Arhitektura i način rada Windows OS-a

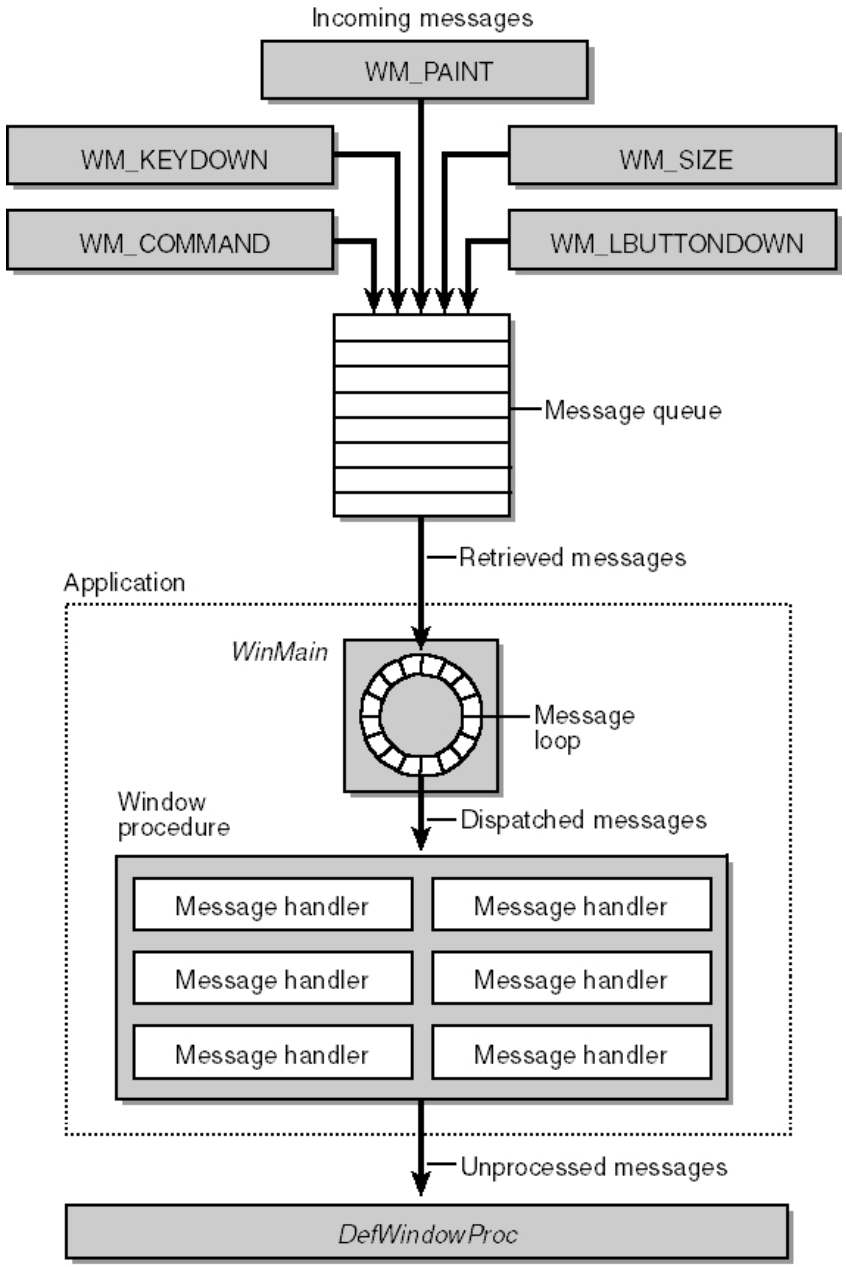
Maja Štula

Ak. God. 2011/2012

Svaka Windows aplikacija sa grafičkim sučeljem **treba imati petlju poruka** kako bi aplikacija funkcionirala “normalno”.

Zadatak petlje poruka je da aplikaciju dovede u stanje čekanje (bez opterećenja procesora) na ulazne događaje koji se aplikaciji prosljeđuju kroz red poruka aplikacije.

Petlja poruka, kada postoji neka poruka, dohvaća poruku iz reda i prosljeđuje je prozoru aplikacije kojem je poruka namijenjena.



