

Indexing Service

Comment travailler avec Index Server



Sous Windows 2000 il existe un outil peu connu et donc peu utilisé, or celui-ci peut fournir une aide très précieuse pour la recherche documentaire. Ce produit permet d'explorer les fichiers eux-mêmes pour rechercher tous ceux qui contiennent les paramètres voulus. Voyons donc comment faire ceci.

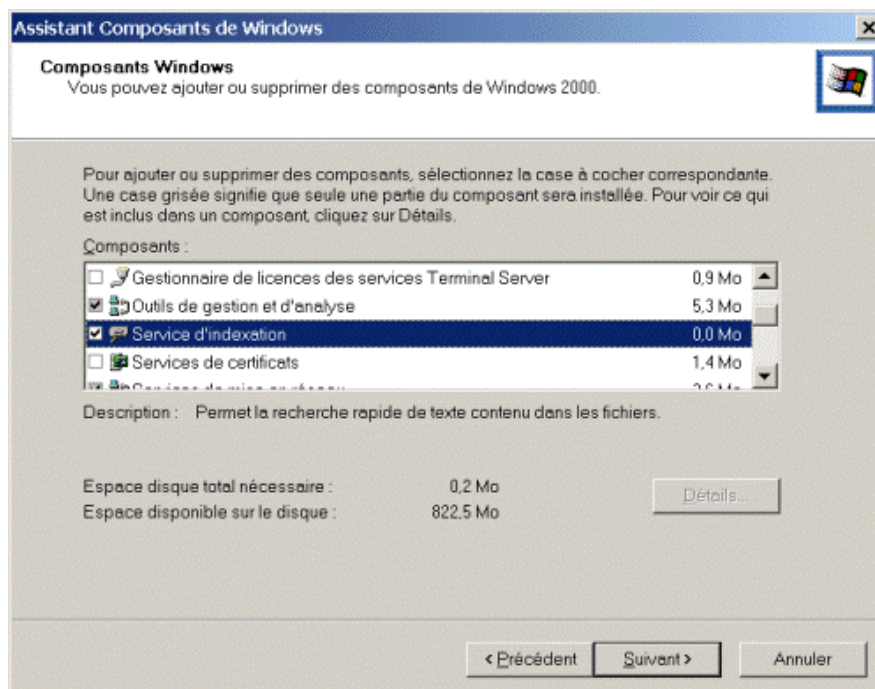
Travailler avec Indexing Service

Le Service d'Indexation (ou Indexing Service) est un utilitaire permettant de faire un moteur de recherche Full Text dans des répertoires donnés. Ainsi ce moteur va indexer les fichiers de type texte selon la fréquence d'apparition des mots (ne prenant pas en compte ceux classiques pré renseignés, par exemple, 'le', 'la', ...). Il va donc créer un **Catalogue** (en quelques sortes une Mini base de données) qui est interrogeable via programmation ou par formulaire.

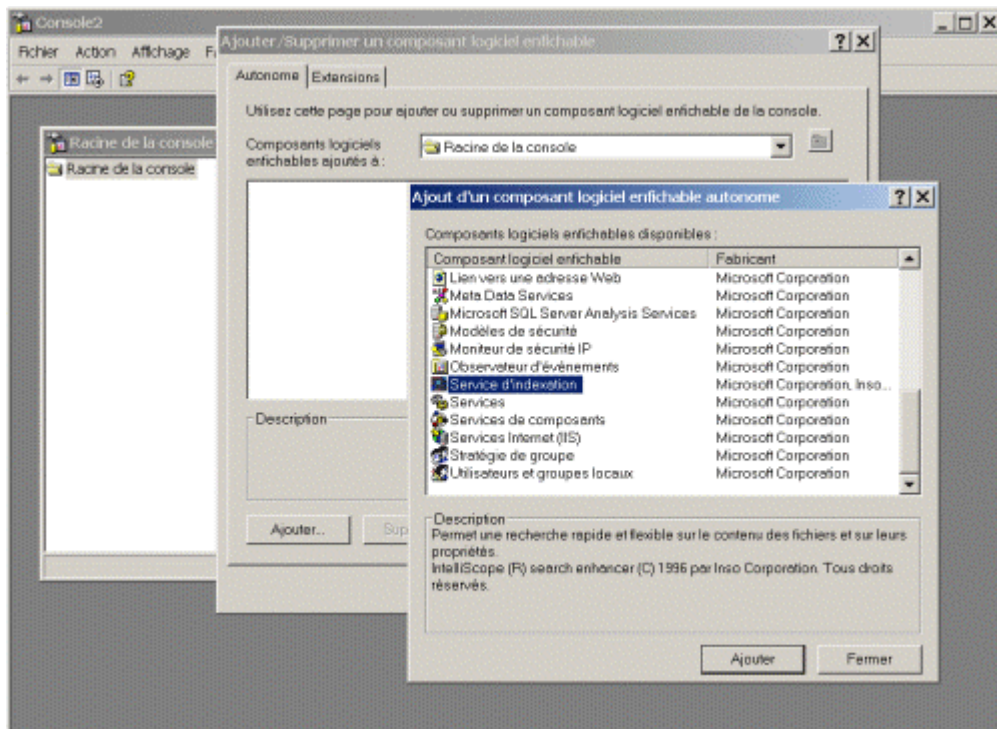
Ce catalogue contiendra des informations sur chaque fichier rencontré (la taille, la date et l'heure de modification, ...) et ces données seront réutilisables lors de la visualisation du résultat de la recherche.

Installation d'Indexing Service

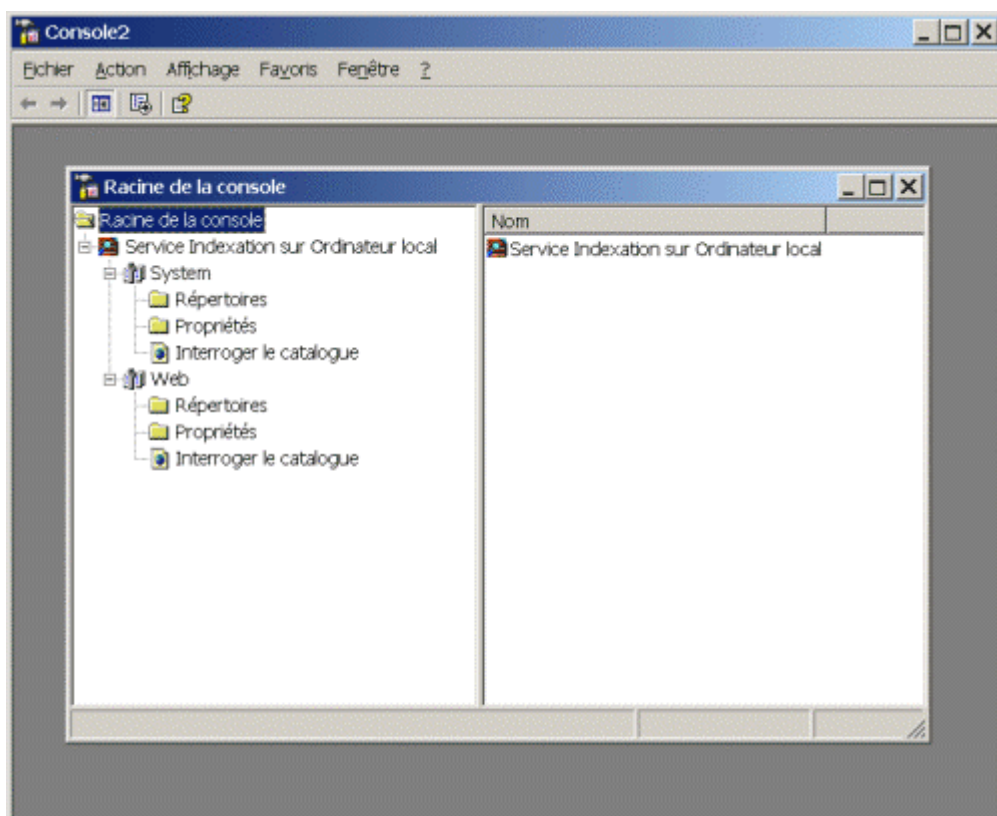
Indexing Service fait parti de la liste des composants de base installés par défaut sur les Systèmes Windows 2000 ou XP. On le retrouve malgré tout dans la liste des composants de Windows comme le montre la figure ci-dessous.



