

Créer un Service Windows avec Dotnet

23/05/2003

Par Elise Dupont

niveau : facile
durée : de 15 à 25 minutes

Avant Propos :

Un service Windows est une application qui tourne en "tâche de fond" afin d'effectuer par exemple des opérations à un même intervalle de temps. Cela permet de lancer une application au démarrage sans avoir besoin d'une interface graphique. Dans ce petit cours nous apprendrons que créer un Service Windows avec Dotnet est très facile et rapide. Il suffit de connaître quelques astuces, principalement pour l'installation du service une fois créé.

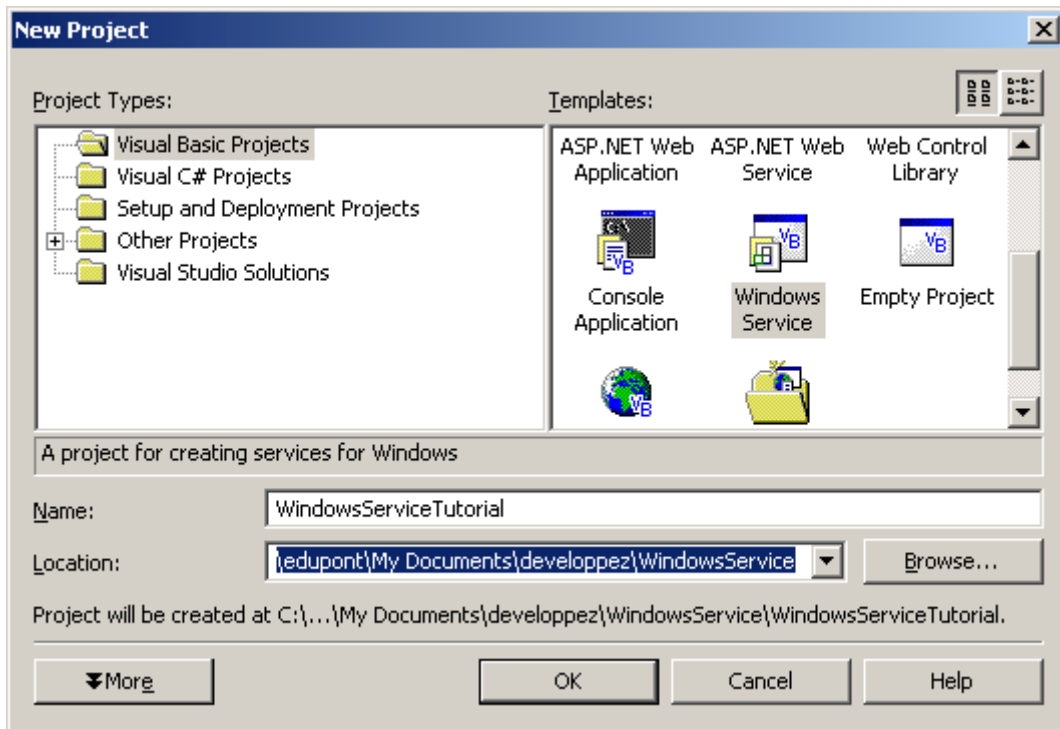
Je vous conseille vivement de télécharger les Codes Sources. La [version VB.NET](#) est là, et [version C#](#) est ici. Ne vous en privez pas, c'est gratuit et ca vous aidera à suivre ce tutoriel.

Sommaire:

1. Créer le projet
2. Gérer les erreurs sans interface
 - 2.1. Ecrire dans le journal des événements
 - 2.2. Envoi automatique de mails
3. Intégrer un fichier de configuration c'est possible !
4. Installer le service

1. Créer le projet

Visual Studio .NET facilite la tâche : il vous suffit de créer un nouveau projet de type "Service Windows". Une partie du code est générée à la création. Il vous faudra tout de même rajouter quelques paramètres, mais cela reste très simple.



