


ASP.Net : Générez des Rapport Excel depuis votre application Web

Article Rédigé pour 

11/10/2003
Par Elise Dupont

niveau : facile
durée : de 20 à 45 minutes
[Code Source Exemple VB.NET](#)
[Code Source Exemple C#](#)

Droit de diffusion:

L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé sur d'autres sites Web sans l'autorisation au préalable de son créateur.

Avant Propos :

Vous avez des utilisateurs (du département finance par exemple) qui adorent Excel, et qui vous demandent tout le temps des rapports Excel ? Non ? Vous avez de la chance, moi si ;-)

Voici un petit article qui vous explique comment exporter vos données (un Dataset par exemple) depuis une application ASP.Net vers un fichier Excel avec un graphique, des moyennes, etc... Libre à vous ensuite d'adapter à vos besoins, mais cet exemple vous donnera une idée des possibilités.

Je vous conseille vivement de télécharger la Démo avec Code Source, choisissez votre langage préféré (vous avez le choix entre [VB.Net](#) et [C #](#)), installez tout ça et testez par vous même, c'est le meilleur moyen pour apprendre. Cet exemple tourne sous Office XP. C'est parti !

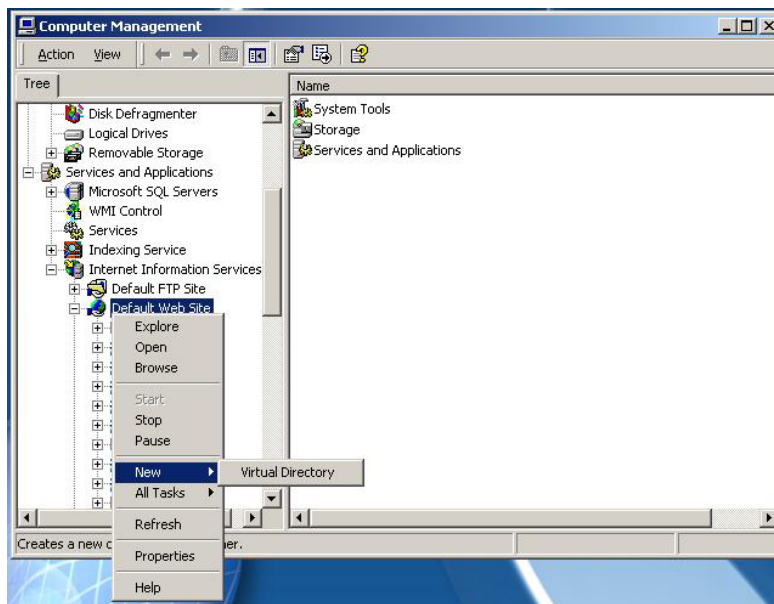
Sommaire:

- [1. Faire tourner la Démo](#)
- [2. Configurer \(serveur IIS, Sécurité des composants COM\)](#)
 - [2.1. Installer Office PIA](#)
 - [2.2. Configurer la sécurité](#)
- [3. Générer un rapport depuis un dataset](#)
 - [3.1. Créer un template avec graphique](#)
 - [3.2. Créer un objet Excel](#)
 - [3.3. Le remplir](#)
 - [3.4. Terminer les process comme il faut](#)
- [4. Renvoyer le rapport au client](#)