Une fois le composant installé, on a un nouveau composant qui apparaît dans la MMC (Microsoft management Console), il suffit donc de taper **mmc.exe** dans la fenêtre "exécuter", et ajouter le "composant logiciel enfichable" : **Service d'Indexation**.



On spécifie alors la machine que l'on souhaite configurer, soit en local soit en Remote sur un serveur du réseau accessible (par exemple le serveur WEB de développement ou de Production sur lequel on veut mettre en place cette solution d'indexation). On obtient alors l'exemple ci-dessous si on choisi le Ordinateur local.



On voit donc apparaître par défaut deux **Catalogues** qui sont ceux installés de base avec ce service (System et Web).

- Le catalogue "**System**" est celui qui permettra d'indexer tous les documents de la machine dans le cadre de l'utilisation de la recherche de Windows (Démarrer > Rechercher), quand on recherche tous les fichiers contenant une chaîne de caractères données.
- Le catalogue "**Web**" est présent dès que IIS et Indexing Service sont installés sur la machine. Par défaut, il indexera tous les documents situés sous "C:\InetPub"

Chaque catalogue est stocké dans un répertoire créé automatiquement par Indexing Service nommé **catalog.wci** qui se situe sous le répertoire visé lors de la création du catalogue. Du coup le catalogue Web est stocké dans le sous répertoire "C:\InetPub\catalog.wci". Chaque catalogue peut contenir une multitude de répertoires à indexer qui peuvent être des Sites WEB ou non. Nous nous concentrerons sur le cas de l'utilisation dans un site WEB, mais il faut bien garder à l'esprit que la source du catalogue peut être située n'importe où (sur le serveur d'Indexation ou non).

Nous allons maintenant voir comment configurer soit même Indexing Service et spécifier les différents types de documents à indexer.

Configuration d'Indexing Service

Il n'y a pas énormément de paramétrages possibles au niveau de l'outil d'indexation lui-même, mais je vais tout de même indiquer ici les différents choix offerts.

Composant IFilter

Tout d'abord, Indexing Services utilise une sorte de standard sur les données demandées lors de la lecture du fichier. Pour simplifier, quand Indexing Service explore un répertoire il vérifie s'il peut interroger chaque type de fichier (suivant l'extension) avec un outil qui est "**Ifilter**", en questionnant la base de registre.

Pour savoir quels sont les "Ifilter" installés sur la machine, il suffit d'ouvrir la base de registre et d'aller dans la clé :

- HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Control\ContentIndex

On voit alors l'élément "**DLLsToRegister**" qui contient la liste de toutes les **dll** utilisables par Indexing Service pour la création de son catalogue. On voit donc par là la liste des DLL :

- C:\WINDOWS\System32\offfilt.dll : Permet de cataloguer tous les documents Office (Microsoft Word, Microsoft Excel, and Microsoft PowerPoint) rencontrés par Indexing Service
- C:\WINDOWS\System32\nlhtml.dll : Permet de cataloguer tous les documents HTML (Attention : Ce composant ne catalogue pas les mots des Balises, car il considère que ceux-ci ne sont pas affichés lors d'une consultation dans une fenêtre Internet Explorer).
- C:\WINDOWS\System32\query.dll : C'est l'indexeur par défaut, il permet de cataloguer les fichiers Full Text
- ...

Ainsi si on a créé une extension particulière (cas de l'article sur les HTTPHandlers en .NET) qui respecte le standard d'un autre, on peut modifier la base de registre pour faire cataloguer ce fichier par un des Ifilter connus.

Prenons le cas où on a créé l'extension **.toto** qui est exécutée par .NET et respecte le standard HTML (voir l'article des HttpHandlers), pour faire cataloguer ce fichier par **nlhtml.dll**, il faut suivre l'explication de cet article :

- **Comment cataloguer un type de fichier non classique par nlhtml.dll (US)**

Quelques versions d'Ifilter

Le Standard Development Kit (SDK) pour la création de son propre IFilter est disponible sur le site de Microsoft à l'adresse suivante :

- [SDK pour IFilter \(US\)](#)

Le plus connu de tous est sans aucun doute PDFfilter, tout simplement parce que cet outil développé par Adobe permet d'indexer les fichiers de type PDF.

- [Présentation d'Ifilter pour les fichiers PDF \(US\)](#)
- [Téléchargement d'Ifilter pour les fichiers PDF](#)

Il existe divers IFilter développés (soit fournis gratuitement, soit payant), Je vous fournis une liste non exhaustive des Filtres connus suivant les types de fichiers :

- [Documents Microsoft Visio : Gratuit](#)
- [Documents StarOffice ou Open Office : Payant](#)
- [Documents Corel WordPerfect 8 : Gratuit](#)
- [Documents RTF : Gratuit](#)
- [Documents AutoCAD \(Extension DWG\) : Gratuit](#)
- [Fichiers XMP \(Adobe eXtensible Metadata Platform\) : Payant](#)
- [Documents XML : Payant](#)
- [Fichiers JPG : Gratuit](#)
- [Fichiers ZIP : Payant](#)
- [Fichiers ZIP \(Développé en .NET\) : Gratuit](#)
- [Fichiers MSG : Payant](#)
- [Fichiers MP3 : Gratuit](#)
- ...

