrebnih programa, defragmentacijom diska, proverama postojanja virusa, instaliranjem servisnih dodataka i drugim akcijama. Iako svaka nova verzija unosi značajna poboljšanja pojednostavljivanjem ili automatizovanjem dela prethodnih aktivnosti, određene aktivnosti korisnici moraju i dalje samostalno da obavljaju.

„Microsoft“ je preduzimao prvi korak ka korisnicima

Proizvodi kompanije „Apple“ orijentisani ka potrošnji, kao što su iPhone iPad, pokazali su svetu da postoji još jedan način za interekciju sa računarskim softverom na intuitivan način – bez potrebe za posedovanjem znanja šta su datoteka, direktorijum, sistemski registar ili instalaciona procedura za aplikacije. Kompanija „Microsoft“ dugo nije ozbiljno shvatala ovakav pristup, ali su je grafikoni prodaje na tržištu naterali da se usmeri na uređaje iz domena potrošačke elektronike i njima odgovarajuće operativne sisteme.

Prva ozbiljna promena koju je izvršio „Microsoft“ dogodila se sredinom februara 2010. godine na Mobile World kongresu (u Barseloni, Španija). Tada je ova kompanija prvi put prikazala javnosti Windows Phone 7, koji je funkcionisao na sasvim drugačiji način sa stanovišta korisnika. Vizuelni dizajn, jednostavnost, kao i prost korisnički interfejs, sa dobro iskorišćenim animacijama za naglašavanje, omogućio je da se ovaj uređaj veoma intuitivno koristi. Tržište je prihvatilo ovakav

pristup, a sada – više od godinu dana nakon što je objavljen Windows Phone 7.5 "Mango" – kompanija „Microsoft" je postala treća po zastupljenosti na tržištu mobilnih operativnih sistema i sve se više približava iOS operativnom sistemu kompanije „Apple" i Android operativnom sistemu kompanije „Google".

NAPOMENA Familija Windows operativnih sistema je već imala određena izdanja koja su namenjena posebnim uređajima (Windows Embedded) i mobilnim telefonima (Windows CE i Windows Mobile), koji su bili raspoloživi pre Windows Phone 7 verzije.

Windows 8 stupa „na scenu"

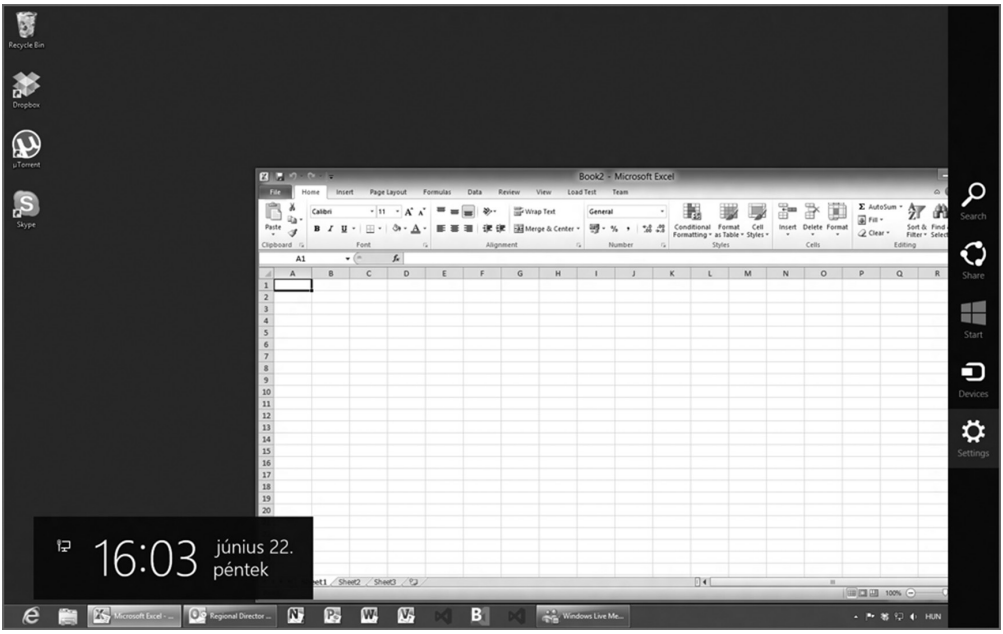
Ovaj pristup orijentisan ka potrošačima, primenjen u Windows Phone 7 operativnom sistemu, postao je osnova za razvoj Windows 8 operativnog sistema. Nakon što startujete operativni sistem (vreme podizanja sistema je značajno skraćeno u odnosu na ono u Windowsu 7), novi Windows 8 Start ekran ne podseća na prethodnu radnu površinu sa linijom poslova u donjem delu. Umesto toga, prikazuje se skup „pločica", a svaka od njih predstavlja određenu aplikaciju, kao što je prikazano na slici 1-3.

SLIKA 1-3 Windows 8 početni ekran

Novi Start ekran je jasna poruka korisnicima. Windows nije samo operativni sistem za one koji se bave obradom podataka i za redovne korisnike računara, već i za potrošačke uređaje dizajnirane tako da se komande mogu koristiti dodirivanjem površine ekrana, tablet računare i slate računare. Površina Start ekrana je veoma intuitivna, a većina korisnika može odmah da počne da koristi računare, bez ikakvih dodatnih instrukcija. Korisnici koji imaju iskustva sa inteligentnim telefonima i tabet računarima i njihovim ekranima osetljivim na dodir smatraju Start ekran veoma jed-

nostavnim za korišćenje, a startovanje aplikacija, zumiranje sadržaja i primena određenih poteza u cilju izvršavanja komandi su slični onima koje su već prethodno koristili.

