Utilisez le SPGridView dans vos composants SharePoint

Par

Date de publication : 20 décembre 2008

Le but de cet article est d'exposer les principales caractéristiques et fonctionnalités proposées par le SPGridView afin d'utiliser celui-ci dans vos composants SharePoint.

I - Pré-requis

Afin de pouvoir reproduire facilement ce qui est décrit dans l'article, vous devez disposer des éléments suivants :

- Une installation SharePoint 2007
- Visual Studio 2008 ou 2005. Notez que le projet disponible en téléchargement a été développé avec VS 2008.

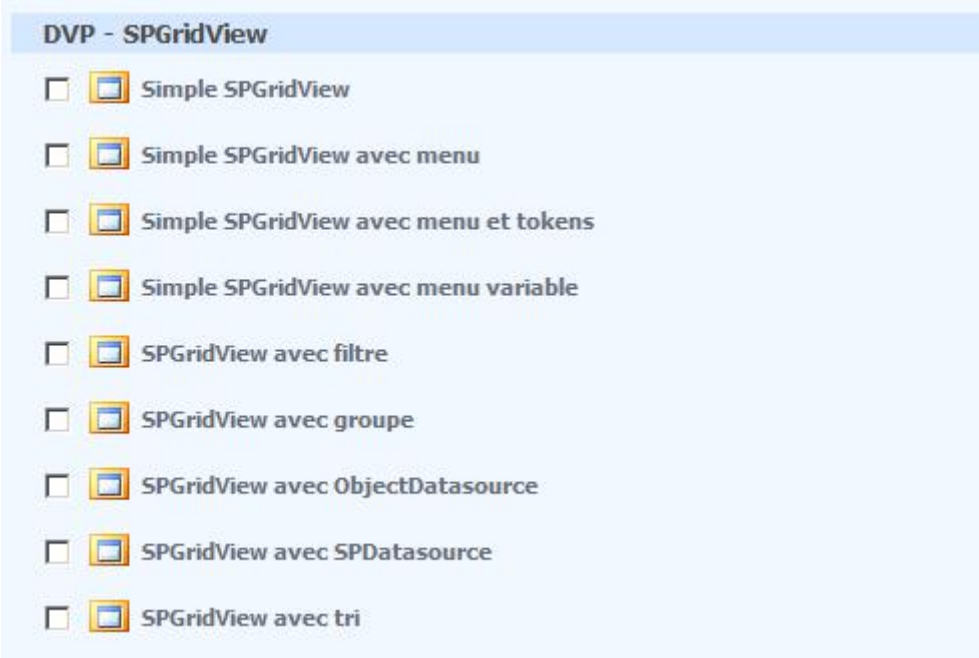
Une connaissance de SharePoint est un atout mais le niveau de l'article est très facile, donc pour les débutants.