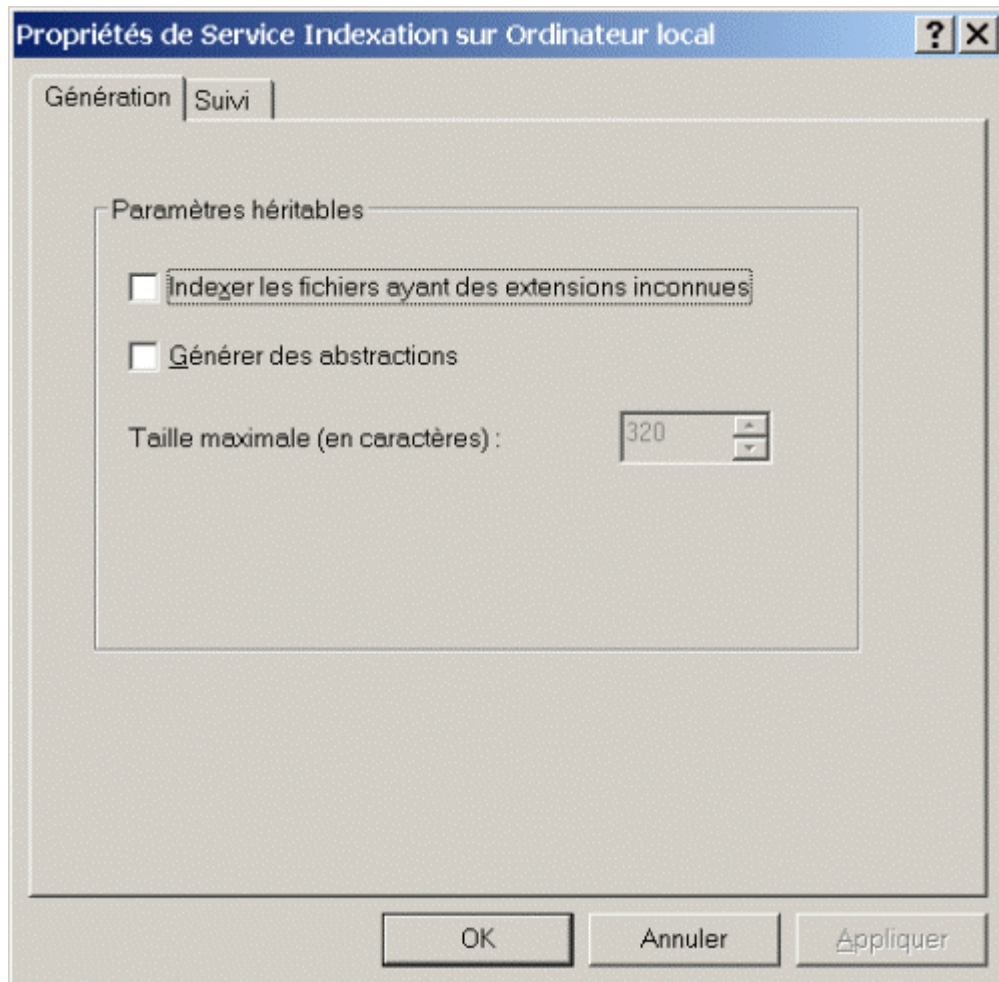
Vous trouverez ici une liste des principaux Filtres à installer dans le cadre de l'installation de SharePoint Portal Server :

- [Page concernant Ifilter de Share Point Portal Server sur le site de GIRAUDY EROL \(MVP SPS\)](#)

Paramétrage d'Indexing Service

Comme je l'ai dit précédemment, une fois les composants Ifilter que l'on veut installer, il y a très peu de paramètres possibles sur Indexing Service, mais je vais tout de même vous les présenter.

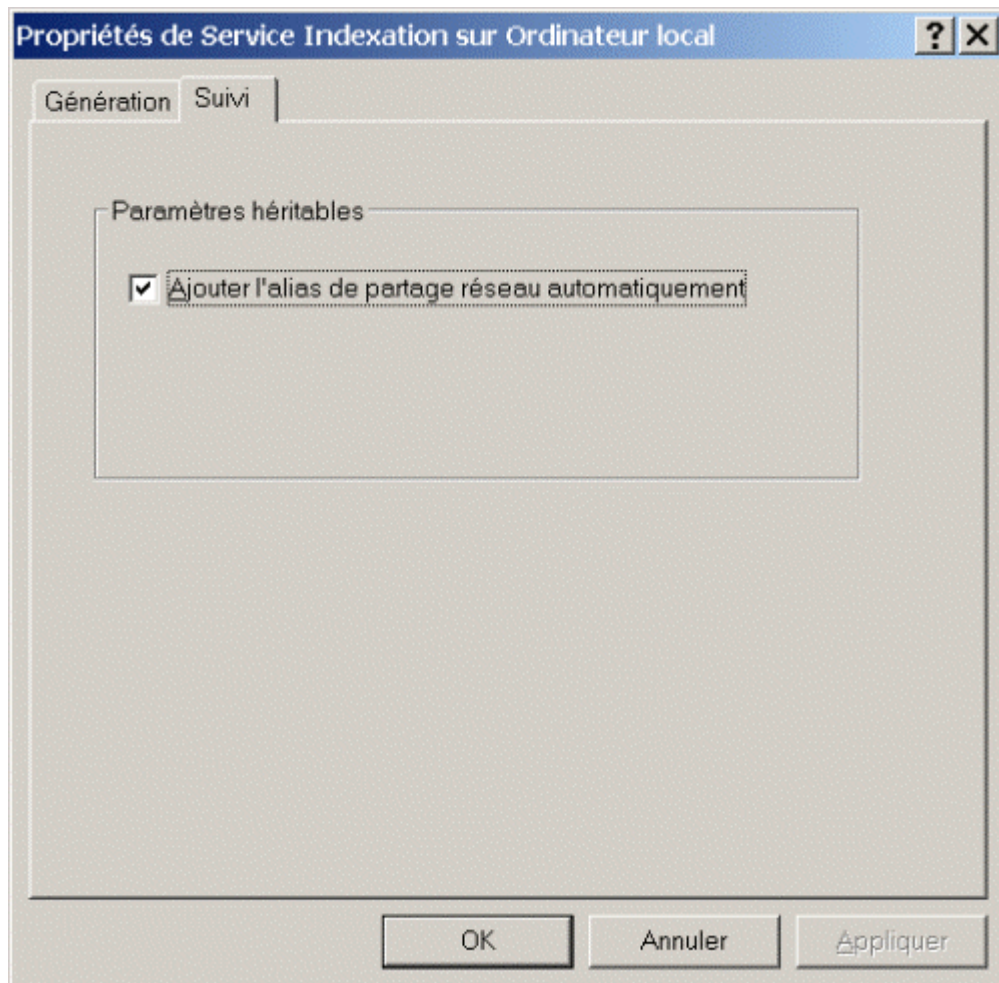
Reprenons notre MMC et en cliquant avec le bouton droit sur "Service d'Indexation sur Ordinateur Local" on voit apparaître "Propriété", les options possibles sont gérables dans cette interface.



Le premier Onglet "Génération" offre l'option de l'indexation des fichiers inconnus ("Indexer les fichiers ayant des extensions inconnues"), cette option est particulièrement nécessaire si vous avez des fichiers spéciaux (créés par vous même) à indexer, comme expliqué dans le lien un peu plus haut.

Le service d'indexation extrait le contenu et les propriétés de ces documents autant que cela lui est possible.

La seconde option ("Générer des abstractions") concerne la génération de résumé. En effet, Indexing service peut créer un résumé de chaque fichier rencontré suivant la taille donnée ("Taille maximale") en nombre de caractères, ce résumé est ensuite utilisable dans l'affichage des résultats comme n'importe quel autre champ.



