Windows Services i Delphi

DAPUG ERFA-møde d. 20. juni 2012

Thomas Vedel
Thomas Vedel Consult

http://www.veco.dk
thomas@veco.dk

Hvad er en Windows Service?

- Et program der kører i baggrunden
- Ikke kræver brugerbetjening under kørslen
- Kan starte automatisk når Windows starter op (uden der er brugere logget på maskinen)
  - Anvender typisk brugerkonto SYSTEM, LOCAL SERVICE eller NETWORK SERVICE
- Kan også startes og stoppes manuelt (af brugere med administrator-rettigheder)

Hvad er en Windows Service? (2)

- Skal installeres i ”Services Manager” inden den kan køre
- Er (næsten helt almindelige) exe-filer uden brugerflade
- Optræder som selvstændige processer i Joblisten
- KAN også laves som DLL-filer, der afvikles af svchost.exe

Resten er demo...

- Men assisteret af en hel del slides og lidt ”tag med hjem source” som huskehjælp... :)

Trin 1: Opret projekt

- File → New → Delphi Projects → Service Application
- Sæt passende Properties:
  - **Name**: Navnet som servicen bliver genkendt på af Windows. Unikt, uden spaces!
  - **DisplayName**: Navn, som man ønsker det skal optræde i Services Manager (kontrolpanel → tjenester)
  - **Dependencies**: Tilføj evt. afhængigheder til andre services som kræves startet først
  - **ServiceType**: Mest **stWin32** (også i 64 bit Windows)
  - **Interactive**: Altid **False** (Ja, **ALTID!**)

Trin 1: Opret projekt (fortsat...)

- Properties (fortsat...)
  - **Password** / **ServiceStartName**: Brugerkonto man ønsker servicen skal køres under.
    - Sættes ikke; kan specificieres i Services Manager
  - **LoadGroup**: Udfyldes kun hvis man laver en sammenhængende gruppe af services
  - **AllowPause**: Med fordel = **False**
    - Giver simplere programkode (ressourcehåndtering)
  - **AllowStop**: Altid **True** (i hvert fald så længe man debugger, og det er **MEGET lidt "pænt"** at have den False)
  - På XE2: Vælg **TargetPlatform** (32 el. 64 bit) og gem projektet

Trin 2: Install / Uninstall

- Services skal installeres inden de kan bruges
  - Installeres med /INSTALL
  - Afinstalleres med /UNINSTALL
  - Hvis service er ”stoppet” i Serivces Manager kan servicen overskrives med ny version (= buildes i delphi) uden krav om /UNINSTALL og /INSTALL
  - Service genkendes af Windows på **Name** og **DisplayName**, så disse properties **ændres aldrig** under udviklingen, hvis servicen er installeret

Trin 2: Install / Uninstall (fortsat...)

- Test (den ”tomme” udgave af) servicen
  - Build projektet
  - Sæt runtime parameter til ”/INSTALL”
    - Run → Parameters → Parameters
  - Kør service
  - Tjek at den dukker op i Services Manager
  - Prøv at starte / stoppe servicen
  - Når den kører kan den ses i **Jobliste**

Trin 3: Tilføj beskrivelse

- Services Manageren kan vise en længere beskrivende tekst for servicen. Den skal vi have på!
- Beskrivelsen – og mange andre oplysninger om en service – gemmes i registry. Ja, det er ***Windows Services*** :)
- Opret beskrivelse på **AfterInstall** event
  - Afinstaller / installer for at se at det virker...

```
procedure TMinDAPUGService.ServiceAfterInstall(Sender: TService);
begin
  with TRegistry.Create(KEY_READ or KEY_WRITE) do
    try
      RootKey := HKEY_LOCAL_MACHINE;
      if OpenKey('\SYSTEM\CurrentControlset\Services\' + Self.Name, False) then
        WriteString('Description', 'Denne service kan alt - dog ikke lægge æg');
    finally
      Free;
    end;
end;
```

Trin 4: Tilføj funktionalitet (1)

- Den kode som servicen skal udføre tilføjes på **OnExecute** eller **OnStart** event.
- **OnExecute** event
  - Kører i servicens (i forvejen oprettede) tråd
  - Simpel kode: Du behøver ikke oprette ny tråd
  - Pause / Resume håndteres automatisk
  - Et loop skal tage KORT TID (max. et par sekunder)

Trin 4: Tilføj funktionalitet (2)