En regardant le code qui a été généré, vous pouvez observer qu'une classe Service a été créée, et qu'elle hérite de System.ServiceProcess.ServiceBase. Vous devrez mettre le contenu du code que doit effectuer votre service (fournir un serveur remoting, vérifier quotidiennement le contenu d'une base de données, peu importe) dans la méthode OnStart() qui sera exécutée au démarrage du service.

De même, si vous avez besoin de faire quelques opérations avant de quitter le service, il faudra les mettre dans la fonction OnStop().

VB.NET :

```
Public Class MonService
Inherits System.ServiceProcess.ServiceBase

    Protected Overrides Sub OnStart(ByVal args() As String )
        'initialiser les compteurs des variables d 'application
        'Code lancé au démarrage du service
        'faire une action
        Try
            Maclasse.MaMethode()
        Catch ex As Exception
            'écrire l'erreur dans l'event log de la machine
        End Try
    End Sub

    Protected Overrides Sub OnStop()
        'Code qui sert à effectuer des operations de "fermeture" avant la fin du service
        MaClasse.Finir()
    End Sub
End Class
```

C#:

```
public class MonService : System.ServiceProcess.ServiceBase
{
    static void Main() //Code lancé au démarrage du service
    {
        try
        {
            Maclasse.MaMethode();
        }
        catch (Exception ex)
```

```

    {
        //écrire l'erreur dans l'event log de la machine
    }
}
protected override void OnStop()
{
    //effectuer des operations de fermeture avant la fin du service
    MaClasse.Finir();
}
}
End Class

```

Ensuite il vous faut rajouter quelques paramètres afin d'identifier le Service de façon unique, surtout lors de l'installation. Dans la méthode Shared Sub Main(), il vous faut spécifier le nom du Service ainsi :

```

Public Class MonService Inherits System.ServiceProcess.ServiceBase
    <MTAThread()> Shared Sub Main()
        'Donnez un nom à votre service
        ServiceName = "Mon premier Service Windows"
    End Sub
End Class

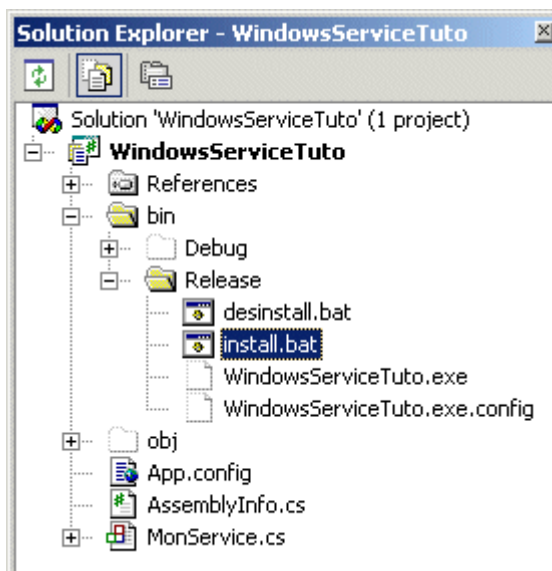
```

```

public class MonService : System.ServiceProcess.ServiceBase
{
    public MonService()
    {
        InitializeComponent();
        ServiceName = "MonServiceWindowsCSharp"; //Donnez un nom à votre service
    }
}

```

Voilà, vous avez construit votre Service Windows. Mais si vous essayez de l'installer comme ça, vous aurez une erreur vous disant qu'il manque un Installateur. Voici à quoi devrait ressembler votre projet :



2. Gérer les erreurs sans interface

Comme toute application normale, vous voulez être avertis en cas d'erreur. Sauf qu'avec un Service Windows il n'y a pas d'interface. Vous avez le choix entre deux alternatives :

- Ecrire dans le journal d'événements (Event log)
- Envoyer un mail à un administrateur (en spécifiant l'email en paramètre par exemple pour que le changement soit simple)

L'un dans l'autre, c'est très facile à implémenter.