Le second Onglet "Suivi" permet de pouvoir ajouter le nom du partage des répertoires partagés comme alias des répertoires ("Ajouter l'alias de partage réseau automatiquement").

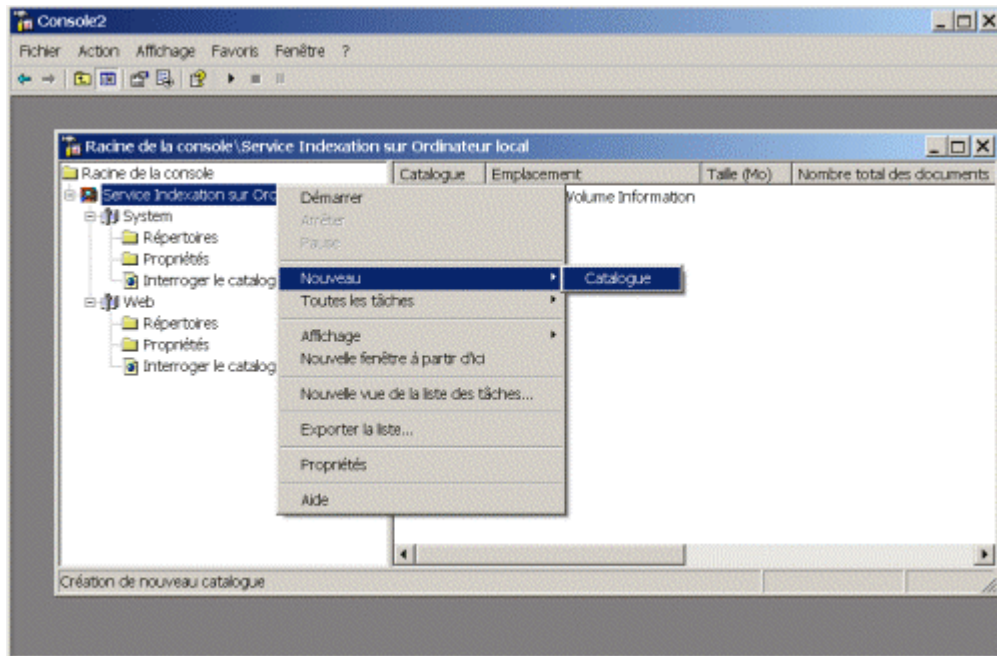
Une fois ceci fait, il faut arrêter et redémarrer le service d'indexation pour que celui-ci prenne en compte les nouveaux paramètres.

Maintenant que le Service d'Indexation est correctement configuré, voyons comment créer et personnaliser son catalogue pour ensuite pouvoir l'interroger.

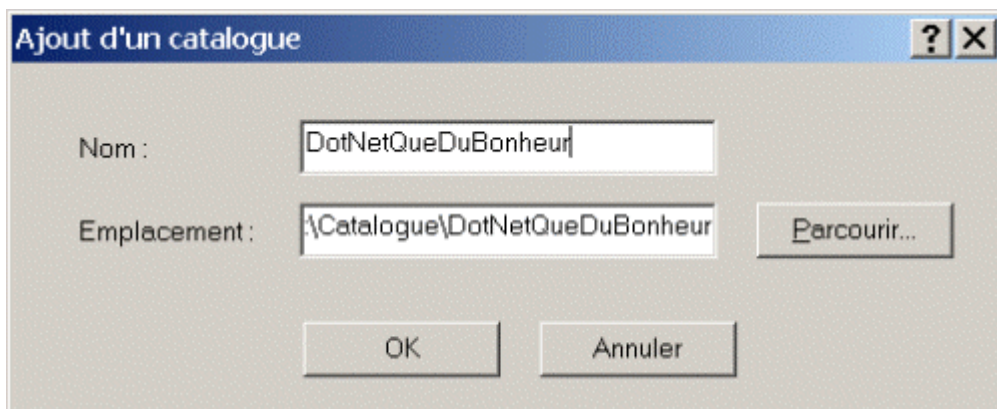
Création et paramétrage d'un Catalogue

Prenons le cas de ma machine de développement pour ce catalogue. Ainsi sur cette machine, tous les sites WEB sont dans le répertoire "E:\SITES_WEB". Je vais donc créer un catalogue qui va indexer tous les documents situés dans les sous-répertoires de "E:\SITES_WEB".

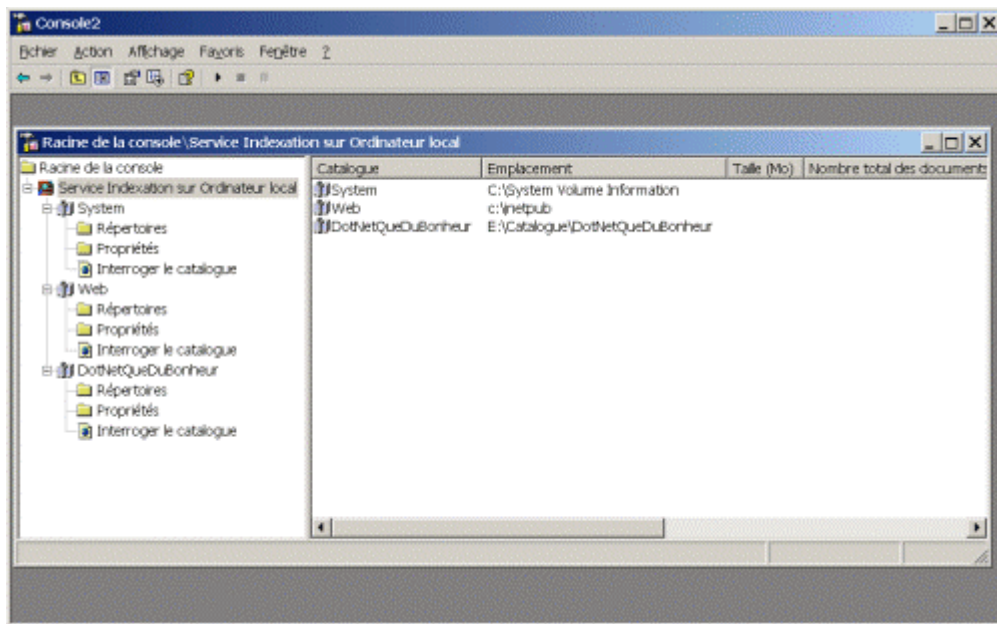
Il faut alors cliquer avec le bouton droit de la souris sur "Service d'indexation sur Ordinateur Local" et aller dans "Nouveau" et ensuite "Catalogue".



