Introduction

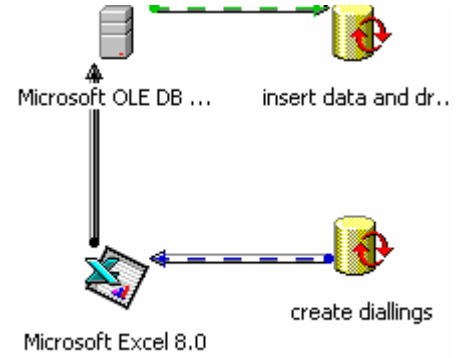
DTS ou « Data Transformation Service » est un outil intégré à SQL Server depuis sa version 7. Cet outil permet de modéliser graphiquement votre process de transformation de données.
Par process de transformation, j'entends :

- Importation de données
- Exportation de données
- Transformation de données

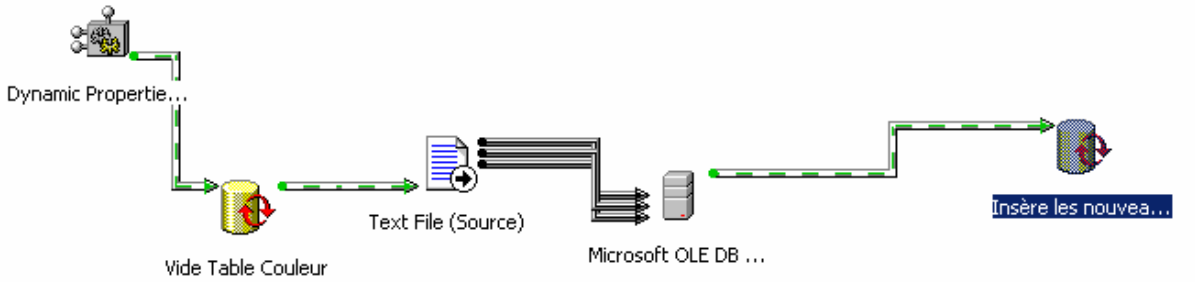
Globalement, il vous suffit d'associer une source à une destination pour pouvoir importer/exporter/transformer des données.

Par exemple, voici un process permettant :

1. De générer un fichier excel à partir d'une requête SQL
2. D'importer le fichier Excel dans une base SQL
3. D'effectuer différents calculs métiers sur cette même base



Pour les besoins de notre exemple, nous allons nous appuyer sur un DTS quelque peu différent :



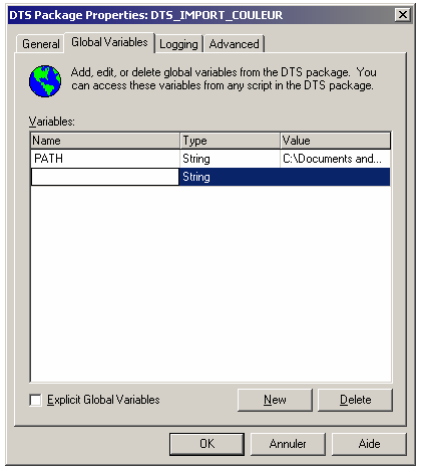
Concrètement la seule différence fondamentale par rapport au DTS précédent est l'emploi de propriétés dynamiques. La possibilité d'utiliser des propriétés dynamiques est l'une des nouvelles fonctionnalités de SQL Server 2000. Cette fonctionnalité permet de définir à la volée les valeurs des

propriétés des objets DTS. Dans ce cas précis, la valeur de la propriété « datasource » du fichier texte source est définie et affecté au moment où le DTS est appelé.

Créer une propriété dynamique

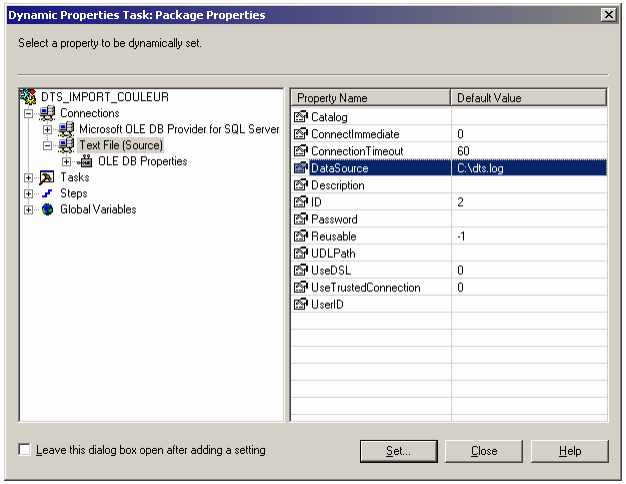
Pour rendre dynamique une propriété, il vous suffit de respecter le process suivant :