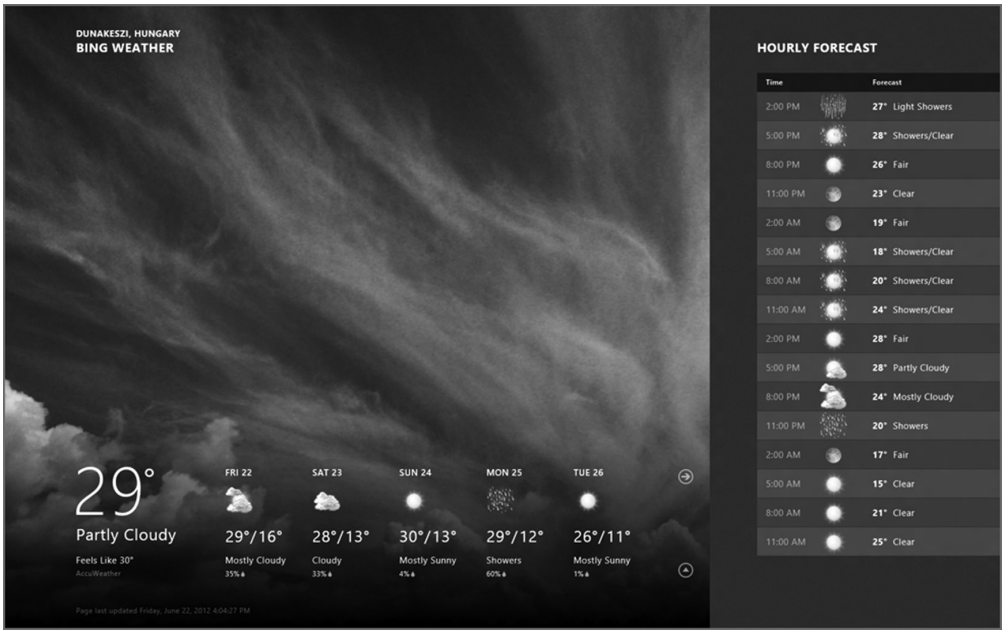
Oni korisnici koji koriste Windows operativni sistem da bi startovali poslovne aplikacije (kao što su Word, Excel ili PowerPoint) ili aplikaciju koja koristi specifičan korisnički interfejs namenjen poslovnim aplikacijama mogu da zaključe da je promena Windows korisničkog interfejsa neodgovarajuća. Dobro, Windows je projektovan da bi u potpunosti bio kompatibilan sa postojećim aplikacijama, tako da sadrži i desktop režim. Ukoliko startujete aplikaciju koja je kreirana za bilo koju od prethodnih verzija Windows operativnog sistema (ili jednostavno ne sadrži korisnički interfejs u Windows 8 stilu), aplikacija će se izvršavati u dobro poznatom desktop okruženju. Na primer, kada se startuje Excel, on se izvršava u desktop režimu, kao što je prikazano na slici 1-4.



SLIKA 1-4 Excel se izvršava u desktop režimu Windows 8 operativnog sistema.

Ovo je drugi lik Windows 8 operativnog sistema, poznat svima koji su prethodno koristili Windows operativne sisteme. Ukoliko niste videli Start meni i statusni indikator koji prikazuje trenutno vreme i datum, pomislićete da koristite Windows 7 operativni sistem.

Novi Windows 8 Start ekran nije samo jednostavan dodatak Windows 7 operativnom sistemu. Na njemu postoji potpuno novi svet aplikacija orijentisanih ka korisnicima, koje se nazivaju *Windows 8 style aplikacije*. Umesto da koriste radnu površinu koja sadrži veliki broj ikona i pravougaonih prozora pojedinačnih aplikacija, korisnici vide samo jednu aplikaciju u jednom trenutku, koja zauzima celokupnu površinu ekrana. Ne postoje naziv prozora, taster za zatvaranje, ivica kojom se može menjati veličina prozora ili bilo koji drugi element (nazvan chrome u terminologiji kojom se opisuje korisnički interfejs) koji bi odvlačio pažnju korisnika od same aplikacije. Weather aplikacija, prikazana na slici 1-5, tipičan je primer primene ovog novog stila.



SLIKA 1-5 Weather aplikacija u Windows 8 operativnom sistemu

U ovoj knjizi ćete naučiti nešto više o ovoj novoj paradigmi koja se odnosi na korisnički interfejs. Posebno je značajno da ćete dobiti instrukcije kako da razvijate aplikacije zasnovane na ovom novom stilu. Pre nego što detaljnije razmotrimo sve razvojne aspekte Windows 8 aplikacija, neophodno je da saznate i istorijat razvoja Windows aplikacija.

ISTORIJAT API INTERFEJSA I ALATA

Priča o Windows operativnom sistemu nije kompletna bez opisivanja aplikacija koje se izvršavaju na ovoj platformi, odnosno programera koji su kreirali ove programe. Kompanija „Microsoft" je uvek kreirala široku zajednicu programera oko svih svojih proizvoda, uključujući i glavni proizvod Windows.

> **NAPOMENA** Značaj ove zajednice često napominju i sami lideri „Microsofta". Ukoliko u internet pretraživaču unesete ime i prezime Steve Ballmer, ne možete da promašite video snimke u kojima taj čovek veoma uzbuđeno govori: „programeri, programeri, programeri…" i ovu reč ponavlja više desetina puta.

Laneje Windows platforma napunila 27 godina. U toku svog dugog „životnog veka", ne samo operativni sistem, već i *interfejsi za programiranje aplikacija* (API interfejsi) i odgovarajući razvojni alati su značajno poboljšani. Sa ove tačke gledišta, Windows 8 predstavlja najveći korak koji je ikada napravljen u istoriji Windows operativnog sistema. Da biste razumeli ono što smo hteli da istaknemo, neophodno je da se vratimo unazad, u vreme razvoja prvih Windows aplikacija.