2.1. Ecrire dans le journal des événements

C'est très simple. Dans votre Catch, vous ajoutez cette fonction :

VB.NET :

```
EventLog.WriteEntry("Mon premier ServiceWindows", ex.Message, EventLogEntryType.Error ,
15)
```

C# :

```
EventLog.WriteEntry("Mon premier ServiceWindows", ex.Message, EventLogEntryType.Error,
15);
```

Ceci écrira une nouvelle ligne dans le journal d'événements.

2.2. Envoi automatique de mails

Vous pouvez aussi envoyer un mail à chaque erreur. Ce qui permet d'être informé de suite. Vous allez vous rendre compte que c'est très simple.

- Ajoutez une référence System.Web
- Importez System.Web.Mail
- Créez une fonction FaireMail()
- Appelez-la depuis le catch

La fonction ressemblera à ça :

VB.NET :

```
Public Function FaireMail(ByVal message As String )
    'Le mail expéditeur sert juste d'indication, mais en fin de compte, même si vous
mettez
    'un mail invalide ou qui n'existe pas en expéditeur, cela fonctionne
    '(vous venez d'apprendre comment créer un logiciel qui fait des mails anonymes ;-)
```

```
    Dim from As String = "expediteur@monentreprise.com"
    Dim mailto As String = "envoyer@monentreprise.com"
    Dim sujet As String = "Message d'erreur - Ne pas repondre"
    Smtplib.Send(from, mailto, sujet, message)
```

```
End Sub
```

C# :

```
public void SendMail(string message)
{
    //Le mail expediteur sert juste d'indication, mais en fin de compte, meme si vous
mettez
    //un mail invalide ou qui n'existe pas en expediteur, cela fonctionne
    //(vous venez d'apprendre comment créer un logiciel qui fait des mails anonymes ;)
```

```

string from = "expediteur@monentreprise.com";
string mailto = "envoyer@monentreprise.com";
string sujet = "Message d'erreur - Ne pas repondre";
SmtpMail.Send(from, mailto, sujet, messagey);

}

```

3. Intégrer un fichier de configuration c'est possible !

Et surtout très simple. C'est comme pour une application classique : ajoutez un fichier du nom monapplication.exe.config, et rajoutez des éléments. Ensuite dans votre code accédez à vos éléments du fichier de configuration ainsi :

VB.NET :

```

EventLog.WriteEntry("Mon premier ServiceWindows",
System.Configuration.ConfigurationSettings.AppSettings("monMessage"),
EventLogEntryType.Information, 15)

```

C#:

```

EventLog.WriteEntry("Mon premier ServiceWindows",
System.Configuration.ConfigurationSettings.AppSettings["monMessage"],
EventLogEntryType.Information, 15);

```

4. Installer le service

Avant de l'installer à proprement parler, il va vous falloir rajouter une classe qui vous permettra d'installer le service. Bien que cela ne soit pas suffisant comme vous allez le voir par la suite. Vous devez donc rajouter une classe comme ceci :

VB.NET :

```

<RunInstallerAttribute(True )> Public Class MyProjectInstaller
Inherits Installer
' Cette classe permet de pouvoir compiler le service afin d'en obtenir un binaire
' Cependant, ce n'est pas suffisant pour installer le service windows.
' Ceci permet juste d'avoir un .exe
' Il faudra une étape supplémentaire pour installer le service sur votre machine
Private monServiceInstaller As ServiceInstaller
Private monProcessInstaller As ServiceProcessInstaller
Public Sub New ()
' Instancie les installeurs
monProcessInstaller = New ServiceProcessInstaller()
monServiceInstaller = New ServiceInstaller()
' Le service sera lancé sous le compte Système
monProcessInstaller.Account = ServiceAccount.LocalSystem
' Le service sera démarré manuellement
monServiceInstaller.StartType = ServiceStartMode.Automatic
' Le nom du service doit être égal au nom de la classe ServiceBase dont
' on dérive (voir le PublicSubNew() de la classe MonService pour comprendre)
monServiceInstaller.ServiceName = "Mon premier ServiceWindows"
' Ajouter les installeurs à la collection (l'ordre n'est pas important)
Installers.Add(monServiceInstaller)
Installers.Add(monProcessInstaller)

End Sub
End Class