WIN32 API PETLJA PORUKA

BOOL GetMessage(
LPMSG *lpMsg*,
HWND *hWnd*,
UINT *wMsgFilterMin*,
UINT *wMsgFilterMax*);

- Funkcija GetMessage dohvaća poruku iz red poruka.
- Prvi argument je adresa strukture tipa MSG u koju će se pohraniti poruka.
- Ako se navede drugi parametar u funkciji (to je handle na neki prozor) onda će petlja dohvaćati poruke samo za taj prozor. Ukoliko se taj argument postavi u NULL funkcija će dohvaćati poruke za sve prozore promatranog thread-a ili procesa unutar kojeg se nalazi poziv funkcije.
- Zadnja dva argumenta se koriste za ograničenje dohvaćenih poruka.

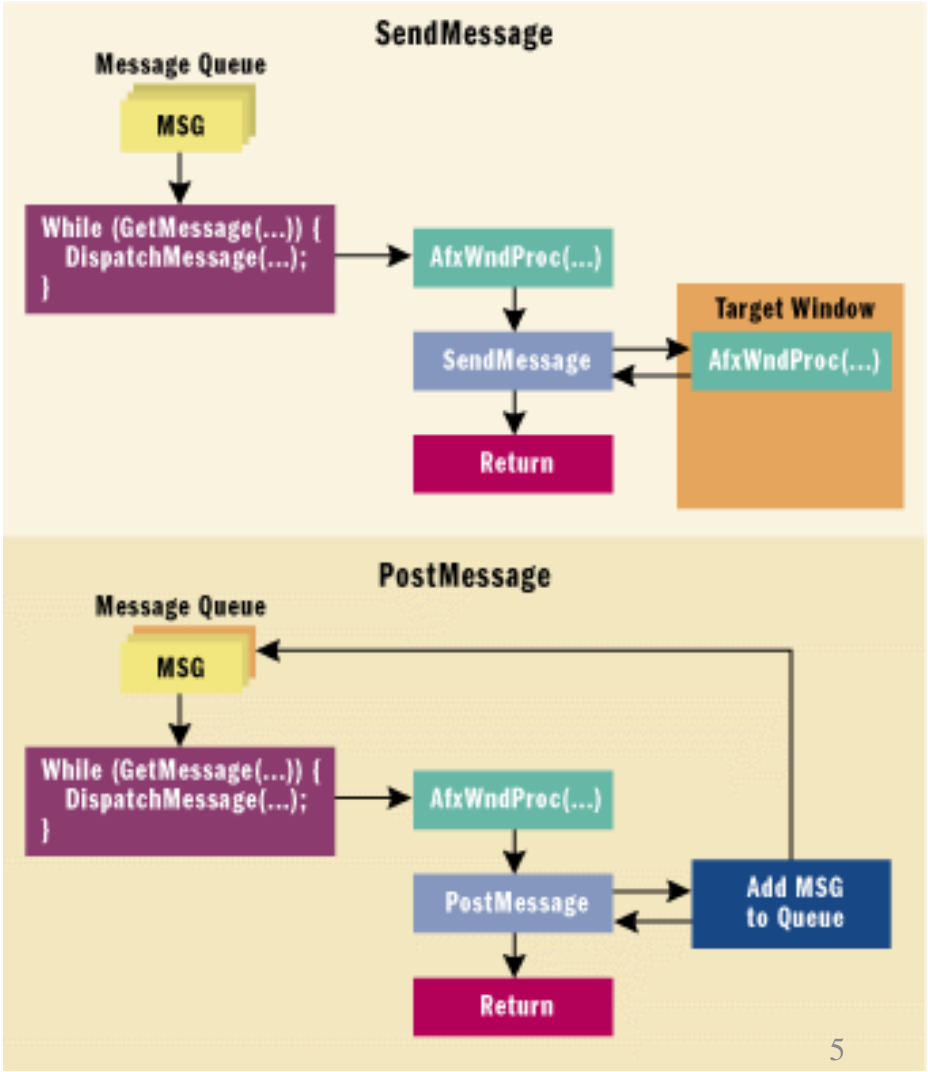
WIN32 API PETLJA PORUKA

- Glavni proces (thread) aplikacije počinje sa petljom poruka nakon inicijalizacije aplikacije i stvaranja barem jednog prozora. Nakon što je pokrenuta petlja poruka nastavlja primati poruke iz reda čekanja i slati ih odgovarajućem prozoru. Petlja poruka završava kada funkcija GetMessage dohvati WM_QUIT poruku iz reda čekanja.
- Samo jedna petlja poruka je potrebna čak i ako aplikacija ima više prozora. DispatchMessage uvijek šalje poruku odgovarajućem prozoru jer svaka poruka u redu čekanja je MSG struktura koja sadrži handle na prozor kojem poruka pripada.