Moć jezika C

Iako je danas programiranje Windows aplikacija sasvim normalna aktivnost, to nije bilo tako u početku razvoja Windows operativnog sistema. U to vreme su programeri koji su razvijali MS-DOS aplikacije smatrali Windows pristup veoma čudnim – kao da je sve u potpunosti izokrenuto naopako. U standardnoj MS-DOS aplikaciji se upravljalo svim elementima sistema, a izvršavane su funkcije operativnog sistema tamo gde su bile neophodne, dok je Windows zahtevao drugačiji pristup. Operativni sistem je upravljao aplikacijom, a bilo je neophodno da mu se pristupa svaki put kada je program izvršavao određenu akciju, kao što su „osvežavanje“ korisničkog interfejsa ili izvršavanje komande iz menija.

To nije bilo usmereno protiv „sirotih“ programera. Korišćenjem C programskog jezika, koji je bio „top“ jezik tog vremena, pisanje najjednostavnije „Hello, world“ aplikacije u MS-DOS operativnom sistemu je bilo veoma jednostavno, kao u sledećem primeru:

```
#include <stdio.h>

main()
{
    printf(„Hello, world“);
}
```

Međutim, da bi se postigao identičan rezultat u Windows operativnom sistemu, bilo je neophodno mnogo više posla. Zahtevalo se pisanje „pomoćnog“ koda, pored ove printf funkcije, a sve to nije delovalo ni najmanje intuitivno, kao što je prikazano u listingu 1-1.

LISTING 1-1 **„Hello, world“ program u duhu Windows 3.1 operativnog sistema (izvod)**

```
#include <windows.h>

/* Eksportovanje ulazne tačke za izvršavanje u Windows sistemu */
long FAR PASCAL _export WndProc(HWND, UINT, UINT, LONG)

/* Ulazna tačka aplikacije */
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
    LPSTR lpszCmdParam, int nCmdShow)
{
    static char szApplication[] = „HelloW“;
    HWND hwnd;
    MSG msg;
    WNDCLASS wndClass;

    /* Kreiranje Windows klase  */
    if (!hPrevInstance)
    {
        wndClass.Style = CS_HREDRAW | CS_VREDRAW;
        wndClass.lpfnWndProc = WndProc;
        /* Nekoliko linija koda je izostavljeno, radi jednostavnosti */
        wndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
        wndClass.lpszMenuName = NULL;
        wndClass.lpszClassName = szApplication;
        RegisterClass(&wndClass);
    }
```

```
    /* Kreiranje instance prozora za klasu */
    hwnd = CreateWindow(szApplication,
        „My Hello World Program“,
        WS_OVERLAPPEDWINDOW,
        /* Nekoliko parametara je izostavljeno, radi jednostavnosti */
        hInstance,
        NULL);
    /* Iniciranje prikazivanja prozora */
    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    /* Upravljanje petljom poruka */
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg)
    }
}

/* Obrada poruka */
long FAR PASCAL _export WndProc(HWND hwnd, UINT message,
    UINT wParam, LONG lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    RECT rect;

    switch (message)
    {
        case WM_PAINT:
            hdc = BeginPaint(hwnd, &ps);
            GetClientRect(hwnd, &rect);
            DrawText(hdc, „Hello, world“, -1, &rect,
                DT_SINGLELINE | DT_CENTER | DT_VCENTER);
            EndPaint(hwnd, &pd);
            return 0;

        case WM_DESTROY:
            PostQuitMessage(0);
            return 0
    }
    return DefWindowProc(hwnd, message, wParam, lParam);
}
```

Ovaj program sadrži veliki broj linija koda, zato što je pisan na osnovu Api interfejsa, koji je imao samo funkcije operativnog sistema niskog nivoa. Iako je ovaj izvorni kod dug, otkriva značajne interne detalje Windows operativnog sistema. Naravno, svi oni su sadržani i u „srcu“ Windows 8 operativnog sistema, u poboljšanom obliku:

- Na početku, program kreira klasu prozora podešavanjem polja wndClass strukture i korišćenjem `RegisterClass` metoda. Klasa *prozora* je koncept za identifikovanje procedure (poznate kao windows procedura) obrade poruka koje se šalju prozoru.
- Program kreira prozor (pomoću `CreateWindow` metoda), koristeći registrovanu klasu prozora, a zatim ga prikazuje pomoću `ShowWindow` metoda. `UpdateWindow` metod šalje poruku prozoru kako bi ponovo bio prikazan korisnički interfejs.
- „Duša“ aplikacije je u petlji za upravljanje porukama, kao što je prikazano u sledećem kodu:

```
while (GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg)
}
```

- Ova petlja dobija poruke iz reda, prevodi pritisnute tastere u ekvivalentne poruke (na primer, ukoliko je korišćen miš), a zatim prosleđuje te poruke do odgovarajuće procedure za upravljanje prozorom.
- Ukoliko je petlja za upravljanje porukama „duša“ aplikacije, onda je procedura za upravljanje prozorom „srce“ aplikacije. U listingu 1-1 u petlji za upravljanje porukama izvršava se `WndProc` metod. Njegov message parametar sadrži kod poruke (događaja kojim se upravlja), a switch naredba se koristi da bi bili kreirani delovi koda za obradu pojedinačnih naredbi.
- `WM_PAINT` poruka definiše da ponovo bude prikazan sadržaj prozora. Pomoću `BeginPaint` metoda, dobija se resurs u kontekstu uređaja, koji se koristi za prikazivanje kljentske oblasti prozora. Ovaj kontrolni uređaj je korišćen za prikazivanje „Hello, world“ poruke na sredini prozora. `ReleasePaint` oslobađa kontekst uređaja, koji je postao veoma ograničeni resurs u sistemu.

U to doba je razvijanje Windows aplikacija bilo vremenski zahtevno i teško, zbog čega su programeri bili prinuđeni da koriste konstrukcije operativnog sistema niskog nivoa u Windows API interfejsu.

C++ pobeđuje C jezik

Brian Kernighan i Dennis Ritchie su objavili prvo izdanje jezika C 1978. godine, a Bjarne Stroustrup je pet godina kasnije kreirao novi jezik koji je dodao objektno-orijentisane aspekte u C. Novi jezik, nazvan C++, uskoro je postao popularan i na Windows platformi.