II - Introduction

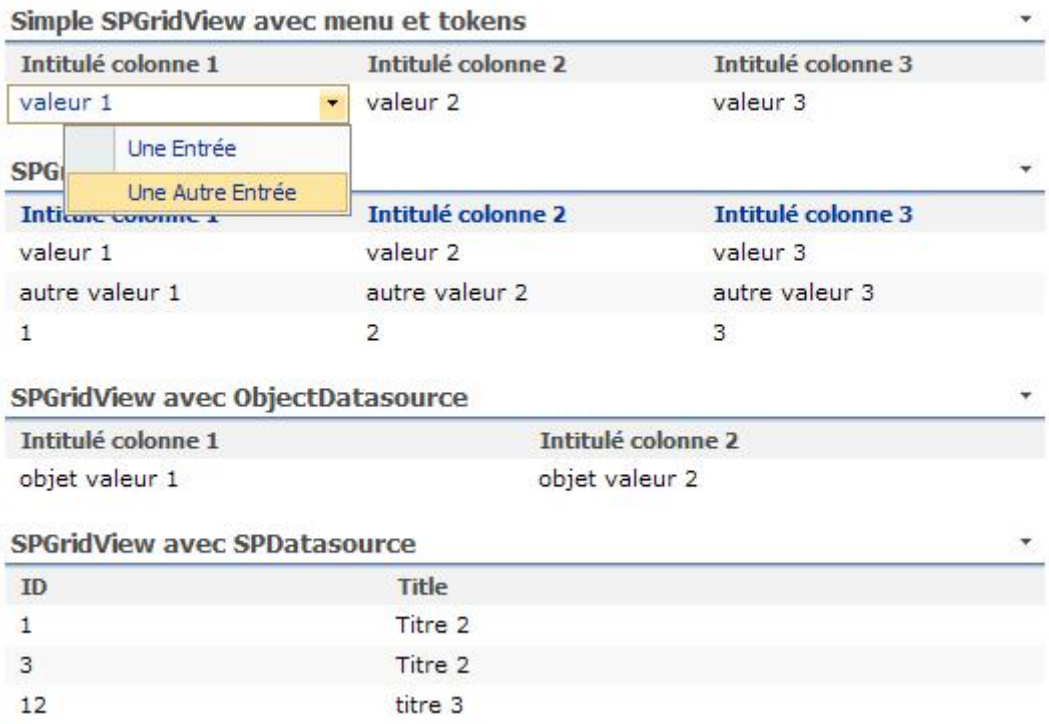
Le SPGridView est l'équivalent du GridView d'ASP.NET mais spécifique à SharePoint. Ce contrôle permet de générer un rendu graphique très proche de ce que SharePoint fait en standard. En outre, les fonctionnalités disponibles au niveau des listes et bibliothèques comme le tri, le regroupement, les filtres et le menu ECB peuvent être plus facilement implémentées avec le SPGridView qu'avec le GridView.

Il est donc très intéressant de s'attarder sur ce contrôle. Voici quelques copies d'écran des webparts que vous pourrez récupérer en téléchargeant le projet.

Tous les webparts avec les exemples expliqués dans l'article :



Quelques exemples de ces webparts déployés :



III - Le SPGridView dans sa plus simple expression

Les quelques lignes de code suivantes représentent l'expression la plus basique de l'utilisation d'un SPGridView dans un webpart SharePoint.

```
protected override void CreateChildControls()
{
  SPGridView SimpleGrid = new SPGridView();
  DataTable Donnees = new DataTable();
  Donnees.Columns.Add("Colonne 1");
  Donnees.Columns.Add("Colonne 2");
  Donnees.Columns.Add("Colonne 3");
  Donnees.Rows.Add("valeur 1", "valeur 2", "valeur 3");
  SimpleGrid.AutoGenerateColumns = false;
  BoundField Colonne1 = new BoundField();
  Colonne1.DataField = "Colonne 1";
  Colonne1.HeaderText = "Intitulé colonne 1";
  SimpleGrid.Columns.Add(Colonne1);
  BoundField Colonne2 = new BoundField();
  Colonne2.DataField = "Colonne 2";
  Colonne2.HeaderText = "Intitulé colonne 2";
  SimpleGrid.Columns.Add(Colonne2);
  BoundField Colonne3 = new BoundField();
  Colonne3.DataField = "Colonne 3";
  Colonne3.HeaderText = "Intitulé colonne 3";
  SimpleGrid.Columns.Add(Colonne3);

  SimpleGrid.DataSource = Donnees;
  SimpleGrid.DataBind();
  Controls.Add(SimpleGrid);
  base.CreateChildControls();
}
```

Ces quelques lignes de code donne ce résultat :

Vous remarquerez que le rendu correspond au rendu habituel des listes et bibliothèques de SharePoint.

IV - Le SPGridView avec un menu ECB

IV-A - Exemple Simple

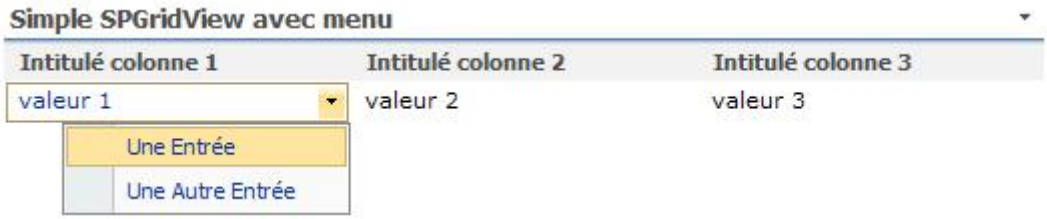
L'un des avantages du SPGridView réside dans le fait qu'il est facile de lui associer un menu ECB (Edit Control Block) et de singer ainsi les composants SharePoint standards.