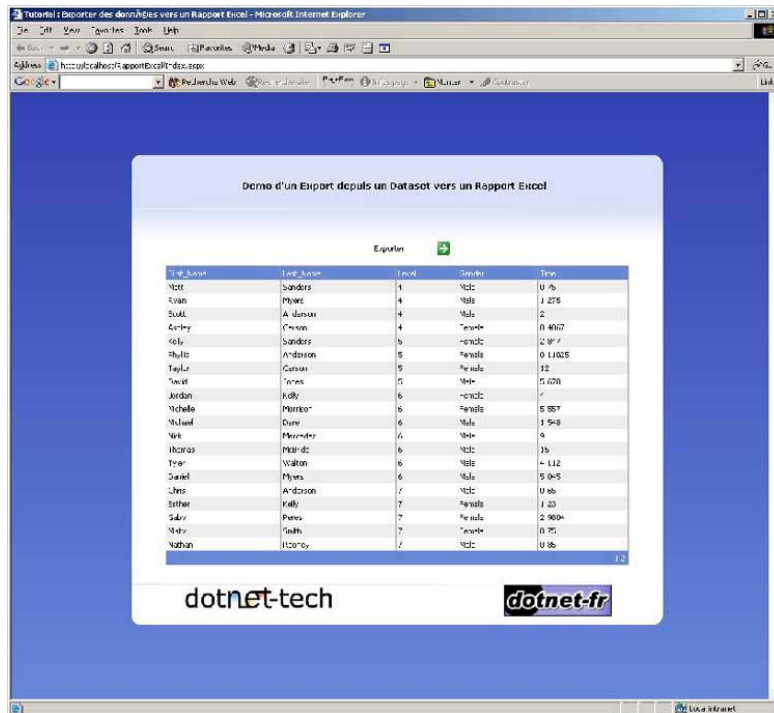
5. Sources additionnelles

1. Faire tourner la Démo

- Téléchargez le code source avec la langue de votre choix
- Dé-zippez le contenu du zip dans un répertoire de votre choix
- Ouvrez le Computer Management
- Créez une Virtual Directory « ExcelDemo » et mappez le chemin vers le répertoire où vous avez dézipé les fichiers



- Configurez la sécurité (voir [chapitre ci après](#))
- Vous pouvez lancer la démo en ouvrant dans IE <http://localhost/ExcelDemo/Index.aspx>
- Vous devriez obtenir une page comme celle ci (la résolution a été réduite pour des raisons de taille):



2. Configurer (serveur IIS, Sécurité des composants COM)

S'il s'agissait d'une application Windows, il n'y aurait aucune configuration ou presque. Malheureusement, en ASP.Net, sur un serveur IIS, il y a quelques petits paramètres de Sécurité pour pouvoir exécuter les composants Excel. De plus, il vous faudra télécharger des composants en plus.

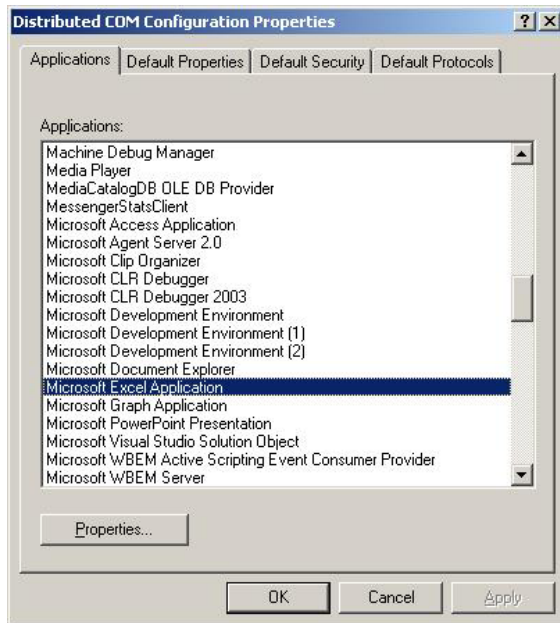
2.1. Installer Office PIA

- Il vous faut d'abord télécharger les "Primary Interop Assemblies" (PIA) pour pouvoir développer des applications .NET pour Office. Téléchargez les ici : [Office XP PIA](#).
- Ensuite Dé-zippez le tout et lancez le batch qui va inscrire toutes ces Assembly dans le GAC. Gardez les fichiers dans un coin, ils vous serviront à faire un « Add Reference » ensuite dans VS.Net.

2.2. Configurer la sécurité

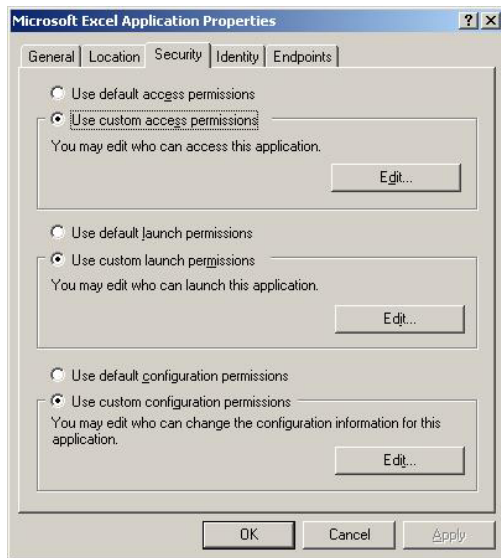
Le Serveur IIS ne vous laissera pas exécuter les composants Office sans que vous ne changiez les paramètres sécurité COM. Voilà la démarche à suivre :