C++ omogućava enkapsuliranje podataka i funkcionalnosti u klase, a postoji podrška i za nasleđivanje i polimorfizam. Pomoću ovih karakteristika, nehijerarhijski API interfejs Windows operativnog sistema je mogao da se predstavi u obliku malog skupa entiteta koji su grupisali strukture podataka i API operacije u određeni logički kontekst. Na primer, sve operacije koje su se odnosile na kreiranje, prikazivanje i upravljanje prozorima u korisničkom interfejsu mogle su da se postave u klasu pod nazivom `Window`.

C++ pristup je pomogao programerima da steknu bolju predstavu o API interfejsu, a pojednostavio je započinjanje Windows programiranja. Na primer, osnovni delovi „Hello, world“ programa, prikazanog u listingu 1-1, mogli su se organizovati oko objekata, kao što je prikazano u listingu 1-2.

LISTING 1-2 Struktura „Hello, world" programa, pisanog u C++ jeziku (izvod)

```
// --- Klasa koja predstavlja glavni program
class Main
{
  public:
    static HINSTANCE hInstance;
    static HINSTANCE hPrevInstance;
    static int nCmdShow;
    static int MessageLoop( void );
};

// --- Klasa koja predstavlja prozor
class Window
{
  protected:
    HWND hWnd;
  public:
    HWND GetHandle( void ) { return hWnd; }
    BOOL Show( int nCmdShow ) { return ShowWindow( hWnd, nCmdShow ); }
    void Update( void ) { UpdateWindow( hWnd ); }
    virtual LRESULT WndProc( UINT iMessage, WPARAM wParam, LPARAM
lParam ) = 0;
};

// --- Klasa koja predstavlja glavni prozor ovog programa
class MainWindow : public Window
{
    // --- Implementacjia je izostavljena, radi jednostavnosti
};

// --- Izvod implementacije Main klase
int Main::MessageLoop( void )
{
    MSG msg;

    while( GetMessage( (LPMSG) &msg, NULL, 0, 0 ) )
    {
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }
    return msg.wParam;
}

LRESULT MainWindow::WndProc( UINT iMessage, WPARAM wParam, LPARAM lParam )
{
    switch (iMessage)
    {
        case WM_CREATE:
            break;
        case WM_PAINT:
            Paint();
            break;
        case WM_DESTROY:
            PostQuitMessage( 0 );
            break;
```

LISTING 1-2 (nastavak)

```
            default:
                return DefWindowProc( hWnd, iMessage, wParam, lParam );
        }
        return 0;
    }
```

U objektno-orijentisanom pristupu koji je ponudio C++ jezik funkcionisanje objekta je definisano u bibliotekama koda koje su se mogle više puta koristiti. Programeri su mogli da kreiraju svoje programe na osnovu ovih biblioteka, tako da je bilo neophodno da se definišu samo one funkcije koje su se razlikovale od ugrađenih. Na primer, bilo je neophodno da se predefiniše Paint() metod, koji je prikazan u listingu 1-2, da bi se ponovo prikazivao korisnički interfejs window objekata.

Windows programiranje se u značajnoj meri promenilo uvođenjem C++ jezika i biblioteka objekata. Kompanija „Microsoft“ je kreirala dve osnovne bilioteke - Microsoft Foundation Classes (MFC) i Active Template Library (ATL), koje i danas održava u svom osnovnom integrisanom razvojnom okruženju nazvanom Visual Studio.

NAPOMENA Uskoro ćete naučiti nešto više o Visual Studio razvojnom okruženju.

Visual Basic

Aplikacije napisane u programskim jezicima C i C++ prikazivale su dosta detalja o funkcionisanju Windows operativnog sistema. U nekim situacijama je neophodno znati sve pojedinosti, ali najčešće to može da bude furstrirajuće i da omete programere da se fokusiraju na funkcionalnost realne aplikacije.

Objavljen u maju 1991. godine, Visual Basic je dramatično promenio stil programiranja. Umesto da prikazuje interne detalje Windows operativnog sistema, Visual Basic ih je prikrio od programera, kojima je stavio na raspolaganje konstrukcije visokog nivoa, kao što su forme, kontrole, moduli, klase i datoteke sa kodom. Umesto pisanja desetina linija za kreiranje veoma jednostavne funkcije u programu, programski jezik Visual Basic je omogućio programerima fokusiranje na realne funkcije svojih aplikacija. Program „Hello, world“ je mogao da se napiše pomoću samo jedne naredbe:

```
MsgBox(„Hello, World!“)
```

Nema podešavanja klasa prozora, nema registrovanja prozora, nema programiranja petlje za očitavanje i analizu poruka! Koncepti visokog nivoa ovog jezika učinili su potpuno nepotrebnim upotrebu ovakvih detalja. Sve ovo je implementirao Visual Basic u toku samog izvršavanja aplikacije.

Zbog korišćenja grafičkog integrisanog razvojnog okruženja (*eng*. Integrated Development Environment; skraćeno - IDE), ovaj način razvoja aplikacija se i dalje najviše preporučuje, jer je najproduktivniji. Programeri grafički dizajniraju prozore aplikacije za dijalog sa korisnicima (ili forme, u terminologiji Visual Basic programskog jezika) prevlačenjem elemenata korisničkog interfejsa (kontrola) na površinu forme iz palete sa kontrolama u integrisanom razvojnom okruženju. Svaka kontrola ima nekoliko rukovalaca događajima, koji odgovaraju na događaje koji se javljaju u okruženju – na primer, kada korisnik klikne odgovarajući taster ili izabere drugu stavku u kombinovanom polju. Programiranje se, zapravo, svodi na pisanje koda kojim se upravlja (rukuje) određenim događajima.