En reprenant quasiment le même code que précédemment :

```
protected override void CreateChildControls()
{
  SPGridView SimpleGrid = new SPGridView();
  DataTable Donnees = new DataTable();
  Donnees.Columns.Add("Colonne 1");
  Donnees.Columns.Add("Colonne 2");
  Donnees.Columns.Add("Colonne 3");
  Donnees.Rows.Add("valeur 1", "valeur 2", "valeur 3");
  SimpleGrid.AutoGenerateColumns = false;

//ici commence la gestion du menu
  MenuTemplate ListMenu = new MenuTemplate();
  ListMenu.ID = "LeMenu";
  MenuItemTemplate EntreeMenu = new MenuItemTemplate("Une Entrée");
  EntreeMenu.ClientOnClickNavigateUrl = "http://developpez.net/forums";
  ListMenu.Controls.Add(EntreeMenu);
  EntreeMenu = new MenuItemTemplate("Une Autre Entrée");
  EntreeMenu.ClientOnClickNavigateUrl = "http://developpez.net/forums";
  ListMenu.Controls.Add(EntreeMenu);
  Controls.Add(ListMenu);
  SPMenuField Colonne1 = new SPMenuField();
  Colonne1.HeaderText = "Intitulé colonne 1";
  Colonne1.TextFields = "Colonne 1";
  Colonne1.MenuTemplateId = "LeMenu";
  SimpleGrid.Columns.Add(Colonne1);
//fin menu
  BoundField Colonne2 = new BoundField();
  Colonne2.DataField = "Colonne 2";
  Colonne2.HeaderText = "Intitulé colonne 2";
  SimpleGrid.Columns.Add(Colonne2);
  BoundField Colonne3 = new BoundField();
  Colonne3.DataField = "Colonne 3";
  Colonne3.HeaderText = "Intitulé colonne 3";
  SimpleGrid.Columns.Add(Colonne3);
  SimpleGrid.DataSource = Donnees;
  SimpleGrid.DataBind();
  Controls.Add(SimpleGrid);
  base.CreateChildControls();
}
```

Vous obtenez le résultat suivant :



à savoir les mêmes données avec deux entrées dans un menu ECB. Dans cet exemple, les deux points de menu redirigent le visiteur vers http://developpez.net/forums. Nous verrons ultérieurement comment exploiter la gestion des URL.

IV-B - Afficher/Masquer des entrées de menu

Il peut être utile d'afficher ou de masquer une entrée de menu en fonction des données présentes dans les colonnes. Imaginons par exemple le scénario suivant :

Vous utilisez un SPGridView pour afficher une liste de rapports terminés ou en cours de production. Vous souhaitez afficher l'option Visualiser le rapport uniquement pour les rapports dont le statut est terminé.

