ULAZNA TOČKA APLIKACIJE, KREIRANJE PROZORA, MESSAGEBOX PROZOR

Maja Štula

Ak. God. 2011/2012

ULAZNA TOČKA WIN32 APLIKACIJE

- Glavna funkcija od koje počinje izvršavanje (ulazna točka) Win32 aplikacije je WinMain.

```
int WINAPI WinMain
( HINSTANCE hInstance,                  // handle to current instance
  HINSTANCE hPrevInstance,      // handle to previous instance
LPSTR lpCmdLine,                        // pointer to command line
    int nCmdShow                        // show state of window );
```

- Funkcija WinMain se poziva na način da se jednostavno napiše:

```
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE
    hPrevInstance,LPSTR lpCmdLine,int nShowCmd)
{   }
```

ULAZNA TOČKA WIN32 APLIKACIJE

- Funkciju WinMain osim za početak izvršavanja aplikacije trebate koristiti ako trebate aplikaciji preko komandne liniji poslati početne parametre
- Inače nema nikakvih izmjena na kodu. Stavite:

//Win32 API aplikacija u C-u

```
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR
    lpCmdLine,int nShowCmd)
{ //kod aplikacije}
```

- Primjeri ulaznih točaka (*entry point*) aplikacija:

//Command prompt aplikacija u C-u

```
void main (int argc, char * argv[])
{ //kod aplikacije }
```

ULAZNA TOČKA .NET 2.x APLIKACIJE

```
//Aplikacija u C#
class MyApp
{
   // Ulazna točka aplikacije
   public static void Main(string[] args)
   {    }
}
```

ULAZNA TOČKA .NET 3.x APLIKACIJE

- .NET 3.x klasa System.Windows.Application uključuje sve osnovne funkcionalnosti aplikacije od ulazne točke aplikacije, pumpe poruka do zaustavljanja aplikacije.
- Samostalne aplikacije u .NET 3.x mogu i dalje koristiti uobičajenu ulaznu točku aplikacije u obliku statičke metode Main bez kreiranje instance klase Application .
- Međutim XBAP (*XAML Browser Application*) aplikacije koje se izvode u pregledniku obavezno moraju koristiti objekt Application.

ULAZNA TOČKA .NET 3.x APLIKACIJE

```
namespace aplikacija
{
        public class AppCode
        {
                public static void Main()
                {  MessageBox.Show("Moja prva WINFX aplikacija", "Naslov", MessageBoxButton.OK);
                    }
        }
}

namespace aplikacija
{
public class AppCode
        {
        public static void Main()
        {  Application app = new Application();
           app.Run(new Window()); }
        }
}
```

MESSAGEBOX PROZOR

WIN32 API MessageBox funkcija

- Funkcija **MessageBox** kreira, prikazuje i radi s prozorom poruke (message box).

```
int MessageBox(
HWND hWnd,          // handle of owner window
LPCTSTR lpText,     // address of text in message box
LPCTSTR lpCaption,  // address of title of message box
UINT uType          // style of message box);
```

Parametri MessageBox funkcije

- *hWnd* - To je handle na prozor iz kojeg se pozvala funkcija MessageBox. Ako je taj parametar NULL tada poruka nije pozvana ni iz jednog prozora.
- *lpText* - Pokazivač na string poruke.
- *lpCaption* - Pokazivač na string naslova poruke. Ako je taj parametar NULL, defaultni naslov je "Error".
- *uType* - Specificira stil poruke. Neke vrijednosti su MB_ABORTRETRYIGNORE, MB_OK, MB_OKCANCEL, MB_ICONERROR, ….

C# MessageBox klasa

- System.Windows.Forms.MessageBox - klasa MessageBox kreira, prikazuje i radi s prozorom poruke. Ne može se kreirati instanca klase već se prozor poruke kreira pozivom statičke metode Show.
- Naslov prozora, poruka, botuni i ikona određeni su parametrima metode Show.

```
MessageBox.Show(“Poruka”);
MessageBox.Show(“Poruka”, “Naslov”,
    (MessageBoxButtons) botuni, (MessageBoxIcon)
    ikona);
```

C# MessageBox klasa

- MessageBoxButtons je enumeracija sa elementima AbortRetryIgnore, OK, OKCancel, RetryCancel, YesNo i YesNoCancel koji određuju botune koji će se prikazati na poruci.
- MessageBoxIcon je također enumeracija sa elementima Asterisk, Error, Exclamation , Hand, Information, None , Question, Stop, i Warning koji određuju koja će se ikona prikazati na poruci.

“OBIČNI PROZORI”

Klasa Form

- Svi prozori moraju biti izvedeni iz .NET klasa predložaka prozora.
- Glavni prozor aplikacije je instanca klase izvedene iz System.Windows.Forms.Form klase iz biblioteke *System.Windows.Forms.dll*.
- Pod pojmom forme u .NET podrazumijevaju se prozori najviše razine stoga svi prozori aplikacije najviše razine trebaju biti izvedeni iz System.Windows.Forms.Form klase.

Visual Studio

- Visual Studio pomoćnik razdvaja kôd forme u dvije datoteke (Form1.cs i Form1.Designer.cs) zbog preglednosti kôda što je u .NET 2.x omogućeno uvođenjem modifikatora *partial*.
- U jednu datoteku se smješta kôd koji je vezan uz dizajn forme, a u drugu kôd vezan uz funkcionalnost i ponašanje forme.

Svojstva klase Form

- Svojstvo WindowState definira početno stanje prozora. Moguće vrijednosti svojstva definirane su enumeracijom System.Windows.Forms.FormWindowState. Uobičajena vrijednost je Normal. Ukoliko bi se svojstvo postavilo na vrijednost Maximized to bi formi dodalo stil WS_MAXIMIZE. Prozor bi se u tom slučaju prikazao maksimimiziran preko cijelog ekrana.
- Svojstvo Size određuje dimenzije prozora.
- Svojstva MinimizeBox, MaximizeBox i ControlBox, koja mogu poprimiti vrijednosti true i false, uključuju ili isključuju stilove WS_MINIMIZEBOX, WS_MAXIMIZEBOX i WS_SYSMENU.

Svojstva klase Form

- Boja pozadine prozora postavlja se svojstvom forme BackColor. Neka svojstva prozora u WIN32 API aplikaciji treba postaviti u WNDCLASSEX strukturi, a neka prilikom kreiranja prozora pozivom funkcije CreateWindowEx.
- U .NET aplikaciji sva svojstva forme su članovi klase. Vrijednosti svojstava su ili osnovni tipovi .NET podataka, ili .NET enumeracije ili instance drugih .NET klasa.
- Npr. skidanje naslova forme moguće je promjenom svojstva FormBorderStyle u None.

Klasa Window

- Kako je .NET 3.x uveo novi grafički podsustav izmijenjene su i klase predložaka prozora.
- Osnovna klasa za kreiranje prozora najviše razine nije više System.Windows.Forms.Form, kao kod .NET-a 2.x, već je to sada klasa System.Windows.Window iz biblioteke *System.dll*.