Kompanija „Microsoft“ je u toku 1993. godine razvila binarni standard component object model (COM) - njime je omogućeno kreiranje objekata koji se mogu više puta koristiti u različitim aplikacijama. Brojne tehnologije kreirane na osnovu COM modela, kao što je Object Linking and Embedding (OLE), omogućile su automatizovanje aplikacija. Sve verzije Visual Basic programskog jezika nakon 1993. godine su zasnovane na COM i OLE modelima. Ovaj pristup je bio tako uspešan da je dijalekt jezika Visual Basic for Applications (VBA) postao programski jezik za pisanje Microsoft Office makroa.

Delphi

Visual Basic nije bio jedini jezik koji se „uhvatio u koštac“ sa programskim jezicima C i C++. Originalno napravljen u kompaniji „Borland“, Delphi je koristio Object Pascal programski jezik. Dok je Visual Basic bio jezik zasnovan na objektima (imao je podršku za klase sa enkapsuliranjem podataka i funkcija, ali nije dozvoljavao nasleđivanje objekata), Object Pascal je bio pravi objektno-orijentisani jezik.

Integrisano razvojno okruženje za Delphi (čija je prva verzija objavljena 1995. godine) bilo je veoma slično Visual Basic IDE okruženju. Delphi je dizajniran kao alat za brzi razvoj aplikacija (*eng*. Rapid Application Development; skraćeno - RAD), koji je podržavao razvoj aplikacija za pristupanje bazama podataka, uključujući tu jednostavne sisteme, ali i one namenjene kompanijama.

Proizvod je veoma brzo evoluirao, uz pojavu pet verzija u prvih pet godina postojanja. Delphi je bio prvi alat koji je omogućavao razvoj 32-bitnih aplikacija za Windows operativni sistem. Slično Visual Basic kontrolama, pružao je mogućnost korisnicima da koriste više od 100 vizuelnih komponenata (organizovanih u Visual Component Library), koje su programeri mogli odmah da koriste. Pored toga, programeri su mogli da jednostavno kreiraju sopstvene vizuelne komponente i da ih dodaju u određenu postojeću biblioteku.

Nastanak .NET tehnologije

U toku 2002. godine je.NET Framework uveo novinu u razvoj windows aplikacija. .NET programi se prevode u međujezik Microsoft Intermediate Language (MSIL). Ovaj međukod se transformiše u operacije koje se mogu izvršavati na određenom CPU procesoru u toku izvršavanja pomoću just-in-time (JIT) prevodioca (kompajlera). Novi pristup uveo je nekoliko novih paradigmi u razvoj Windows aplikacija, uključujući sledeće:

- Pre uvođenja .NET tehnologije, svaki jezik (i razvojno okruženje) koristio je sopstvenu biblioteku za izvršavanje. Učenje novog jezika je podrazumevalo i učenje nove bibiloteke koja se koristi prilikom izvršavanja. Zahvaljujući uvođenju .NET tehnologije, svi jezici koriste iste rutine prilikom izvršavanja. U toku 2002. godine su samo dva jezika imala podršku u kompaniji „Microsoft“ (C# i Visual Basic.NET). Danas je u upotrebi više od 100 .NET jezika. „Microsoft“ je dodao F# u ovu listu, a podržava IronPython i IronRuby (koje razvijaju posebne zajednice programera).
- Pomoću mehanizma za upravljanje otpadom, u toku izvršenja se automatski upravlja alokacijom memorije i uklanjanjem nepotrebnih objekata. Ovo funkcionisanje pomaže poboljšanju produktivnosti, a omogućava programerima da kreiraju kod koji je manje podložan greškama.

  Automatsko uklanjanje nepotrebnih objekata smanjuje i verovatnoću kreiranja programa koji imaju probleme prilikom upravljanja memorijom.

- Umesto API metoda niskog nivoa, programeri mogu da koriste objekte kojima je prikrivena složenost API interfejsa. Umesto upravljanja detaljima koji se odnose na Windows operativni sistem, mogu da koriste apstrakcije višeg nivoa, čime se bitno poboljšava njihova produktivnost.
- .NET obezbeđuje značajnu kooperaciju između COM i .NET objekata. Sam .NET kod ne samo da može da pristupa Com objektima, već i obezbeđuje objekte koji se mogu koristiti u COM svetu.

.NET se naziva upravljačkim okruženjem, a njegovi jezici se nazivaju upravljanim (nadziranim) jezicima – za razliku od osnovnih jezika, kao što su C, Object Pascal (Delphi) i C++, koji omogućavaju kreiranje koda za specifični CPU.

NAPOMENA .NET Framework nije i prvo okruženje za upravljanje izvršavanjem. Ta „titula" pripada programskom jeziku Java, koji je objavio „Sun Microsystems" u toku 1995. godine. .NET je bio odgovor „Microsofta" na Java fenomen; veliki broj karakteristika je inspirisan Java implementacijom kompanije „Sun".

Objavljeno zajedno sa .NET radnim okvirom, integrisano razvojno okruženje Visual Studio je odigralo značajnu ulogu u uspehu .NET tehnologije. Visual Studio je imao dvadesetak projektnih šablona pomoću kojih je omogućeno brže započinjanje razvoja Windows aplikacija. Ultimate izdanje Visual Studio integrisanog razvojnog okruženja danas sadrži više od 100 projektnih šablona.

NAPOMENA Više detalja o Visual Studio integrisanom razvojnom okruženju naći ćete u Poglavlju 4.

Visual Studio šabloni su veoma korisni. Na primer, možete da kreirate aplikaciju koja je prikazana na slici 1-6 uz samo nekoliko klikova, koristeći Windows Forms Application šablon.

SLIKA 1-6 Jednostavna Windows Forms aplikacija kreirana pomoću Visual Studio integrisanog razvojnog okruženja

Koristeći alate iz Visual Studio integrisanog razvojnog okruženja, možete veoma jednostavno da definišete vizuelne karakteristike prozora koji je prikazan na slici 1-6. Listing 1-3 sadrži samo kod koji treba da ručno unesete u ovom primeru.