Pour ce petit exemple, je vais poster la totalité du code car certaines choses changent. Vous aurez des explications après le listing.

```
public class SimpleGridAvecMenuVariable : System.Web.UI.WebControls.WebParts.WebPart
{
  const string STATUTTERMINE = "En Cours";
  const string COLONNESTATUT = "Statut";
  const string COLONNETITRE = "Titre";

  MenuItemTemplate EntreeMenu = null;

  protected override void CreateChildControls()
  {
    SPGridView SimpleGrid = new SPGridView();
    DataTable Donnees = new DataTable();
    Donnees.Columns.Add(COLONNETITRE);
    Donnees.Columns.Add(COLONNESTATUT);
    //ajout des rapports, deux en cours et deux terminé
    Donnees.Rows.Add("Rapport 1", "En Cours");
    Donnees.Rows.Add("Rapport 2", "Terminé");
    Donnees.Rows.Add("Rapport 3", "En Cours");
    Donnees.Rows.Add("Rapport 4", "Terminé");

    SimpleGrid.AutoGenerateColumns = false;
    //Gestion de l'évènement RowDataBound, c'est là que l'on va tester
    //si le statut du rapport est "Terminé" ou pas.
    SimpleGrid.RowDataBound += new GridViewRowEventHandler(SimpleGrid_RowDataBound);
    MenuTemplate ListMenu = new MenuTemplate();
    ListMenu.ID = "LeMenu";
    EntreeMenu = new MenuItemTemplate("Visualiser Rapport");
    EntreeMenu.ClientOnClickNavigateUrl = "http://developpez.net/forums";
    ListMenu.Controls.Add(EntreeMenu);
    MenuItemTemplate AutreEntree = new MenuItemTemplate("Autre EntrÃ©e");
    AutreEntree.ClientOnClickNavigateUrl = "http://developpez.net/forums";
    ListMenu.Controls.Add(AutreEntree);
    Controls.Add(ListMenu);
    SPMenuField TitreRapport = new SPMenuField();
    TitreRapport.HeaderText = COLONNETITRE;
    TitreRapport.TextFields = COLONNETITRE;
    TitreRapport.MenuTemplateId = "LeMenu";
    SimpleGrid.Columns.Add(TitreRapport);
    BoundField Colonne2 = new BoundField();
    Colonne2.DataField = COLONNESTATUT;
    Colonne2.HeaderText = COLONNESTATUT;
    SimpleGrid.Columns.Add(Colonne2);
```

```
    SimpleGrid.DataSource = Donnees;
    SimpleGrid.DataBind();
    Controls.Add(SimpleGrid);
    base.CreateChildControls();
  }

  void SimpleGrid_RowDataBound(object sender, GridViewRowEventArgs e)
  {
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
      string Statut = (string)DataBinder.Eval(e.Row.DataItem, COLONNESTATUT);
      //le SPMenuField se trouve à l'index 0 puisque c'est la première colonne
      Microsoft.SharePoint.WebControls.Menu CtrlMenu =
      e.Row.Cells[0].Controls[0] as Microsoft.SharePoint.WebControls.Menu;

      if (Statut == STATUTTERMINE)
      {
        CtrlMenu.HiddenMenuItems.Add(EntreeMenu);
      }
    }
  }
}
```

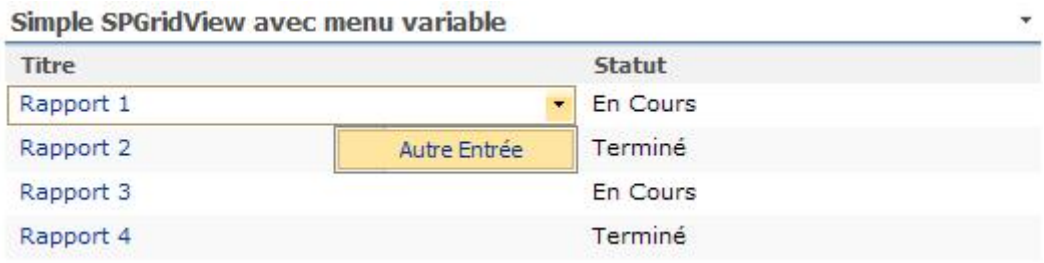
Tout d'abord, trois constantes sont déclarées. L'entrée du menu que l'on souhaite afficher ou masquer en fonction du statut du rapport est également déclarée comme membre de classe car elle est partagée entre les méthodes **CreateChildControls()** et **SimpleGrid_RowDataBound()**.

Ensuite, dans la méthode **CreateChildControls()**, une datatable est remplie avec quatre rapports dont deux sont en statut **Terminé** et deux autres en status **En Cours**.

On spécifie que l'on gère l'évènement **RowDataBound** et on crée le menu comme précédemment. Dans la méthode **SimpleGrid_RowDataBound**, on pointe sur la colonne de type **SPMenuField** et on vérifie la valeur de la colonne **Statut**. En fonction du résultat, on ajoute l'entrée de menu dans la collection des éléments cachés symbolisés par **HiddenMenuItems**.

Ceci donne donc les résultats suivants :

Pour un rapport en cours, on ne voit qu'une des deux options :



Pour un rapport terminé, on voit bien les deux options :

IV-C - Utilisation de Tokens dans les URL

Jusqu'à présent, nous avions utilisé http://developpez.net.forums/ comme URL de destination des entrées menu ajoutées à notre menu ECB. Bien que nous apprécions tous ce site :), l'objectif est plutôt d'utiliser des URL construites en passant en paramètres certaines valeurs des colonnes du SPGridView.