```

C#:

```

[RunInstaller(true )]

```

```

public class MyProjectInstaller: Installer
{
    public MyProjectInstaller() :base ()
    {
        //Cette classe permet de pouvoir compiler le service afin d'en obtenir un
        //Cependant, ce n'est pas suffisant pour installer le service windows.
        //Ceci permet juste d'avoir un .exe
        //Il faudra une étape supplémentaire pour installer le service sur votre machine
        //Instancie les installeurs
        ServiceInstaller monServiceInstaller = new ServiceInstaller();
        ServiceProcessInstaller monProcessInstaller = new ServiceProcessInstaller();
        // Le nom du service doit être égal au nom de la classe ServiceBase dont on
        // (voir le Public Sub New() de la classe MonService pour comprendre)
        monServiceInstaller.ServiceName = "MonServiceWindowsCSharp";
        monServiceInstaller.DisplayName = "Mon ServiceWindows CSHARP";
        // Ajouter les installeurs à la collection (l'ordre n'est pas important)
        this .Installers.Add(monServiceInstaller);
        // Le service sera lancé sous le compte Système
        monProcessInstaller.Account = System.ServiceProcess.ServiceAccount.LocalSystem;
        // Ajouter les installeurs à la collection (l'ordre n'est pas important)
        this .Installers.Add(monProcessInstaller);
    }
}

```

Tout est expliqué dans le code, je ne pense pas qu'il faille le répéter ici.

L'installation à proprement parler suit les étapes suivantes :

- Compilez votre projet pour obtenir les fichiers binaires et exécutables
- Placer les bin dans un répertoire du genre "c:\program files\Monservice"
- Lancer une invite de commande .NET (à ne pas confondre avec une invite de commande DOS toute simple)
- Utiliser l'outil installutil.exe (sous le répertoire d'installation du framework)