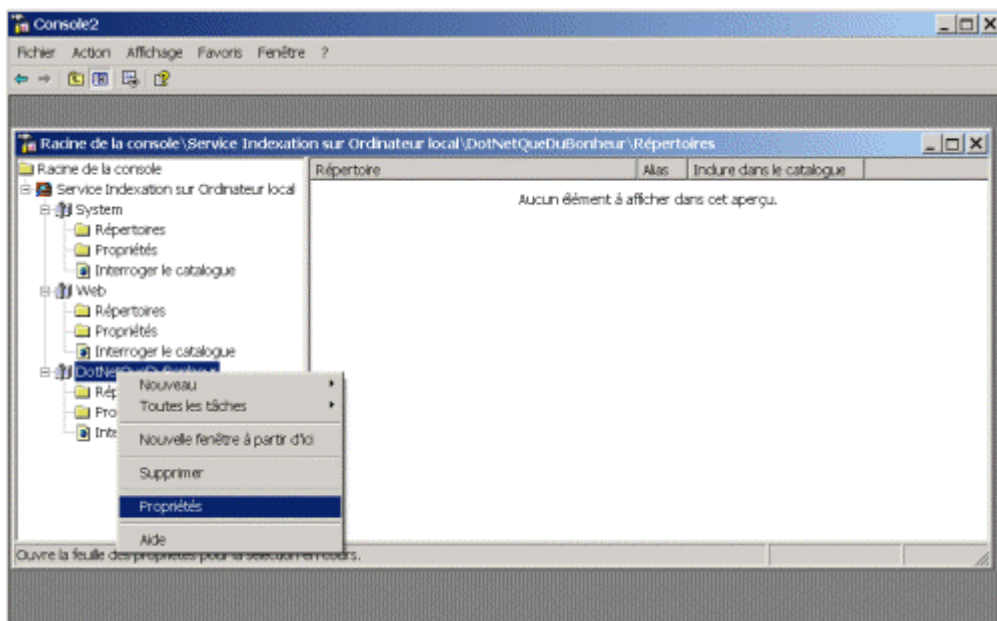
On choisit alors le nom du Catalogue (celui qui sera appelé des pages ASP) et le répertoire dans lequel celui-ci sera stocké. On peut donc se Créer un répertoire Catalogue qui contiendra l'ensemble des Catalogues existants sur le serveur (si on veut séparer les catalogues entre eux dans le cas de plusieurs sites Web sur la même machine par exemple).



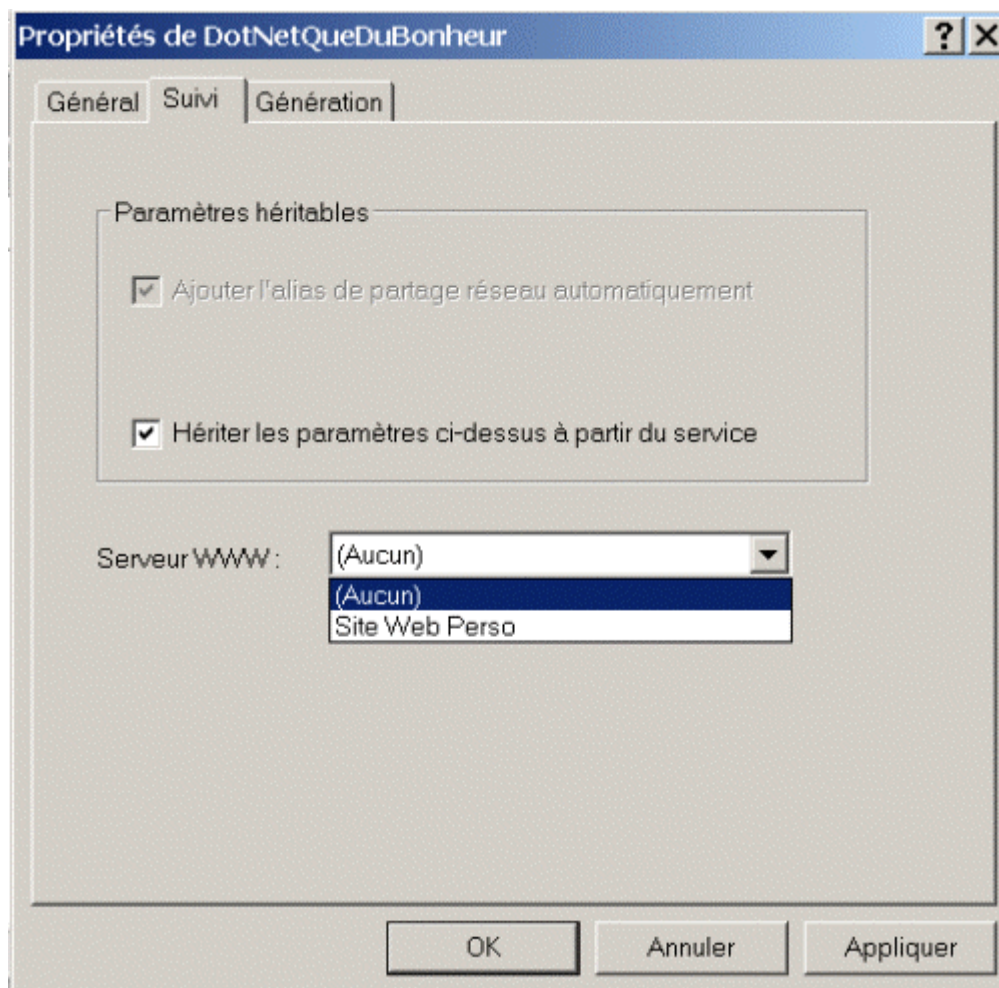
On obtiens donc notre nouveau catalogue "DotNetQueDuBonheur" qui sera stocké dans le répertoire "E:\Catalogue\DotNetQueDuBonheur" qui contient déjà le Sous-répertoire Catalog.wci.



Maintenant que ce catalogue est créé, il ne reste plus qu'à le paramétrer afin qu'il indexe bien la liste des répertoires voulus. Pour cela, il faut cliquer avec le bouton droit sur le nom de notre nouveau catalogue et choisir "Propriété".



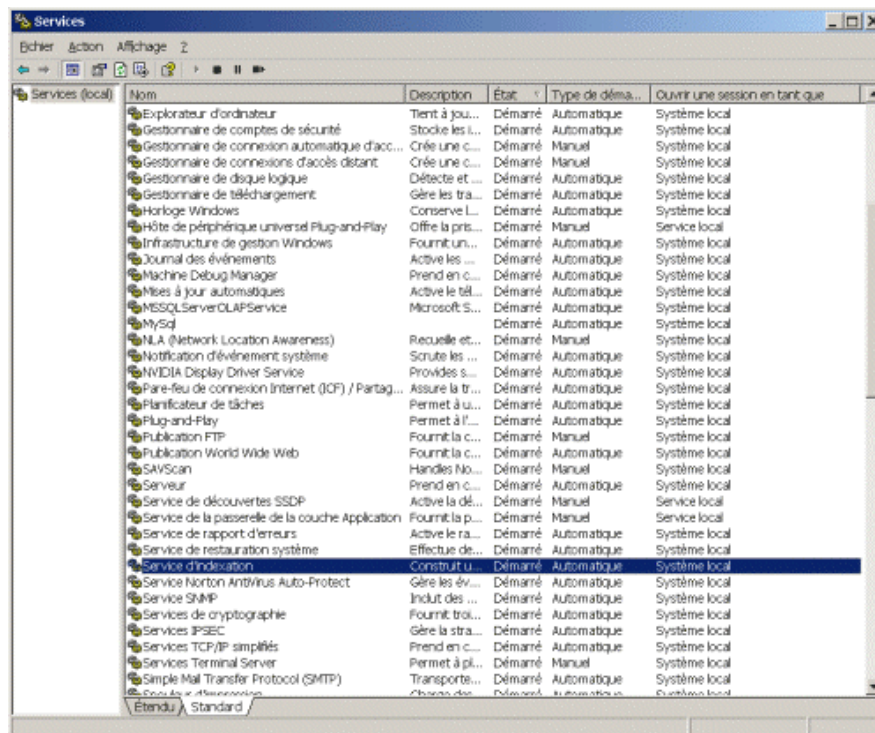
On va ensuite dans l'onglet "Suivi" afin de spécifier le Site WEB qui sera indexé.