Pour atteindre cet objectif, vous disposez des propriétés suivantes :

Propriété	Description
TokenNameAndValueFields	Permet de définir des tokens réutilisables dans les URL et d'y associer des colonnes
ClientOnClickNavigateUrl	Permet de spécifier l'URL associée à un élément du menu ECB. Cette propriété peut se combiner avec TokenNameAndValueFields
NavigateUrlFields	Permet de spécifier quels sont les champs requis pour construire l'URL lorsque l'on clique sur le lien principal (pas une entrée du menu ECB)
NavigateUrlFormat	Permet de définir le format de l'URL. Cette propriété se combine aec NavigateUrlFields

Lorsque vous définissez votre SPGridView, vous pouvez utiliser les propriétés comme suit :

```
SPGridView SimpleGrid = new SPGridView();
DataTable Donnees = new DataTable();
Donnees.Columns.Add("Colonne 1");
Donnees.Columns.Add("Colonne 2");
Donnees.Rows.Add("valeur 1", "valeur 2");
SimpleGrid.AutoGenerateColumns = false;
MenuTemplate ListMenu = new MenuTemplate();
ListMenu.ID = "LeMenu";
MenuItemTemplate EntreeMenu = new MenuItemTemplate("Une Entrée");
//ici on construit une URL dynamique avec les Tokens %COL1% et %COL2% que l'on
//définit plus loin
EntreeMenu.ClientOnClickNavigateUrl = "?Col1=%COL1%&Col2=%COL2%";
ListMenu.Controls.Add(EntreeMenu);
EntreeMenu = new MenuItemTemplate("Une Autre Entrée");
//Même chose
EntreeMenu.ClientOnClickNavigateUrl = "?Col1=%COL1%&Col2=%COL2%";
ListMenu.Controls.Add(EntreeMenu);
Controls.Add(ListMenu);
SPMenuField Colonne1 = new SPMenuField();
Colonne1.HeaderText = "Intitulé colonne 1";
Colonne1.TextFields = "Colonne 1";
Colonne1.MenuTemplateId = "LeMenu";
//ici on spécifie que Colonne 1 et Colonne 2 correspondent respectivement à
//{0} et {1} c'est à dire les paramètres utilisés à la ligne suivante
Colonne1.NavigateUrlFields = "Colonne 1,Colonne 2";
//On spécifie le format de notre URL avec les paramètres
Colonne1.NavigateUrlFormat = "?Col1={0}&Col2={1}";
//On spécifie les tokens %COL1% et %COL2% utilisés dans ClientOnClickNavigateUrl
Colonne1.TokenNameAndValueFields = "COL1=Colonne 1,COL2=Colonne 2";

SimpleGrid.Columns.Add(Colonne1);
BoundField Colonne2 = new BoundField();
Colonne2.DataField = "Colonne 2";
Colonne2.HeaderText = "Intitulé colonne 2";
SimpleGrid.Columns.Add(Colonne2);
SimpleGrid.DataSource = Donnees;
SimpleGrid.DataBind();
Controls.Add(SimpleGrid);
base.CreateChildControls();
```

Les commentaires insérés dans le code sont suffisamment explicites.

V - Utiliser le tri dans un SPGridView

La gestion du tri dans le SPGridView n'est pas très complexe. Au niveau de la méthode **CreateChildControls()**, il suffit de spécifier que le contrôle autorise le tri et de lui associer un évènement de tri.

Dans notre cas, nous utilisions une DataTable pour lier le SPGridView aux données. Il faut à présent utiliser une DataView(ou tout autre objet permettant de trier les données) qui permet d'appliquer un ordre de tri et d'associer cette DataView au SPGridView, comme ceci :

```
....
VueDonnees = Donnees.DefaultView;
SimpleGrid.Sorting += new GridViewSortEventHandler(SimpleGrid_Sorting);
SimpleGrid.AllowSorting = true;
SimpleGrid.DataSource = VueDonnees;
SimpleGrid.DataBind();
```