La commande est très simple : **>installutil -i monservice.exe** (Voir les fichiers batch que j'ai ajouté dans le code source à télécharger)

```
Visual Studio .NET Command Prompt
Setting environment for using Microsoft Visual Studio .NET tools.
(If you also have Visual C++ 6.0 installed and wish to use its tools
from the command line, run vcvars32.bat for Visual C++ 6.0.)

C:\>cd "C:\Documents and Settings\edupont\My Documents\developpez\WindowsService\WindowsServiceTutorial\bin"

C:\Documents and Settings\edupont\My Documents\developpez\WindowsService\WindowsServiceTutorial\bin>installutil -i WindowsServiceTutorial.exe
Microsoft (R) .NET Framework Installation utility Version 1.0.3705.288
Copyright (C) Microsoft Corporation 1998-2001. All rights reserved.

Running a transacted installation.

Beginning the Install phase of the installation.
See the contents of the log file for the c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe assembly's progress.
The file is located at c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.InstallLog.
Installing assembly 'c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe'.
Affected parameters are:
i =
logfile = c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.InstallLog
assemblypath = c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe
Installing service Mon premier ServiceWindows...
Service Mon premier ServiceWindows has been successfully installed.
Creating Eventlog source Mon premier ServiceWindows in log Application...

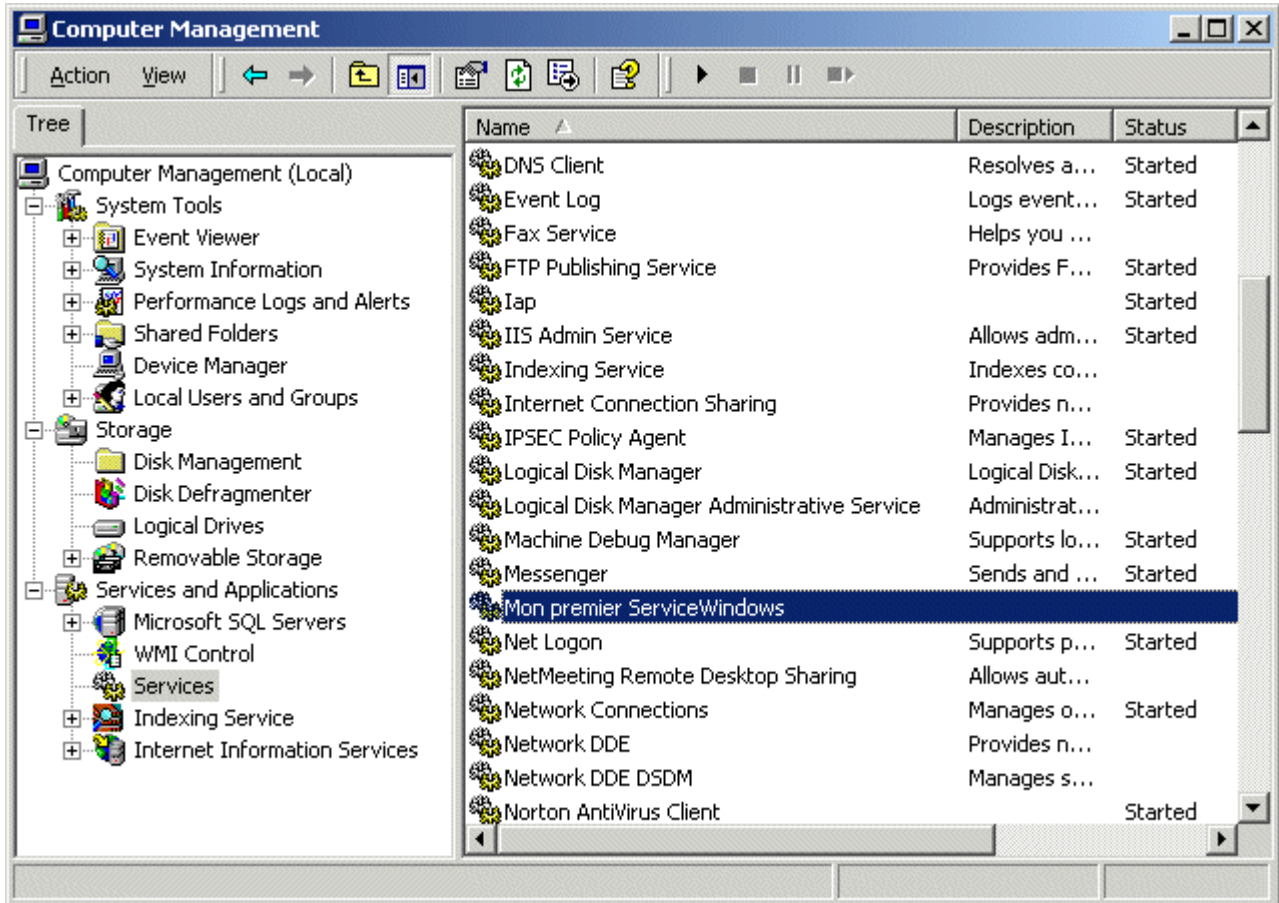
The Install phase completed successfully, and the Commit phase is beginning.
See the contents of the log file for the c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe assembly's progress.
The file is located at c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.InstallLog.
Committing assembly 'c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe'.
Affected parameters are:
i =
logfile = c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.InstallLog
assemblypath = c:\documents and settings\edupont\my documents\developpez\windowsservice\windowsservicetutorial\bin\windowsservicetutorial.exe

The Commit phase completed successfully.

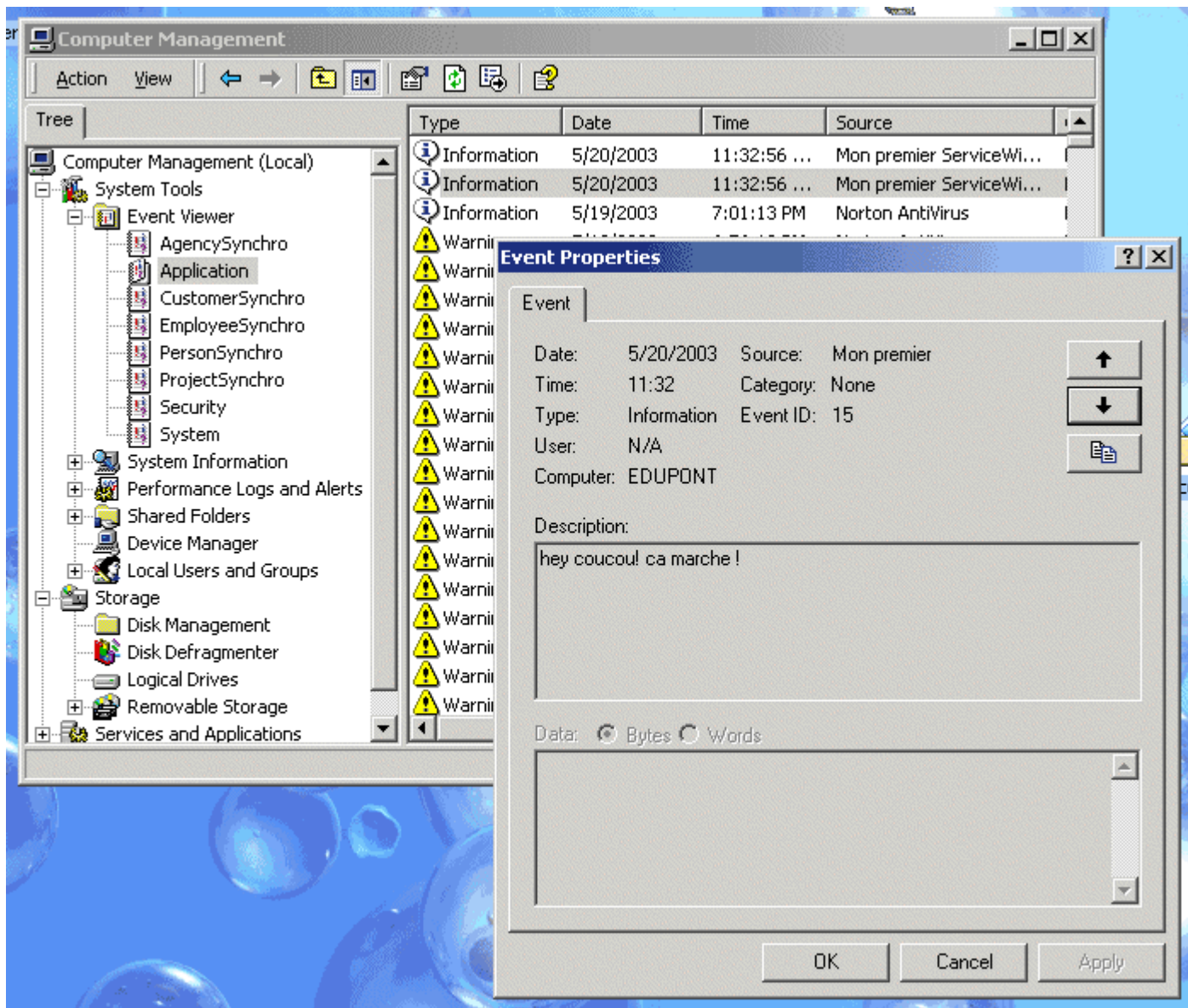
The transacted install has completed.

C:\Documents and Settings\edupont\My Documents\developpez\WindowsService\WindowsServiceTutorial\bin>_
```

Lancer le gestionnaire de service et démarrer le service. Si le lancement échoue, un message d'erreur apparaît.



Regarder le journal d'événements pour voir le résultat du démarrage et du message :



Ca y est, votre service fonctionne. Je vous conseille vivement de télécharger les Codes Sources, voir soi-même et faire des essais vaut mieux que tous les cours. La [version VB.NET](#) est là, et [version C#](#) est ici.



L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé ailleurs sans autorisation préalable

elise.dupont@europe.com - www.dotnet-tech.com - 2003