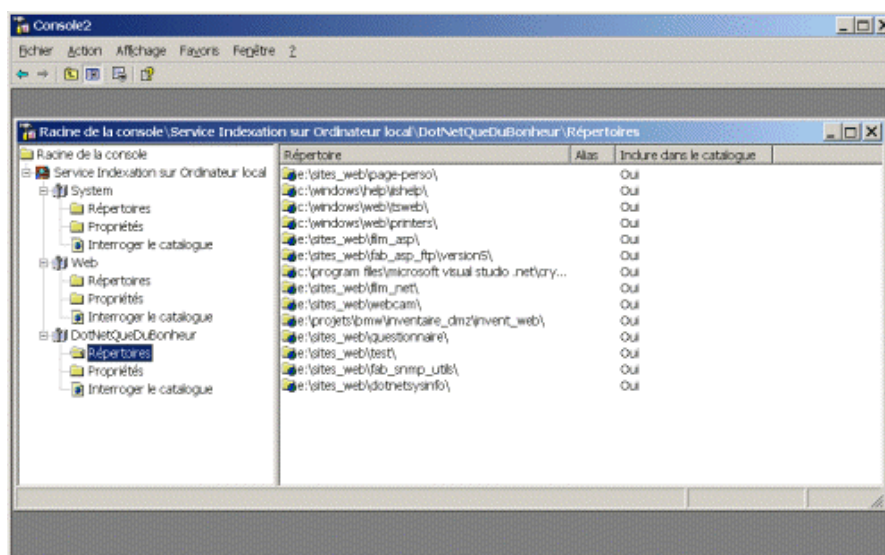
Dans le cas d'un serveur WEB hébergeant plusieurs sites, on les voit tous apparaître dans menu déroulant. On valide enfin ce choix et ferme cette fenêtre.

Les autres onglets permettent de voir les informations sur ce catalogue ou d'affiner un peu le paramétrage de l'indexation, nous laisserons les valeurs par défaut pour cet exemple.

Une fois ce choix fait il faut lancer le service d'indexation afin qu'il récupère les paramètres du site WEB (Répertoire racine du site, liste des répertoires virtuels, ...). Pour cela, il faut cliquer avec le bouton droit sur "Service d'indexation sur Ordinateur Local" et choisir : "Démarrer". On voit alors le service d'indexation avec le statut démarré dans la liste des services de la machine.



Si on retourne alors dans la liste des répertoires de notre catalogue, on voit apparaître la liste des tous les répertoires virtuels du site WEB choisi.



En cliquant avec le bouton droit sur "Répertoires", choisissant "Nouveau" et enfin "Répertoire", on peut ajouter tout autre répertoire qui n'aurait pas été ajouté automatiquement. Il faudra juste lui spécifier le chemin d'accès et un Alias si on souhaite lui en attribuer un.

Maintenant, notre catalogue est créé et entrain de se charger (via le service d'indexation actif), nous allons voir comment accéder à ce catalogue.

Exemple d'utilisation d'indexing Service en ASP3

En ASP 3, on peut questionner un catalogue d'Indexing Service de deux façons, soit en ADO, soit en passant par des objets spécifiques.

Il existe 2 objets pour Indexing Service en ASP3 :

- L'Objet **Query**
 - L'Objet **Utility**
-

L'objet **Query** permet de construire sa propre requête qui va être transmise à Indexing Service afin de récupérer le résultat dans un RecordSet. C'est avec cet objet que l'on spécifie les caractères à rechercher, les colonnes que l'on veut récupérer, le catalogue à utiliser, ...

Vous trouverez une présentation plus détaillée ici :

- [Objet ASP Query pour Index Server \(FR\)](#)
-

L'objet **Utility** est très spécifique, il permet de faire une optimisation de la requête en faisant par exemple une réduction de la portée d'exécution (par exemple en se limitant à un sous répertoire bien précis d'IIS). Celui-ci n'est pas obligatoire pour l'exécution d'une requête sur Indexing Service.

Vous trouverez une présentation plus détaillée ici :

- [Objet ASP Utility pour Index Server \(FR\)](#)
-

Une fois cette requête créée, voyons quels sont les paramètres que l'on peut récupérer d'un catalogue. Ils sont très nombreux mais parmi ceux-ci les plus utilisés sont :

- **Rank** : Le Rang (calculé lors de l'indexation du document)
- **DocTitle** : Titre du Document
- **Characterization** : Résumé du Document (créé lors de l'indexation du document seulement si l'option est activée pour le catalogue)
- **Vpath** : Chemin Virtuel d'accès au fichier
- **Create** : Date de Création du Document
- **Write** : Date de dernière modification du Document
- **HitCount** : Nombre d'occurrence du mot dans le Document
- **Size** : Taille en Octet du fichier
- **FileName** : Nom du Fichier

Vous trouverez une liste détaillée ici :

- [Liste des Propriétés Accessibles pour Index Server \(FR\)](#)