1. Depuis votre invite de commande DOS entrez la commande « dcomcnfg.exe »
2. La fenêtre « Distributed COM Configuration Properties » va s'afficher au bout de quelques secondes



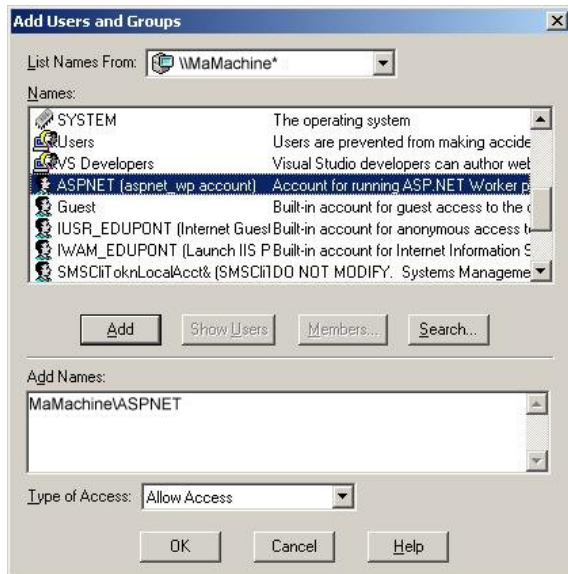
3. Dans la premier Onglet « Application » déroulez la liste jusqu'à ce que vous trouviez l'élément « Microsoft Excel Application » - Double cliquez dessus

4. Dans la fenêtre qui apparaît, cliquez sur l'onglet « Security »



5. Cochez l'option « Use custom Access permissions » plus cliquez sur le bouton Edit en dessous

6. Ajoutez le compte ASPNET – cliquez ok, vous revenez a la fenêtre Security



7. Cochez l'option « Use custom launch permissions » et faites de même : Bouton Edit > Ajoutez le compte ASPNET -

8. Validez

Voilà, la sécurité est configurée.

3. Générer un rapport depuis un dataset

3.1. Créer un template avec graphique

Il vous faut d'abord un fichier Excel sur lequel vous allez vous baser. Il contiendra les graphiques, la mise en page, il vous suffira ensuite de l'ouvrir, de le remplir avec les bonnes données. Inspirez-vous du fichier Template.xls fournit avec le Coude source à télécharger => [Code Source Exemple VB.NET](#) et [Code Source Exemple C#](#)

3.2. Créer un objet Excel

Pour créer un objet Excel (ou l'ouvrir s'il existe déjà), la marche à suivre est la suivante :

- Ajoutez une référence dans votre projet vers l'assembly Microsoft.Office.Interop.Excel, cette assembly fait partie du lot que vous avez téléchargé plus haut.
- Importez **Microsoft.Office.Interop** dans votre classe
- Le code pour ouvrir un fichier Excel puis la feuille désirée est le suivant :

Visual Basic .NET

```

1: oExcelApp = New Excel.ApplicationClass() 'Creer
l'objet Excel
2: oExcelApp.Visible = False 'Ne pas l'afficher
3: oBooks = oExcelApp.Workbooks
4: 'ouvrir le fichier Excel désiré
(oRien=System.Reflection.Missing.Value)
5: oBook = oBooks.Open("monfichierexcel.xls", oRien,
oRien, oRien, oRien, oRien,
oRien, oRien, oRien, oRien, oRien, oRien, oRien,

```

```

oRien)
6: 'ouvrir la feuille n° nIndex
7: oSheet = oBook.Worksheets(nIndex)

```

C#

```

1: // Créer l'objet Excel
2: oExcelApp = new
Microsoft.Office.Interop.Excel.ApplicationClass();
3: oExcelApp.Visible = false; // Ne pas l'afficher
4: oBooks = oExcelApp.Workbooks;
5: ouvrir le fichier Excel désiré
(oRien=System.Reflection.Missing.Value)
6: oBook = oBooks.Open("monfichierexcel.xls", oRien,
oRien, oRien, oRien, oRien,
oRien, oRien, oRien, oRien, oRien,
oRien, oRien, oRien);
7: //ouvrir la feuille n° nIndex
8: oSheet = oBook.Worksheets(nIndex);