LISTING 1-3 Kod koji stoji iza forme prikazane na slici 1-6

```
using System;
using System.Windows.Forms;

namespace HelloWorldWinForms
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            label1.Text = „Hello from Windows Forms!“;
        }
    }
}
```

Iako je .NET tehnologija obogaćena bibliotekama objekata i sjajnim alatima, i dalje je koristila API interfejs koji je bio zasnovan na dizajnu prvih verzija Windows operativnog sistema.

Nove tehnologije korisničkog interfejsa

Dugo se Windows korisničkim interfejsom upravljalo pomoću Graphics Device Interface (GDI) API interfejsa, koji je evoluirao u GDI+ nakon objavljivanja Windows XP operativnog sistema. GDI i GDI+ su API interfejsi zasnovani na rasterima, a sve standardne kontrole u korisničkom interfejsu Windows operativnog sistema su ih koristile za renderovanje. Jedini način da programer promeni inicijalni prikaz bilo koje standardne kontrole bilo je predefinisanje Windows događaja koji je prikazivao korisnički interfejs kontrola.

Kada je uveden Windows Presentation Foundation (WPF) grafički podsistem u .NET Framework 3.0 (i novije verzije, kao ugrađena komponenta Windows Vista operativnog sistema), GDI paradigma je totalno promenjena. Umesto kreiranja korisničkog interfejsa na imperativni način (odnosno, korišćenjem instrukcija napisanih u programskom jeziku), WPF za opisivanje elemenata korisničkog interfejsa primenjuje eXtensible Application Markup Language (XAML), koji je derivat eXtensible Markup Language (XML) jezika. WPF koristi i izuzetne mogućnosti hardverske akceleracije grafičkih procesnih jedinica (GPU jedinica) ugrađenih u same računare.

Silverlight (internet aplikacioni radni okvir kompanije „Microsoft“) takođe primenjuje XAML prilikom definisanja korisničkog interfejsa. U listingu 1-4 prikazan je veoma jednostavan XAML primer u kome se implementira „Hello, world“ aplikacija primenom Silverlight tehnologije.

LISTING 1-4 MainPage.xaml datoteka „Hello, world“ aplikacija

```
<UserControl x:Class="HelloFromSL.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

LISTING 4-1 (nastavak)