Voyons maintenant un exemple d'utilisation de cette méthode.

```
<%
'Creation des Objets de travail avec Indexing Service
set Q = Server.CreateObject("ixsso.Query")
set util = Server.CreateObject("ixsso.Util")

Dim SearchString, NextRecordNumber

NextRecordNumber = 0

' Déclaration de la chaîne de caractères à rechercher
SearchString = "Microsoft"

' Réduction du champs de recherche sur le Répertoire virtuel
' IIS de l'application et tous ses sous répertoires
' Si on veut le répertoire exact, il faut mettre "Shallow" à la place de "deep"
' Il existe aussi l'option "hierarchical" mais très peu utilisée,
' elle donne la hiérarchie des répertoires
util.AddScopeToQuery Q, "/RepertoireVirtuel", "deep"

' Déclaration du nom du Catalogue qui est interrogé
Q.CATALOG="DotNetQueDuBonheur"

Q.Query = SearchString

' On classera les résultats en fonction du Ranking obtenu
Q.SortBy = "rank[d]"

' Colonnes que l'on souhaite récupérer
Q.Columns = "DocTitle, DocAuthor, Vpath, FileName, " & _
" Size, Write, Characterization, rank, Contents, Create, HitCount"

' Nombre maxi de résultat (on ne récupérera que les 300
' premiers Document si on en obtient plus)
Q.MaxRecords = 100

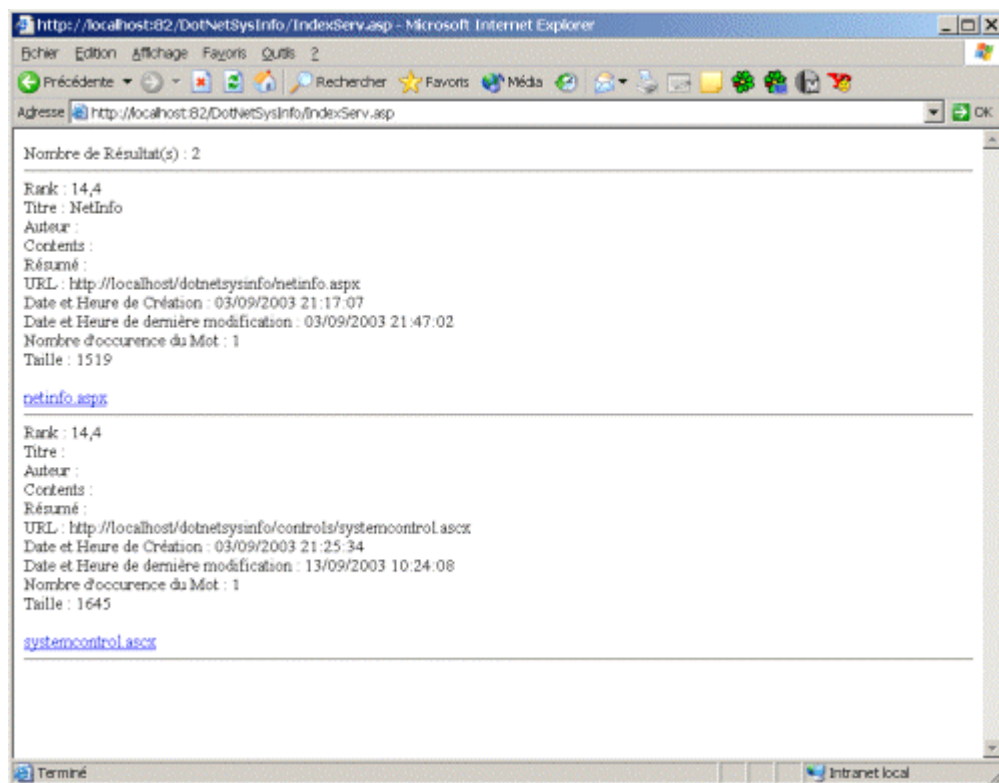
' Chargement du RecordSet avec le résultat de la recherche
set RS = Q.CreateRecordSet("nonsequential")

RS.PageSize = 10
ActiveQuery = TRUE

If Not RS.EOF Then
    Response.Write "Nombre de Résultat(s) : "& RS.RecordCount &"<BR><HR>"

    Do While Not RS.EOF
        ' ----- Affichage des éléments trouvés
        response.write ("Rank : " & RS("Rank")/10& "<br>")
        response.write ("Titre : " & rs("DocTitle")& "<br>")
        response.write ("Auteur : " & rs("DocAuthor")& "<br>")
        response.write ("Contents : " & rs("Contents")& "<br>")
        ' Attention: le résumé ne renvoie quelque chose que dans le cas
        ' où l'option de création d'un résumé est activé pour ce catalogue.
        response.write ("Résumé : " & Server.htmlEncode(rs("Characterization"))&"<br>")
        response.write ("URL : http://" & Request("server_name")&RS("Vpath")&"<br>")
        response.write ("Date et Heure de Création : " & rs("Create")& "<br>")
        response.write ("Date et Heure de dernière modification : " & rs("write")& "<br>")
        response.write ("Nombre d'occurence du Mot : " & rs("HitCount")& "<br>")
        response.write ("Taille : "&rs("Size")&"<br><br>")
        Response.Write ("<a href='"& RS("Vpath")&"' Target='_blank'>")
        Response.Write ( Server.htmlEncode( RS("FileName") ) &"</a><BR><HR>")
        RS.MoveNext
    Loop
Else
    Response.Write "Pas de résultat"
End If
%>
```


Le résultat donnera donc ceci :



Pour le "requêtage" d'un catalogue Indexing Service en utilisant ADO, la différence est principalement dans la création de la requête (query) où on va simuler le SQL.

Vous trouverez un exemple d'utilisation d'indexing Service en ADO ici :

- **Requête de catalogue Indexing Service en ASP 3 via ADO (US)**

Maintenant que nous savons comment attaquer le catalogue en ASP3, voyons comment ce même catalogue peut être utilisé cette fois ci en ASP.NET.

Exemple d'utilisation d'indexing Service en ASP.NET

Maintenant que nous avons vu l'utilisation d'Indexing Service en ASP 3, voyons un exemple tout aussi simple en ASP.NET (plus particulièrement en VB.NET).

Recherchons le Même mot dans le même catalogue. On affichera cette fois le résultat directement dans un Datagrid (avec une génération automatique des colonnes). Le but ici est de montrer qu'en seulement une dizaine de lignes, on peut interroger le catalogue de son choix.

L'exemple ici sera écrit en VB.NET, car beaucoup de sources très bien faites existent pour le C#, et n'étant pas expert en C#, je préfère vous fournir les adresses de ces très bons exemples.

Fichier TestIndex.aspx

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="TestIndex.aspx.vb" Inherits="DotNetSysInfo.TestIndex"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//FR">
<HTML>
<HEAD>
<title>Test d'utilisation d'indexing Service</title>
<meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.0">
<meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie3-2nav3-0">
</HEAD>
<body>
<form id="Form1" method="post" runat="server">
<asp:DataGrid id="MonDataGrid" runat="server" Width="100%" BorderColor="#3366CC"
BorderStyle="None" BorderWidth="1px" BackColor="White" CellPadding="4">
<SelectedItemStyle Font-Bold="True" ForeColor="#CCFF99"
BackColor="#009999"></SelectedItemStyle>
<ItemStyle ForeColor="#003399" BackColor="White"></ItemStyle>
<HeaderStyle Font-Bold="True" ForeColor="#CCCCFF" BackColor="#003399"></HeaderStyle>
<FooterStyle ForeColor="#003399" BackColor="#99CCCC"></FooterStyle>
<PagerStyle HorizontalAlign="Left" ForeColor="#003399"
BackColor="#99CCCC" Mode="NumericPages"></PagerStyle>
</asp:DataGrid>
</form>
</body>
</HTML>
```

Fichier TestIndex.aspx.vb

```
' Exemple simple de recherche dans un catalogue Index Services EN vb.net

Imports System.Data
Imports System.Data.OleDb
Imports System.Collections
Imports System.Collections.Specialized

Public Class TestIndex
    Inherits System.Web.UI.Page
    Protected WithEvents MonDataGrid As System.Web.UI.WebControls.DataGrid

    Private MonoleDbSelectCommand1 As New System.Data.OleDb.OleDbCommand()
    Private MondbAdapter As New System.Data.OleDb.OleDbDataAdapter()
    Private MadbConnection As New System.Data.OleDb.OleDbConnection()
    Private LaRequette As String = ""
    Protected WithEvents DataGrid1 As System.Web.UI.WebControls.DataGrid
    Private MaDataTable As New DataTable()

    #Region " Code généré par le Concepteur Web Form "

    'Cet appel est requis par le Concepteur Web Form.
    <System.Diagnostics.Debug>StepThrough() Private Sub InitializeComponent()