1-Ajouter une variable globale au DTS (clique droit sur l'espace de travail puis propriétés du DTS)



2-Ajouter une tache de propriété dynamique

3-Sélectionnez la propriété que vous souhaitez rendre dynamique



4-Affectez cette dernière à la variable globale créée précédemment

Et voilà, vous pouvez désormais modifier à la volée la datasource du fichier texte source.

Appeler un DTS à travers une application C#

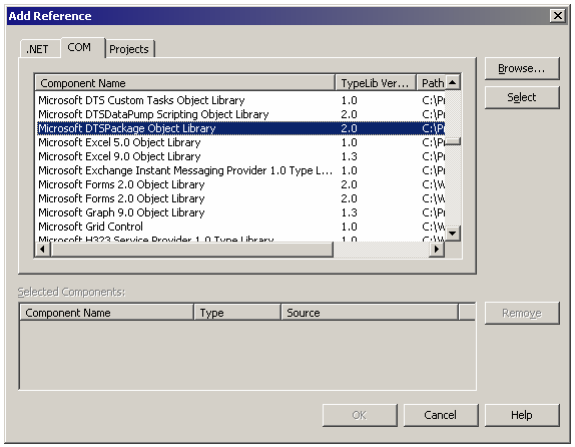
Pré-requis

Avant de pouvoir appeler un DTS à travers C#, il est impératif de référencer le composant COM non managé « Microsoft DTSPackage Object Library » au sein de votre projet Visual Studio .NET.

Vous le trouverez via le menu Projet / Ajouter une référence/

Note
Cette DLL est livré lors de l'installation du client SQL Server.

Une fois cette référence ajoutée, vous pouvez désormais l'utiliser au sein de votre projet.



But de la classe

Au même titre que le Data Access Application Block de Microsoft, la classe DTSLauncher a pour objectif de faciliter et d'encapsuler l'ensemble des appels aux DTS placés dans le repositery SQL Server.
En cas d'erreur le message d'erreur ainsi que l'étape à l'intérieure de laquelle l'erreur est survenue nous seront retournés.

Propriétés

Une fois la DLL référencée, nous allons définir les différentes méthodes dont nous allons avoir besoin. Le tableau ci-dessous référence chacune d'entres-elles :

Propriété	**Type**	**Description**
`ServerName`	`String`	`Nom du serveur SQL`
`UserName`	`String`	`Login SQL Server`
`PackageName`	`String`	`Nom du package a exécuter`
`FileID`	`String`	`Nom`

`DynamicsParameters`	`Hashtable`	`Collections contenant les différents paramètres dynamiques nécessaires au DTS. Exemple :`
		`ht = new Hashtable();`
		`ht.Add(<Nom de la variable globale >,<Valeur de la variable>);`

		soit : ht.Add("PATH",@"C:\Temp\couleur2.csv"); DTS.DynamicsParameters = ht;
Password	String	Password SQL Server
Path	String	Répertoire ou se trouve le fichier

Méthodes

Méthode	Valeur renvoyée	Description
public string LaunchDTS()	"OK" dans le cas où le DTS s'est achevé avec succès, le message d'erreur et l'étape dans le cas contraire.	Initialise, exécute le DTS. Logg les erreurs en cas d'échec d'une des étapes.
private void PrepareDTSPackage()	N/A (procédure)	Charge le DTS en mémoire et affecte les valeurs des différentes variables

Appel

```
//Instanciation de la classe
    Harpoutlian.Common.Data.DTSLauncher DTS = new DTSLauncher();
//Nécessaire auxvariables
System.Collections.Hashtable ht = new Hashtable();
//Ajout des variables globales
    ht.Add("PATH",@"C:\Temp\couleur2.csv");

    //Nom du DTS
    DTS.PackageName  = "DTS_IMPORT_COULEUR";
//Server SQL
    DTS.ServerName = "LOCALHOST";
//Login SQL
    DTS.UserName = "sa";
//Pwd
    DTS.Password = "sa";
    //Variables dynamiques
DTS.DynamicsParameters = ht;
    //Exécution du DTS
string msg = DTS.LaunchDTS();
    //Si le message est différent de OK alors un pb est survenue au cours de l'éxécution
if(msg<>"ok")
    {
```

```
        Response.Write("Erreur:" & msg);
      }
```

Et voilà! Vous trouverez le code de la classe DTSLauncher ci-dessous.