```
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock FontSize="48">Hello from Silverlight</TextBlock>
    </Grid>
</UserControl>
```

Ovaj kod generiše stranicu prikazanu na slici 1-7. Tekst na toj slici se prikazuje kao rezultat izvršavanja linije koda prikazane podebljano u listingu 1-4.

SLIKA 1-7 Rezultat izvršavanja listinga 1-4

WPF i Silverlight su sjajne tehnologije. Njihovo korišćenje izgleda kao zamena „Lamborghini" motora iz tridesetih godina 20. veka (GDI/GDI+) savremenim motorom koji je proizveden 2012. godine (WPF/Silverlight).

Ove tehnologije ne samo da definišu korisnički interfejs, već mogu da deklarišu njegovo dinamičko funkcionisanje, bez potrebe za pisanjem koda (ili uz minimalno pisanje). One povezuju korisnički interfejs sa slojem (modelom) logike same aplikacije. Sledi prikaz nekih od najznačajnijih svojstava ovih tehnologija:

- Ove tehnologije su dizajnirane i oblikovane tako da možete da kreirate bogate, moćne desktop ili internet aplikacije sa složenim korisničkim interfejsom. Pored obezbeđivanja veoma jednostavnih elemenata korisničkog interfejsa (kao što su polja za prikazivanje tekstova, tasteri, liste, kombinovana polja za unos, slike i ostalo), daju vam i slobodu da kreirate sadržaj sa *animacijama* i *multimedijalnim* elementima, kao što su video i zvuk. Nasuprot tradicionalnom pristupu razvoja korisničkog interfejsa pomoću pravougaonih elemenata, WPF i Silverlight omogućavaju da se menja celokupan izgled aplikacije.
- Ove tehnologije obezbeđuju veoma *fleksibilan sistem* za raspoređivanje elemenata - on omogućava da se jednostavno definišu rasporedi elemenata, koji se mogu automatski prilagođavati na osnovu određenog broja faktora (kao što su raspoloživa veličina ekrana, broj prikazanih elemenata, veličina prikazanih elemenata i uvećanje).

- *Stilovi* i *šabloni* su svojstva koja utiču na bolju saradnju između programera i dizajnera. Programeri implementiraju logiku aplikacije, tako da nikada ne definišu direktno vizuelna svojstva korisničkog interfejsa. Umesto toga, oni signalizuju programski da se menja stanje korisničkog interfejsa. Dizajneri kreiraju vizuelne elemente korisničkog interfejsa, uzimajući u obzir broj njegovih mogućih stanja.
- Silverlight i WPF primenjuju, deklarativno (drugim rečima, bez potrebe za pisanjem dodatnog koda), *povezivanje podataka* – odnosno, tehnologiju koja se koristi za povezivanje elemenata korisničkog interfejsa sa podacima ili drugim njegovim elementima. Povezivanje podataka funkcioniše u kooperaciji sa stilovima, šablonima, rasporedima elemenata, pa čak i animacijama. Ovaj mehanizam je posebno koristan u line-of-business (LOB) aplikacijama. Korišćenjem informacija koje se odnose na povezivanje podataka, koje se dobijaju od baze podataka i obrađuju pomoću logike aplikacije, elementi se mogu deklarativno povezati sa elementima korisničkog interfejsa.

NAPOMENA Sistem za upravljanje rasporedom omogućava raspoređivanje elemenata korisničkog interfejsa. Zahvaljujući uvođenju fleksibilnog sistema za upravljanje rasporedom (kao onog koji koriste WPF, Silverlight i Windows 8), može da se definiše veza između elemenata korisničkog interfejsa, a oni se mogu u njemu automatski postavljati u toku izvršavanja. Na primer, može da se definiše sistem za upravljanje rasporedom tako da uređuje kontrole korisničkog interfejsa u dve kolone i da ih ravnomerno raspoređuje.

Stilovi i šabloni u WPF i Siverlight tehnologijama se koriste slično kao i stilovi i šabloni u sistemima za objavljivanje dokumenata. Stil sadrži kolekciju zajedničkih atributa elemenata korisničkog interfejsa, a šablon je skelet sa definisanim lokacijama za prikazivanje sadržaja tih elemenata.

Primarna prednost korišćenja WPF i Silverlight tehnologija je, pre svega, u mogućnosti jasnog razdvajanja programerskih zadataka koji se odnose na korisnički interfejs od onih koje izvršavaju sami dizajneri. Današnje aplikacije orijentisane ka korisnicima obezbeđuju prednost u poređenju sa uslovima u kojima su zadaci programera i dizajnera međusobno isprepleteni.

NAPOMENA XAML ima značajnu ulogu u Windows 8 aplikacijama, kao što ćete naučiti u Poglavlju 3.

CATCH-22 RAZVOJA WINDOWS APLIKACIJA

U toku evolucije Windows aplikacija, tehnologije i povezane tehnike su se razvijale u dva smera. Jedan smer je osnovni razvoj, koji je započeo programskim jezikom C u prvim danima razvoja Windows operativnog sistema. Drugi smer je upravljani razvoj, u kome se primenjuje .NET Framework, zajedno sa povezanim tehnologijama i jezicima.

Zbog prevođenja koda za specifični CPU i za pristup Windows API interfejsu na niskom nivou, osnovne aplikacije mogu da obezbede bolje performanse, koje se ne postižu upravljanim tehnologijama. Međutim, prilikom pisanja poslovnih aplikacija ovakav kod je manje produktivan, a glomazniji je u odnosu na upravljani kod i .NET Framework. Pošto, u suštini, poslovne aplikacije pristupaju bazama podataka, smanjenje performansi dodatnih jezika se ne može ni primetiti – aplikacija troši najveći deo vremena na komunikaciju sa određenim servisima i bazama podataka.

Kada programeri Windows aplikacija odlučuju o pravoj tehnologiji i alatu za razvoj određene aplikacije, oni imaju sličan problem kao i Yossarian, pilot bombardera u noveli čiji je autor Joseph Heller, *Catch-22* (New York: Alfred A. Knopf, 1995).

U brojnim situacijama, posebno kada se kreiraju desktop aplikacije, ne postoji idealan izbor između razvoja zasnovanog na osnovnim jezicima i .NET tehnologiji. Iako su WPF i Silverlight izvanredne tehnologije za kreiranje spektakularnih aplikacija, nije ih moguće koristiti u osnovnim jezicima za razvoj - nisu dovoljno duboko ugrađene u operativni sistem. Na primer, startovanje WPF aplikacije zahteva određeno vreme dok se WPF podsistem ne učita u memoriju. U velikom broju situacija u kojima je neophodno postizanje vrhunskih performansi prilikom izračunavanja .NET daje mnogo lošije rezultate nego kod koji je pisan osnovnim jezicima.

Visual Studio je sjajan primer ove šizofrene situacije. On koristi kombinovanu bazu koda. Najveći deo komponenata implementiran je korišćenjem C++ programskog jezika, a delovi su napravljeni korišćenjem C# jezika (i WPF). Iskačući ekran treba da se odmah prikaže nakon startovanja aplikacije. On je implementiran u C++ jeziku pomoću GDI+ tehnologije, jer WPF nije dovoljno brza tehnologija. Editor koda u Visual Studio integrisanom razvojnom okruženju je implementiran pomoću WPF-a, zato što sadrži bogat i za korišćenje jednostavan skup alata za hostovanje tako složene komponente sa sjajnim korisničkim interfejsom.

U toku razvoja LOB aplikacija, produktivnost je najznačajniji faktor. Prednost boljih performansi razvoja zasnovanog na osnovnom kodu ne znači previše u razvoju same aplikacije, zato što, u većini situacija, sloj baze podataka predstavlja „usko grlo“ kada je reč o performansama sistema. Kada se koriste savremenije tehnologije, faza programiranja poslovne aplikacije postaje mnogo brža i otpornija na različite greške.