    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Init
        'CODEGEN : cet appel de méthode est requis par le Concepteur Web Form
        'Ne le modifiez pas en utilisant l'éditeur de code.
        InitializeComponent()
    End Sub

    #End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

        Me.MondbAdapter.SelectCommand = Me.MonoleDbSelectCommand1
        Me.MonoleDbSelectCommand1.Connection = Me.MadbConnection

        ' Initialisation de la connexion avec notre catalogue
        Me.MadbConnection.ConnectionString = "Provider=MSIDXS;Data Source=DotNetQueDuBonheur"

        ' Création de la requette pour ADO.NET avec la recherche du mot Machine et DotNet
        LaRequette = "SELECT Rank, VPath, DocTitle, Filename, Characterization, Write " & _
        "FROM SCOPE('DEEP TRAVERSAL OF ""/DotNetSysInfo""') " & _
        "WHERE NOT CONTAINS(VPath, ""_vti_"" OR ""_.config"" ) " & _
        "AND CONTAINS(Contents, ""Machine"" OR ""DotNet"" ) " & _
        "OR CONTAINS(DocTitle, ""Machine"" OR ""DotNet"" ) " & _
        "ORDER BY Rank DESC"

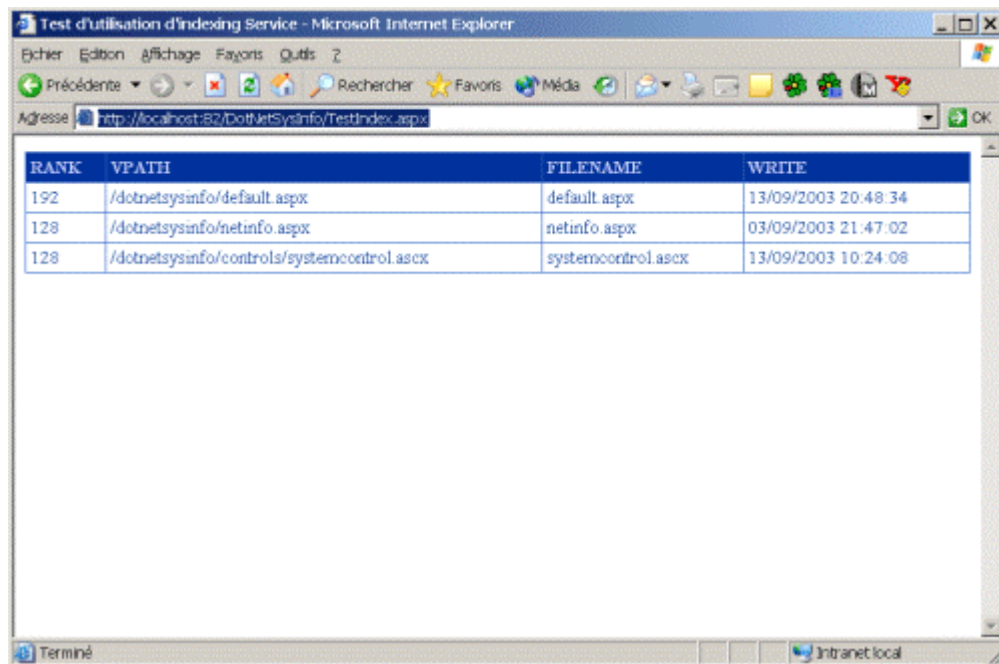
        ' Assigne la requette et charge le résultat
        Me.MondbAdapter.SelectCommand.CommandText = LaRequette
        Me.MondbAdapter.Fill(MaDataTable)

        ' Envoie le résultat dans le Datagrid
        Me.MonDataGrid.DataSource = MaDataTable
        Me.MonDataGrid.DataBind()

    End Sub

End Class
```

Le résultat donnera donc ceci :



The screenshot shows a web browser window titled "Test d'utilisation d'Indexing Service - Microsoft Internet Explorer". The address bar displays "http://localhost:82/DotNetSysinfo/TestIndex.aspx". The main content area contains a table with the following data:

RANK	VPATH	FILENAME	WRITE
192	/dotnetsysinfo/default.aspx	default.aspx	13/09/2003 20:48:34
128	/dotnetsysinfo/netinfo.aspx	netinfo.aspx	03/09/2003 21:47:02
128	/dotnetsysinfo/controls/systemcontrol.ascx	systemcontrol.ascx	13/09/2003 10:24:08

The status bar at the bottom indicates "Terminé" and "Intranet local".

Comme je vous l'ai dit cet exemple est extrêmement simple. En fait sous .NET toute la difficulté est de construire la requête qui sera envoyée à Indexing Service.

Les requêtes envoyées à Indexing Service respectent en grande partie les standards SQL, mais certains mots sont spécifiques à cet outil. Voyons ces mots nécessaires pour construire notre requête :

- **SCOPE** : Permet de limiter la portée de la Requête à un répertoire spécifique du catalogue (cas des Sites WEB), il correspond à la fonction "AddScopeToQuery" de l'objet "Utils" utilisé en ASP 3
- **CONTAINS** : Permet de spécifier l'existence d'un mot dans un des paramètres (VPath, Contents, etc. ...)
- **FREETEXT** : Peut exécuter une recherche comme CONTAINS mais plus proche du langage naturel
- **FORMSOF** : Permet de donner une base de mot, ainsi le résultat aura toutes les formes possibles de ce mot (par exemple : ('FORMSOF(inflectional, "code") ') donnera tous les mots ayant code inclus comme codeur, décode, encoder, ...)
- **NEAR** : Accompagné de CONTAINS, il permet de tester la Proximité d'un mot avec un autre (par exemple : CONTAINS(' "Visual" NEAR() "Studio" ') testera toutes les fois où ces 2 mots sont l'un proche de l'autre)

Un très bon tutorial pour la création de cette requête est visible ici :

- [Présentation de l'utilisation d'Indexing Service avec ADO.NET \(US\)](#)

Pour tous les développeurs C#, voici deux URL vous donnant de très bons exemples de développement .NET sur Indexing Service :

- [ASP.NET et Index Server sur CodeProject \(US\)](#)
- [ASP.NET et Index Server sur devhood \(US\)](#)

Conclusion

Cet article est une présentation des possibilités d'utilisation de l'outil d'indexation documentaire de Microsoft, il reste de nombreuses possibilités à découvrir tout particulièrement dans le développement de client pour le catalogue. J'espère que ça vous aura donné envie de vous servir de ce produit qui est encore plutôt méconnu.

Si vous souhaitez en savoir plus sur cet outil de recherche documentaire de Microsoft, vous pouvez aller sur les sites dédiés :

- [Création et utilisation d'un catalogue en ASP \(FR\)](#)
- [MSDN Indexing Service \(US\)](#)
- [Paramétrages de base d'Indexing Service \(US\)](#)
- [Tutorial d'utilisation d'Indexing Service en .NET \(US\)](#)
- [Site d'aide de Microsoft pour l'Option Pack NT4 en Ligne \(FR\)](#)
- [Spécificité des requêtes SQL avec Indexing Service \(FR\)](#)

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F____)