VueDonnees est une DataView déclarée en tant que membre de classe. Ensuite, dans la méthode gérant l'évènement de tri, vous faites comme suit :

```
void SimpleGrid_Sorting(object sender, GridViewSortEventArgs e)
{
  ColonneTri = e.SortExpression;
  VueDonnees.Sort =
  string.Format("{0} {1}",
                e.SortExpression,
                OrdreTri);
  SimpleGrid.DataBind();
}
```

Cette méthode récupère la colonne de tri et vérifie par le biais de propriétés quelle était la précédente colonne sujette à un tri et dans quelle direction. Voici les propriétés en question :

```
public string NouvelOrdreTri
{
  set
  {
    ViewState["OrdreTri"] = value;
  }
  get
  {
    if (ViewState["OrdreTri"] != null)
      return ViewState["OrdreTri"].ToString();
    else
      return "asc";
  }
}
public string OrdreTri
{
  get
  {
    if (ViewState["ColonneTri"] != null &&
     ViewState["ColonneTri"].ToString() == ColonneTri)
    {
      string Tri = (NouvelOrdreTri == "asc") ? "desc" : "asc";
      NouvelOrdreTri = Tri;
      return Tri;
 }
    else
    {
      ViewState["ColonneTri"] = ColonneTri;
      return "asc";
    }
  }
}

public string ColonneTri
{
  get
  {
    return _ColonneTri;
  }
  set
  {
```

```
      _ColonneTri = value;
    }
  }
```

VI - SPGridView et binding

Outre le binding avec une datatable ou un dataset, vous pouvez lier le SPGridView à d'autres objets.

VI-A - Liaison avec un SPDataSource

Si vous souhaitez afficher des données issue d'une liste SharePoint, le plus simple est probablement d'utiliser **SPDataSource**. Voici un exemple permettant d'afficher les documents de la bibliothèque **Documents** du site où est déployé le webpart.

```
protected override void CreateChildControls()
{
 SPGridView SimpleGrid = new SPGridView();
 SPDataSource Donnees = new SPDataSource();
 Donnees.List = SPContext.Current.Web.Lists["Documents"];
 Donnees.Scope = SPViewScope.Recursive;
 SimpleGrid.AutoGenerateColumns = false;
 BoundField ColonneID = new BoundField();
 ColonneID.DataField = "ID";
 ColonneID.HeaderText = "ID";
 SimpleGrid.Columns.Add(ColonneID);
 BoundField ColonneTitre = new BoundField();
 ColonneTitre.DataField = "Title";
 ColonneTitre.HeaderText = "Title";
 SimpleGrid.Columns.Add(ColonneTitre);
 SimpleGrid.DataSource = Donnees;
 SimpleGrid.DataBind();
 Controls.Add(SimpleGrid);
 base.CreateChildControls();
}
```

Ce qui donne ceci :

SPGridView avec SPDatasource

ID	Title
1	Titre 2
3	Titre 2
12	titre 3

En plus de sa facilité évidente, SPDataSource permet également d'effectuer des requêtes plus rafinées via sa propriété **SelectCommand** à laquelle il est possible de transmettre une requête CAML.

VI-B - Liaison avec un Object Datasource

Vous pouvez également encapsuler votre accès aux données dans une DAL et lier le SPGridView à cet objet. En voici une illustration simple :

Code de la classe qui génère les données :

```
public class ExempleObjectDataSource
{
  public DataTable DonneesGrid()
  {
    DataTable Donnees = new DataTable();
```

```
    Donnees.Columns.Add("Colonne1");
    Donnees.Columns.Add("Colonne2");
    Donnees.Rows.Add("objet valeur 1", "objet valeur 2");
    return Donnees;
  }
}
```