Zato, kada pišete desktop aplikaciju, osnovni kod obezbeđuje optimalne performanse, ali u ovoj situaciji ne možete da koristite bogati skup alata koje pružaju WPF i Silverlight. Kada koristite novije programske jezike, možete da postignete veću produktivnost, ali se gube performanse finalne aplikacije. Konačno, možete da koristite određeno „kombinovano“ rešenje – kao što to čini Visual Studio, ali ćete u toj situaciji imati puno glavobolja. Izbor tehnologije nije jednostavna odluka, zar ne?

Ova situacija se značajno menja sa dolaskom Windows 8 operativnog sistema. Web programeri sa iskustvom u pisanju HTML5/cascading style sheets 3 (CSS3)/JavaScript koda, vešti C++ programeri i programeri koji su koristili .NET tehnologiju mogu da se osećaju kao da je njihovo znanje dovoljno i odgovarajuće za razvoj Windows 8 aplikacija. Ne postoji privilegovani skup ljudi kao što su bili C programeri na početku ere razvoja Windows aplikacija. Svako može da koristi iste tehnologije i iste alate – bez ikakvih kompromisa. Kako je to moguće? Saznaćete u Poglavlju 3.

PREGLED POGLAVLJA

Windows platforma je značajno evoluirala u prethodnih 27 godina. Transformisana je iz jednostavnog MS-DOS proširenja u složeni i sveobuhvatni operativni sistem koji se koristi na većini računara širom sveta. Windows je projektovan za one ljude koji se bave obradom podataka i za iskusne korisnike računara. Iako se svakom novom verzijom približavao manje naprednim korisnicima, nikada nije postao operativni sistem za jednostavne uređaje potrošačke elektronike, kakvi su iPhone i iPad uređaji kompanije „Apple“.

Windows 8 to menja iz korena. Pored tradicionalnog desktop režima, koji je svima poznat, ovaj operativni sistem ima novi izgled, sa Windows 8 aplikacijama koje pružaju korisnicima sasvim novo intuitivno iskustvo – aplikacija zauzima čitav ekran i zaokuplja potpunu pažnju korisnika.

Promenjen je operativni sistem, a razvojni alati i API interfejsi su značajno poboljšani. U prošlosti su samo C i C++ programeri bili privilegovani za kreiranje Windows aplikacija, a danas C++, Visual Basic, C# i Delphi programeri mogu da razvijaju aplikacije u svojim omiljenim integrisanim razvojnim okruženjima. Zavisno od izabranog jezika i alata, programeri moraju da koriste različite API interfejse i tehnologije. Za razliku od osnovnih programskih jezika (kao što su C, C++ i Delphi), koji omogućavaju kreiranje koda specifičnog za CPU i bolje performanse, nadzirani kod omogućava mnogo veću produktivnost pri razvoju poslovnih aplikacija i korišćenje modernih tehnologija za razvoj korisničkog interfejsa, kao što su WPF i Silverlight.

Windows 8 stil razvoja aplikacija obezbeđuje isti API interfejs za kod iz obe grupe jezika – bez kompromisa. Štaviše, ovaj API interfejs je raspoloživ i HTML5/CSS3/JavaScript programerima.

Windows 8 menja ne samo korisnički interfejs operativnog sistema, već i način na koji korisnici interaguju sa sistemom. Pored podrške za tradicionalne ulazne uređaje, kao što su tastatura i miš, obezbeđuje i prvoklasnu podršku za interakciju sa operativnim sistemom i Windows 8 aplikacijama, podržavajući složene gestove na ekranu osetljivom na dodir, olovku i senzore, kao što su žiroskop i akcelerometar.

U Poglavlju 2 naučićete nešto više o ovim novim mogućnostima Windows operativnog sistema.

VEŽBE

1. Koji operativni sistem kompanije „Microsoft" u Windows familiji je dizajniran i razvijen tako da je orijentisan ka potrošačima?
2. Da li Windows 8 aplikacije imaju naslovnu liniju prozora i ivice?
3. Navesti programske jezike koji se često koriste u razvoju Windows aplikacija.
4. Koju grupu jezika mogu da koriste web programeri u cilju kreiranja Windows 8 aplikacija primenom stila specifičnog za ovaj operativni sistem?

NAPOMENA Odgovore na pitanja možete da pronađete u Dodatku A.

NAUČILI STE U OVOM POGLAVLJU

Tema	Ključni koncepti
Windows operativni sistem	Windows je familija operativnih sistema, koja je nastala sa Windows 1.0 sistemom, objavljenim 20. novembra 1985. godine, kao dodatnom komponentom MS-DOS operativnog sistema. Familija uključuje i operativne sisteme namenjene prenosnim uređajima (Windows Embedded) i mobilnim telefonima (Windows CE, Windows Mobile i Windows Phone). Najnovije verzije su Windows 8 i Windows Phone 7.5.
Windows API interfejsi	Aplikacije koje funkcionišu u Windows operativnom sistemu mogu da pristupaju odgovarajućim servisima operativnog sistema korišćenjem Windows programskih interfejsa aplikacija (eng. application programming interface; skraćeno - API).
C i C++ programski jezici	Kao dva programska jezika koji se mogu koristiti za razvoj Windows aplikacija, C i C++ koriste sve funkcije operativnog sistema, uljučujući i one koje su specifične za hardver i one koje su veoma niskog nivoa. Ovi jezici su korišćeni za razvoj aplikacija od početka razvoja Windows operativnog ssitema. C++ je proširio C jezik objektno-orijentisanim elementima i postao veoma stabilan programski jezik u poslednjih četvrt veka.
Visual Basic programski jezik	Objavljen 1991. godine, Visual Basic je revolucionarni programski jezik koji je značajno smanjio neophodne napore programera, omogućavajući korišćenje koncepata korisničkog interfejsa na visokom nivou, kao što su forme, kontrole i događaji. Obezbedio je i jednostavnu integraciju sa modelom objekata komponenata (component object model - COM) i sa Object Linking and Embedding (OLE) komponentama.
.NET Framework	Microsoft .NET Framework je okruženje za izvršavanje, koje se nalazi između operativnog sistema i aplikacija. Njegov primarni zadatak je obezbeđivanje pristupa svim servisima operativnog sistema na kontrolisani način, koji omogućava programerima da budu mnogo produktivniji. C# i Visual Basic .NET su dva najčešće korišćena jezika za kreiranje aplikacija za .NET Framework.
WPF i Silverlight	Windows Presentation Foundation (WPF) i Silverlight su moderne i sofisticirane platforme za kreiranje korisničkog interfejsa, koje se zasnivaju na .NET radnom okviru. One omogućavaju kreiranje aplikacija koje sadrže veoma složen korisnički interfejs, uključujući i fleksibilan raspored elemenata, multimedijalne elemente i animacije. Primenjuju tehnike kao što su stilovi, šabloni i povezivanje podataka da bi se omogućio veoma efikasan način za povezivanje sloja korisničkog interfejsa aplikacija sa samom logikom.

Tema	Ključni koncepti
Windows 8 style aplikacije	Windows 8 style aplikacije su nova forma aplikacija koja se uvodi u Windows 8 operativni sistem. One se potpuno razlikuju od tradicionalnih desktop aplikacija, zbog potpuno drugačijeg vizuelnog prikaza i načina na koji se upravlja korisničkim interakcijama. Windows 8 aplikacije zauzimaju čitav ekran (odnosno, ne sadrže okvir aplikacije), a od korisnika se više ne zahteva da prikazuju veći broj prozora u okviru aplikacije i da upravljaju njima.