Code

```
using System;

namespace Harpoutlian.Common.Data
{
   /// <summary>
   /// Summary description for DTSLauncher.
   /// </summary>
   public class DTSLauncher
   {

      #region Private Members
      private DTS.Package _DTSPackage;
      private string _serverName, _userName,_password, _packageName, _path;
      private System.Collections.Hashtable _dynamicsParameters;
      private System.Int32 _fileID;
      #endregion

      #region Constructor
      public DTSLauncher()
      {
         //
         // TODO: Add constructor logic here
         //
      }
      #endregion

      #region Accesseur
      public string ServerName
      {
         get{return _serverName;}
         set{_serverName = value;}
      }
      public System.Collections.Hashtable  DynamicsParameters
      {
```

```
    get{return _dynamicsParameters;}
    set{_dynamicsParameters = value;}
}
public string UserName
{
    get{return _userName;}
    set{_userName = value;}
}
public string Password
{
    get{return _password;}
    set{_password = value;}
}
public string PackageName
{
    get{return _packageName;}
    set{_packageName = value;}
}
public string Path
{
    get{return _path;}
    set{_path= value;}
}
public System.Int32  FileID
{
    get{return _fileID;}
    set{_fileID= value;}
}


#endregion

#region Public Methods
public string LaunchDTS()
{
    int ErrorNumber, HelpNumber;
    _DTSPackage = new DTS.Package();
    string DTSStepExecResult_Failure;
    string msg="";
    string Source, Desc,Help, Interface;
    PrepareDTSPackage();
    try
```

```
    {
       _DTSPackage.Execute();
       msg= "OK";
    }
    catch
    {
       foreach (DTS.Step DTSStep in _DTSPackage.Steps)
       {
          DTSStepExecResult_Failure = DTSStep.ExecutionResult.ToString();
          if(DTSStepExecResult_Failure=="DTSStepExecResult_Failure")
          {
             DTSStep.GetExecutionErrorInfo(out ErrorNumber,out Source ,out Desc,out Help,out HelpNumber,out Interface);
             if(ErrorNumber!=0)
             {
                msg  +="Step " + DTSStep.Name + ": " + DTSStep.Description;
                msg += "Error: " + ErrorNumber + " ";
                msg  +="Source: " + Source + " ";
                msg += "Description: " + Desc + " ";
              }
          }

       }
    }
    finally
    {
       try
       {
          _DTSPackage.UnInitialize();
          System.Runtime.InteropServices.Marshal.ReleaseComObject(_DTSPackage);
          _DTSPackage= null;
       }
       catch(System.Runtime.InteropServices.COMException e)
       {
          //msg = _packageName + " " + _fileID + " " + e.Source.ToString();
       }
    }
    return msg;
}


#endregion
```

```
    #region Private Methods
    private void PrepareDTSPackage()
    {
      object pVarPersistStgOfHost = null;
      _DTSPackage.FailOnError = true;
      _DTSPackage.LoadFromSQLServer
        (
          _serverName,
          _userName,
          _password,
          DTS.DTSSQLServerStorageFlags.DTSSQLStgFlag_Default,
          null,
          null,
          null,
          _packageName,
          ref pVarPersistStgOfHost
        );

        foreach(System.Collections.DictionaryEntry entry in _dynamicsParameters)
      {
        _DTSPackage.GlobalVariables.Remove((string)entry.Key);
        _DTSPackage.GlobalVariables.AddGlobalVariable( (string)entry.Key,  (string)entry.Value);
      }

  }
    #endregion

  }
}
```

Article de :

Cyril Harpoutlian
Consultant - MCP ASP.NET
http://www.harpoutlian.com
charpoutlian@hotmail.com

Publié sur :

http://www.c2i.fr
Le 1er portail francophone sur Microsoft .NET.