- **OnStart** event
  - Fyrer inden **OnExecute**
  - Beregnet til initialisering (det kunne f.eks. være oprettelse af passende antal tråde hvis man har komplekse eller langvarige opgaver)
  - Stor fleksibilitet
  - Håndtering af Pause / Resume er bøvlet
    - Frigivelse af ressourcer
    - Genstart af tråde

Trin 4: Tilføj funktionalitet (3)

- Hvor skal man så placere sin kode?
  - Brug så vidt muligt **OnExecute** og lav kortvarige loops. Det er på alle måder ”pænt” og nemmest at håndtere.

```
procedure TMinDAPUGService.ServiceExecute(Sender: TService);
const SecBetweenRuns = 10;
var Count: Integer;
begin
  Count := 0;
  while not Terminated do
  begin
    Inc(Count);
    if Count >= 10*SecBetweenRuns then
    begin
      Count := 0;
      KaldDinKodeSomGørNogetIAndenUnit; // Det er her det sker...
    end;
    Sleep(100);
    ServiceThread.ProcessRequests(False);
  end;
end;
```

Trin 4: Tilføj funktionalitet (4)

- I mere komplekse tilfælde: Brug OnStart
  - Tilføj ”Thead Object” modul til services projektet
  - Definer thread variabel i servicens **private** sektion
  - Create tråden, initialiser den og start den i OnStart eventen
  - Lav kode som terminerer tråden, venter på at tråden kører færdig og frigiver den i OnStop og OnShutdown eventene (lav én fælles procedure som kaldes begge steder fra)
  - Smid koden ”som gør det vi gerne vil” over i særskilt unit (som på foregående side)
  - Sæt servicens ”AllowPause” property til false...

Debugging

- En service kan (næsten) umuligt debugges...
  - Det er derfor vi smider koden ”som gør det vi gerne vil have” over i særskilt unit som vi KAN debugge ved at bruge unit'en i et helt almindeligt projekt.
  - Brug CodeSiteLogging (inkluderet i XE2) til at logge fejlsituationer fra kørende services (i produktion).
    - TService har en (ubrugelig) LogMessage metode
    - Brug i stedet CodeSiteLogging.
      - Hvordan man logger fra services, som jo ikke er ”helt normale programmer” er beskrevet i CodeSiteLogging hjælpen
      - LogError, LogEvent, LogWarning sender til Windows Event Log

Hvad med brugerflade???

- Ofte vil man have brug for at konfigurere en service, men den kan ikke have en brugerflade
  - Lav en kontrolpanel applet til styring af servicen – men den kan lige som servicen ikke debugges...
  - Tilføj derfor en helt normal TForm til applet'en og put al funktionalitet i denne form, så den kan testes i stand alone projekt
  - De facto standard: Alle parameter som skal bruges af en service gemmes i... registry!
  - Lyder snørklet? det er det ikke (særlig meget) :)

Kontrolpanel applet how-to

- Opret kontrolpanel applet projekt i Delphi (32 bit til 32 bit service, 64 bit til 64 bit service)
- Sæt følgende properties (og lad resten være)
  - Name: Programnavn som Windows genkender apletten på. Bliver vist som ”Procesnavn” i Jobliste
  - Caption: Den (meget korte) beskrivelse / det navn på appletten som vises i kontrolpanelet.
- Tilføj en helt almindelig TForm til appletten og kald formens ShowModal metode fra applettens OnActivate event. (Pænest: Create formen dynamisk fra OnActivate eventen)
- That's it!

Overførsel af parametre

- Kontrolpanel applettens brugerflade laves så den passer til de ting man vil kunne styre i serivcens opsætning
- Alle parametre gemmes / hentes fra registry i en undernøgle til servicen med navnet ”Parameters”
- Forslag: Tilføj mulighed for at starte / stoppe servicen samt se ”hvordan den har det.”

ServicesUtils ”Hjælpe-unit”

- Jeg har lavet en ”hjælpe-unit” med rutiner til at
  - Overføre parametre til/fra en service via registry
  - Starte / stoppe en service
  - Tjekke status for en service (kørende, stoppet, ikke installeret osv.)
- Unit'en er sammen med projektskabelon vedlagt som bilag til denne præsentation.

Links

- Mere info om servicesprogrammering i Delphi (en del info er temmelig ”bedaget”...)
  - Services Tutorial:
    http://www.tolderlund.eu/delphi/service/service.htm
  - Delphi Tricks: System Information:
    http://www.delphitricks.com/source-code/systeminfo/1/
  - About Delphi:
    http://delphi.about.com/od/windowsshellapi/a/delphi-windows-service-applications.htm
  - Spørg mig :)
    thomas@veco.dk