```

3.3. Le remplir

Maintenant vous pouvez remplir le fichier avec les données de votre dataset. Il suffit de le parcourir et écrire cellule par cellule dans le fichier Excel :

Visual Basic .NET

```

1: For Each oCol In dtData.Columns
2:     nCol += 1
3:     oSheet.Cells(1, nCol) = oCol.ColumnName
4: Next oCol
5: 'Remplir ligne par ligne, colonne par colonne avec
les données
6: For Each oRow In dtData.Rows
7:     nCol = 0
8:     nRow += 1
9:     For Each oCol In dtData.Columns
10:         nCol += 1
11:         oSheet.Cells(nRow, nCol) =
oRow(oCol.ColumnName).ToString()
12:     Next oCol
13: Next oRow

```

C#

```

1: foreach(DataColumn oCol in dtData.Columns)
2: {
3:     nCol += 1;
4:     oSheet.Cells[1, nCol] = oCol.ColumnName;
5: }
6: //Remplir ligne par ligne, colonne par colonne avec
les données
7: foreach(DataRow oRow in dtData.Rows)
8: {
9:     nCol = 0;
10:    nRow += 1;
11:    foreach (oCol in dtData.Columns)

```

```

12:     {
13:         nCol += 1;
14:         oSheet.Cells[nRow, nCol] =
oRow[oCol.ColumnName].ToString();
15:     }
16: }
17:

```

3.4. Terminer les process comme il faut

Maintenant qu'on a utilisé les objets, il faut nettoyer tout ça, libérer l'application, car nous sommes dans le contexte d'une application Web. Voici le code :

Visual Basic .NET

```

1:  If Not IsNothing(oBook) Then oBook.Close(True, strTemplate,
oRien)
2:  If Not IsNothing(oSheet) Then
System.Runtime.InteropServices.Marshal.ReleaseComObject(oSheet)
3:  oSheet = Nothing
4:  If Not IsNothing(oBook) Then
System.Runtime.InteropServices.Marshal.ReleaseComObject(oBook)
5:  oBook = Nothing
6:  If Not IsNothing(oBooks) Then
System.Runtime.InteropServices.Marshal.ReleaseComObject(oBooks)
7:  oBooks = Nothing
8:  If Not IsNothing(oExcelApp) Then
9:      oExcelApp.Quit()
10:
System.Runtime.InteropServices.Marshal.ReleaseComObject(oExcelApp)
oExcelApp = Nothing
11: End If

```

C#

```

1:  if (oBook != null) oBook.Close(true, strTemplate, oRien);
2:  if (oSheet != null)
System.Runtime.InteropServices.Marshal.ReleaseComObject(oSheet);
3:  oSheet = null;
4:  if (oBook != null)
System.Runtime.InteropServices.Marshal.ReleaseComObject(oBook)
5:  oBook = null;
6:  if (oBooks != null)
System.Runtime.InteropServices.Marshal.ReleaseComObject(oBooks)
7:  oBooks = null;
8:  if (oExcelApp != null)
9:  {
10:     oExcelApp.Quit();
11:
System.Runtime.InteropServices.Marshal.ReleaseComObject(oExcelApp)
oExcelApp = null;
12:  }

```

4. Renvoyer le rapport au client

Ensuite on renvoie le fichier à l'utilisateur, pour qu'il puisse le télécharger et le sauver sur son poste :

Visual Basic .NET

```
1: Response.Clear()
2: Response.ContentType = "application/x-msexcel"
3: Response.AddHeader("Content-Disposition", "inline;
filename=MyReport.xls")
4: Response.WriteFile(oReport.ReportTemplate.ToString)
```

C#

```
1: Response.Clear();
2: Response.ContentType = "application/x-msexcel";
3: Response.AddHeader("Content-Disposition", "inline;
filename=MyReport.xls");
4: Response.WriteFile(oReport.ReportTemplate);
```

Voilà, je vous ai donné les grandes lignes, mais je vous conseille de vous aider avec le code source à télécharger. Libre à vous de modifier l'Excel pour qu'il soit adapté à vos besoins.
Bon Code !

5. Sources additionnelles

[HOW TO: Transfer Data to an Excel Workbook by Using Visual Basic .NET](#)

[INFO: Considerations for Server-Side Automation of Office](#)

[Manipulate Excel Spreadsheet Data in ASP Using ADO](#)

[INFO: Limitations of Office 2000 Web Components When Used Server-Side](#)



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