WIN32 API RAD SA PORUKAMA

BOOL PostMessage(
HWND *hWnd*,
UINT *Msg*,
WPARAM *wParam*,
LPARAM *lParam*);

LRESULT SendMessage(
HWND *hWnd*,
UINT *Msg*,
WPARAM *wParam*,
LPARAM *lParam*);



Dohvaćanje poruka u MFC aplikaciji

- MFC aplikacija treba imati jedan (i samo jedan) objekt tipa *CWinApp* class. Taj objekt uključuje u sebi petlju poruka za dohvaćanje poruka i prosljeđuje poruke prozorima aplikacije.

Primjer:

```
class CFCMApp : public CWinApp
{
    DECLARE_MESSAGE_MAP()
};
CFCMApp theApp;
```

Dohvaćanje poruka u .NET 2.x C# aplikaciji

- System.Windows.Forms.Application (.NET 2.x aplikacije) – je klasa koja pruža statičke metode i svojstva za upravljanje aplikacijom tj. za zaustavljanje aplikacije, za obradu poruka i sl.
- Namespace: System.Windows.Forms
  Assembly: System.Windows.Forms (system.windows.forms.dll)

Name	Description
Application.Run ()	Begins running a standard application message loop on the current thread, without a form.
Application.Run (ApplicationContext)	Begins running a standard application message loop on the current thread, with an ApplicationContext.
Application.Run (Form)	Begins running a standard application message loop on the current thread, and makes the specified form visible. Supported by the .NET Compact Framework.

Dohvaćanje poruka u .NET 2.x C# aplikaciji

- Metode klase System.Windows.Forms.Application su:
  - Run metoda pokreće petlju poruka aplikacije za trenutni thread i prikazuje formu.
  - Exit ili ExitThread zaustavlja petlju poruka.
  - DoEvents obrađuje poruke.
  - AddMessageFilter dodaje filter poruka.
  - IMessageFilter omogućava zaustavljanje nekog *eventa* ili obavlja posebne operacije prije pozivanje *event handler-a*.
- Ne može se kreirati instanca ove klase jer su metode deklarirane kao statičke.
- Konzolna aplikacija ne treba petlju poruka pa ne treba ni Application objekt.

Dohvaćanje poruka u .NET 3.x C# aplikaciji

- System.Windows.Application (.NET 3.x) je osnovna klasa WPF aplikacije koja pruža sljedeće servise aplikaciji:
  - Upravljanje životnim ciklusom aplikacije (Run metoda pokreće petlju poruka, Shutdown metoda zaustavlja izvođenje aplikacije, ShutdownMode, Activated, Deactivated, DispatcherUnhandledException, Exit, SessionEnding, Startup, Current).
  - Upravljanje resursima i prozorima aplikacije (StartupUri, MainWindow, Windows, Properties, Resources, FindResource, GetContentStream, GetResourceStream, LoadComponent).
  - Parametri komandne linije i izlazni kodovi aplikacije.
  - Upravljanje navigacijom (FragmentNavigation, LoadCompleted, Navigated, Navigating, NavigationProgress, NavigationStopped, NavigationFailed, SetCookie, GetCookie).

Dohvaćanje poruka u .NET 3.x C# aplikaciji

- Za razliku od klase Application iz .NET 2.x klasa Application iz .NET 3.x više nije statička (ili članovi klase nisu statički) pa je u aplikaciji potrebno instancirati klasu Application (ili izvedenu iz klase Application).
- To se obično radi kroz XAML kôd definiranjem XAML taga što je ekvivalent instanciranju klase.

Prevođenje poruka u događaje

- Klasa Form sadrži metodu WndProc (protected override void WndProc(ref Message m)) koju nasljeđuje iz klase ContainerControl. Ta metoda predstavlja proceduru za obradu poruka promatranog prozora.
- Implementacija te metode dobivenu poruku prevodi u podizanje odgovarajućeg događaj.
- Jedna poruka se može prevesti u jedan ili u više događaja. Više poruka se može prevesti u jedan događaj.

Prevođenje poruka u događaje

```
protected override void WndProc(ref Message m)
    {
       //Poruka označena simboličkom konstantom
       WM_LBUTTONDOWN ima brojčanu vrijednost
       heksadecimalno 0x0201 tj. dekadski 513
       if (m.Msg == 513)
          MessageBox.Show(this, "WM_LBUTTONDOWN
                poruka");
       //pozivanje temeljne metode
       base.WndProc(ref m);
    }
```

Prevođenje poruka u događaje

- Prevođenje poruka u događaje u .NET 3.x nije više dohvatljivo programeru, tj. ne možemo više dohvatiti metodu u klasi Window koja ustvari prevodi poruke u događaje koji se podižu na prozoru.