

Comment utiliser Windows Desktop Search au sein de vos applications

par [Louis-Guillaume MORAND \(Page perso de Louis-Guillaume MORAND\)](#)

Date de publication : 10 octobre 2008

Dernière mise à jour :

Dans cet article, vous allez apprendre comment utiliser Windows Desktop Search au sein de vos applications afin de fournir des fonctionnalités de recherches avancées à vos utilisateurs.

Introduction.....	3
1 - Utilisations possibles de Windows Desktop Search.....	3
2 - Utilisation d'ADO.Net.....	4
3 - Implémentation de ISearchQueryHelper.....	5
4 - Pour aller plus loin.....	6
4.1 - La recherche fulltext.....	6
4.2 - Les dates et heures.....	6
4.3 - Les regroupements.....	7
4.4 - Connaître la version de WDS installée.....	7
Conclusion.....	7
6 - Téléchargements.....	7

Introduction

Il est de plus en plus courant qu'un utilisateur lambda, comme vous ou moi, stocke un nombre considérable d'informations sur différents systèmes de stockage et en particulier les disques durs. Ces disques durs ont une capacité toujours plus grande et le nombre de fichiers peut rapidement devenir excessif pour pouvoir s'y retrouver rapidement. Il va donc forcément arriver un jour où vous allez chercher un fichier sur votre disque dur et où la recherche Windows mettra plusieurs minutes pour trouver ledit fichier. Différentes entreprises ont alors amené l'indexation (ce qu'utilisent les gros moteurs de recherche Internet) au sein de l'ordinateur du particulier; Microsoft l'a fait grâce à son outil Windows Desktop Search. Ce dernier peut être installé sur Windows XP et l'est déjà par défaut au sein de Windows Vista.

Nous allons voir au sein de cet article comment nous pouvons utiliser cette outil dans vos applications quelles soient personnelles ou professionnelles.


1 - Utilisations possibles de Windows Desktop Search

D'un point de vue contextuel, plusieurs raisons peuvent vous pousser à utiliser WDS (Windows Desktop Search) au sein d'une application. Cela peut tout d'abord servir pour retrouver un certain nombre de fichiers à utiliser ou alors, proposer à l'utilisateur de retrouver rapidement un fichier particulier pour agir avec ce dernier. Bien entendu, ces recherches de fichiers peuvent se faire via un certain nombre de paramètres comme le titre, la date du fichier et bien d'autres choses que la recherche par défaut de Windows ne permet pas toujours.

Il existe un certain nombre de façons d'utiliser WDS.

1- Le protocole search-ms. Comme le protocole internet HTTP, il permet avec une seule ligne de faire une recherche au sein de Windows Explorer. Vous pouvez ainsi cliquer sur Démarrer > Exécuter et saisir


```
search-ms:query=plage&crumb=folder:C:\MesPhotos&
```

Ce qui lancera une recherche de tous les fichiers ayant utilisant le mot "plage" et étant situés dans le dossier MesPhotos. Pour plus d'infos sur le protocole search-ms, cliquez  [ici](#).

2- En utilisant la syntaxe SQL Windows Search qui ne permet que de faire des sélections et des regroupements mais aucune modification. La syntaxe ressemble à du T-SQL et permet d'interroger les champs des tables d'indexation. En voici un exemple :

```
SELECT System.ItemName FROM systemindex WHERE CONTAINS("System.FileName", "*plage*")
```

Pour plus d'informations, cliquez  [ici](#).

3- En utilisant Advanced Query Syntax. AQS est un peu comme le SQL vu précédemment mais plus puissant puisqu'il contient des agrégats permettant de simplifier énormément les recherches. Ces facilités d'usage sont nombreuses ( [cf la documentation](#)). Il est ainsi possible de rechercher des photos contenant le mot plage, sans retourner les autres fichiers se nommant "*plage*". Pour cela, il suffit de filtrer les résultats en utilisant le filtre : "kind:pics". Il est possible avec AQS de faire des requêtes très complexes et très fines

4- En utilisant OLE DB à travers ADO.Net, ce que nous allons voir en détail par la suite.

5- Et enfin en implémentant l'interface ISearchQueryHelper que nous allons également voir en détail.

Toutes ces méthodes sont utiles mais les deux utilisables de façon efficace et rapide au sein d'une application .Net sont les deux dernières. L'une aura pour avantage d'être très simple tandis que l'autre sera plus "propre".

2 - Utilisation d'ADO.Net

La méthode la plus simple pour interroger une source de données en .Net reste l'utilisation d'ADO.Net. C'est non seulement la plus simple mais aussi la plus couramment utilisée. Avec Windows Desktop Search, nous ne dérogerons pas à la règle et nous n'aurons que deux petites choses sur lesquelles faire attention.

Tout d'abord en choisissant un provider adapté : il s'agit du provider OleDb, et ensuite en spécifiant une chaîne de connexion particulière pour nous permettre d'interroger le gestionnaire de l'indexation de Windows Desktop Search. La chaîne de connexion est particulière à WDS et reste simple: "**Provider=Search.CollatorDSO;Extended Properties='Application=Windows';**"

Commençons par créer notre connexion puis par l'ouvrir

```
OleDbConnection connection = new  
OleDbConnection("Provider=Search.CollatorDSO;Extended Properties='Application=Windows';");  
connection.Open();
```

Puis déclarons notre objet *OleDbCommand* qui représentera notre requête auprès de l'index

```
var command = new OleDbCommand { Connection = connection, CommandText = "ma requete" };
```


Note: notez que le code précédent provient de C# 3.0 et utilise le mot-clé var ainsi que les nouveaux initialiseurs d'objets. En 2.0, il correspondrait à

```
OleDbCommand co = new OleDbCommand();  
co.Connection = connection;  
co.CommandText = "ma requete";
```

Et enfin le code de lecture du résultat

```
OleDbDataReader reader = command.ExecuteReader();  
while (reader.Read())  
{  
    // traitement de votre choix sur les résultats retournés  
    // exemple MessageBox.Show(reader.GetString(0));  
}  
reader.Close();
```

Cela n'est pas plus complexe que cela, et il est ainsi possible d'afficher très rapidement les informations de fichiers retrouvés via WDS à condition bien entendu que la requête soit correcte.


 *Windows Desktop Search est tout sauf bavard et en cas d'exception ne vous attendez pas à trouver rapidement la source du problème. Par exemple, si l'une des informations demandées (nom de colonne) était erronée, il vous renverrait une exception avec pour seul texte : **E_FAIL(0x80004005)**. Pensez donc à vérifier les noms des colonnes lorsque vous avez cette erreur. La difficulté étant de connaître ces noms de colonnes puisque malheureusement, d'une version à l'autre de Windows Desktop Search, celles-ci changent...*

Voici donc un exemple complet de recherche




```
OleDbConnection connection = new  
OleDbConnection("Provider=Search.CollatorDSO;Extended Properties='Application=Windows';");  
connection.Open();  
var command = new OleDbCommand { Connection = connection,  
CommandText = String.Format("SELECT Top 5 System.ItemUrl, System.ItemName, System.Title  
FROM systemindex WHERE CONTAINS(\"System.FileName\", \"{0}*  
\\'\", "plage") } ;  
OleDbDataReader reader = command.ExecuteReader();  
lsvSearch1.Items.Clear();
```

```
while (reader.Read())
{
    lsvSearch1.Items.Add(new ListViewItem(new[] { reader.GetString(0) }));
}
reader.Close();
```

3 - Implémentation de ISearchQueryHelper

Une autre solution, plus correcte car permettant de faire des requêtes plus avancées, est l'utilisation de l'interface *ISearchQueryHelper*. Ici, la manipulation est plus complexe car non seulement le traitement de vos requêtes ne sera pas tout à fait identique mais surtout, vous n'utiliserez plus du tout le même type de requête. Tout d'abord, nous allons "presque" en finir avec nos requêtes au format SQL Windows Search, nous allons maintenant utiliser ADS ( **Advanced Query Syntax**) qui a une syntaxe bien particulière puisqu'il s'agit de placer le caractère ":" entre la propriété du filtre et la valeur du filtre. Exemple:

```
name:plage
```

Revenons à notre interface. Pour utiliser cette interface, vous allez devoir télécharger et installer le  **SDK de Window Vista** et repérer le fichier *SearchAPI.tlb* qui se trouve dans **C:\Program Files\Microsoft SDKs\Windows\v6.0\Lib**. Une fois celui-ci trouvé, vous allez devoir utiliser  **tlbimp** afin de gérer une assembly managée à partir des informations déclarées dans le fichier *.tlb. Vous pourriez avoir besoin de télécharger le  **SDK du framework .Net** pour avoir l'outil TLBImp.

```
tlbimp.exe SearchApi.tlb /out:SearchApi.dll
```

A quoi cela va-t-il nous servir? Tout simplement à faciliter la connexion en nous fournissant la connexion adaptée, puis en transformant nos requêtes AQS en SQL.

Pour cela, nous allons commencer par importer la librairie dans votre projet puis en ajoutant le namespace correspondant:

```
using SearchAPILib;
```

Nous aurons ensuite besoin de deux objets. Tout d'abord un *CSearchManager*, qui nous servira à obtenir notre deuxième objet, un objet implémentant *ISearchQueryHelper*: un *CSearchQueryHelper*. Pour cela, deux lignes suffisent:

```
CSearchManager manager = new CSearchManagerClass();
CSearchQueryHelper helper = cManager.GetCatalog("SYSTEMINDEX").GetQueryHelper();
```

Nous allons ensuite configurer notre *CSearchQueryHelper* qui se chargera de générer une requête personnalisée pour nous grâce à sa méthode *GenerateSQLFromUserQuery()*. Commençons par limiter le nombre de résultats retournés puis précisons l'information que nous souhaitons récupérer

```
helper.QuerySelectColumns = "\"System.ItemNameDisplay\"";
helper.QueryMaxResults = 5;
```

Nous devons ensuite récupérer une connexion valide; cette fois-ci, nous n'avons pas besoin de préciser la chaîne de connexion:

```
var connection = new OleDbConnection(cHelper.ConnectionString);
```

Puis, pour créer notre objet commande, nous demandons à notre helper de générer la commande:

```
var cmd = new OleDbCommand(helper.GenerateSQLFromUserQuery(txtSearch2.Text)
```

Et il ne nous reste plus qu'à lire les résultats comme nous l'avons fait précédemment

```
using (OleDbDataReader reader = cmd.ExecuteReader())
{
    lsvSearch2.Items.Clear();
    while (!reader.IsClosed && reader.Read())
    {
        lsvSearch2.Items.Add(new ListViewItem(new[] { reader.GetString(0) }));
    }
    reader.Close();
}
```

Ainsi, si à première vue, cette méthode ne semble pas apporter grand chose, elle permet avant tout de se passer de chaîne de connexion mais aussi et **surtout** de pouvoir utiliser les paramètres d'Advanced Query Search pour exécuter des recherches très fines, il est ainsi possible de concatener "kind:image" au texte que l'utilisateur aura saisi pour ne retourner que des fichiers de type image. Cela est bien entendu possible avec tous les critères que permet AQS.

4 - Pour aller plus loin

Windows Desktop Search n'est néanmoins pas un SGBD à part entière et dans de nombreux cas il vous faudra faire preuve de ruse pour atteindre le résultat souhaité. En voici quelques exemples:

4.1 - La recherche fulltext

Dans la requête que j'ai prise en exemple tout du long de cet article, nous recherchions les objets dont le nom contenait telle ou telle chaîne de caractère:

```
SELECT Top 5 System.ItemUrl, System.ItemName, System.Title FROM systemindex WHERE
CONTAINS(\"System.FileName\", \"*plage*\")
```

Comment faire pour rechercher la chaîne sur tous les champs? Tout simplement en omettant une partie du filtre

```
SELECT Top 5 System.ItemUrl, System.ItemName, System.Title FROM systemindex WHERE CONTAINS(\"*plage*
\")
```

4.2 - Les dates et heures

Et oui, qu'importe le langage de requête utilisé, ce n'est jamais la panacée de travailler avec celles-ci et vous devrez également adapter vos requêtes en fonction de vos besoins avec WDS.

Supposons que nous ne voulions retourner que les objets créés un jour J. Nous serions tentés de faire:

```
SELECT ... WHERE System.DateCreated = '2008-10-1'
```

Malheureusement, il y a 99% de chance que cela ne vous retourne rien car les dates de fichiers sont basées sur date + heure (ex "2008-10-1 13:10:23"). Ainsi, vous allez devoir travailler sur un interval plutôt qu'une date précise:

```
SELECT ... WHERE System.DateCreated <= '2008-10-1' AND System.DateCreated < '2008-10-2'
```

Et pour les dates, il suffit d'utiliser un format datetime

```
SELECT ... WHERE System.DateCreated <= '2008-10-1 10:00:00' AND
System.DateCreated < '2008-10-1 12:00:00'
```

Ce qui vous retournera tous les fichiers créés le 1er octobre 2008 entre 10h et midi.

4.3 - Les regroupements

Comme tout bon langage "SQL" qui se respecte, le SQL Windows Search permet de grouper les résultats simplement en ajoutant le mot clé **GROUP BY** et les noms des colonnes de votre choix. Pensez donc à l'utiliser lorsque nécessaire

4.4 - Connaître la version de WDS installée

Selon la version utilisée, les requêtes ne seront pas forcément identiques. Pour vérifier ce point, servez-vous de la clé de registre

HKEY_LOCAL_MACHINE/Software/Microsoft/Windows Desktop Search/Version

Conclusion

Il n'est pas courant que vous ayez à faire une recherche parmi les fichiers de votre disque dur, mais si un jour cela devait arriver, quelque soit l'algorithme utilisé (méthode récursif, multithreading), le temps de recherche sera toujours bien plus long que l'utilisation de Windows Desktop Search. Bien entendu, cela nécessite que le service d'indexation soit activé et la recherche ne pourra se faire que sur les dossiers ou partitions pour lesquelles l'indexation est autorisée. Garder donc toujours en tête que Windows Desktop Search peut vous être utile pour référencer des fichiers ou au contraire pour les proposer à l'utilisateur.

6 - Téléchargements

Pour les sources de l'article, c'est [ici](#)