Enfin, le code à utiliser dans le webpart.

```
protected override void CreateChildControls()
{
  SPGridView SimpleGrid = new SPGridView();
  ObjectDataSource Donnees = new ObjectDataSource(
   "DVP.ExemplesSPGridView.ExempleObjectDataSource, DVP.ExemplesSPGridView, ...
 PublicKeyToken=59e820c1f1877d1c",
  "DonneesGrid");
  Donnees.ID = "ObjetDonees";
  Controls.Add(Donnees);
  SimpleGrid.AutoGenerateColumns = false;
  SimpleGrid.ID = "GridAvecObjet";
  BoundField Colonne1 = new BoundField();
  Colonne1.DataField = "Colonne1";
  Colonne1.HeaderText = "Colonne1";
  SimpleGrid.Columns.Add(Colonne1);
  BoundField Colonne2 = new BoundField();
  Colonne2.DataField = "Colonne2";
  Colonne2.HeaderText = "Colonne2";
  SimpleGrid.Columns.Add(Colonne2);
  SimpleGrid.DataSourceID = "ObjetDonees";
  Controls.Add(SimpleGrid);
  SimpleGrid.DataBind();
  base.CreateChildControls();
}
```

Où il suffit d'instancier un nouveau ObjectDataSource en référencant la signature de l'assemblage (incomplète dans l'exemple), et de spécifier la méthode qui renvoie les données. Ensuite, la construction du SPGridView se fait comme d'habitude.

Le résultat est donc le suivant :

SPGridView avec ObjectDatasource

Colonne1	Colonne2
objet valeur 1	objet valeur 2
ligne 2	ligne 2 col 2

VII - Utiliser les filtres dans un SPGridView

Pour utiliser les filtres dans le SPGridView, vous pouvez simplement réutiliser l'exemple avec l'ObjectDataSource. Etant donné que presque tout est exactement identique, voici les seules variations entre les deux :

Dans la méthode **CreateChildControls()**, vous devez ajouter ces trois lignes :

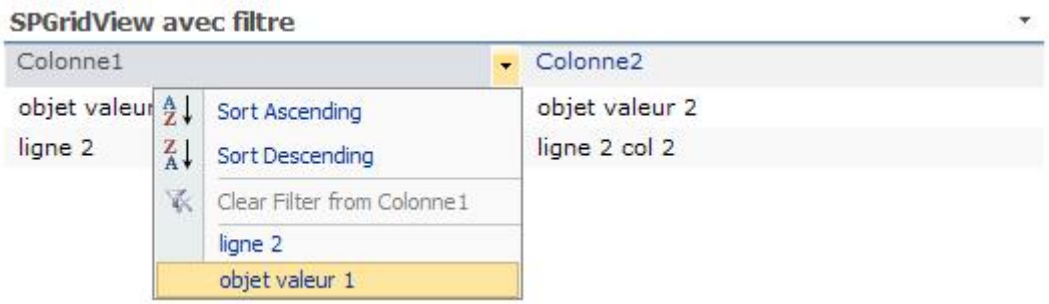
```
SimpleGrid.AllowFiltering = true;
SimpleGrid.FilterDataFields = "Colonne1,Colonne2";
SimpleGrid.FilteredDataSourcePropertyName = "FilterExpression";
SimpleGrid.FilteredDataSourcePropertyFormat = "{1}='{0}'";
```

Vous spécifiez que le SPGridView peut-être filtré. Vous spécifiez également que le nom de la propriété du contrôle source, en l'occurrence l'objet **Donnees** est **FilterExpression**. Ensuite, vous spécifiez la condition de filtre via **FilteredDataSourcePropertyFormat**. Cette propriété supporte notamment des opérateurs tels que LIKE.

Enfin l'autre différence est que vous devez appeler la méthode **DataBind()** dans l'évènement **OnPreRender** comme ceci :

```
protected override void OnPreRender(EventArgs e)
{
  SimpleGrid.DataBind();
  base.OnPreRender(e);
}
```

Le résultat est intéressant puisqu'il vous permet d'effectuer des filtres de la même manière que les listes et bibliothèques standard :



VIII - Utiliser le regroupement dans un SPGridView

Enfin, il est également possible de bénéficier du regroupement. Toujours en repartant de l'exemple avec l'ObjectDataSource. Vous devez ajouter ces lignes dans la méthode **CreateChildControls()**.

```
SimpleGrid.AllowGrouping = true;
SimpleGrid.GroupField = "Colonne1";
SimpleGrid.GroupDescriptionField = "Colonne1";
SimpleGrid.GroupFieldDisplayName = "Colonne1";
```

Ce qui donne :

IX - Créer une toolbar dans un SPGridView

Paul Robinson a écrit un post sur son **blog** qui décrit comment ajouter une barre d'outils à un WebPart. Suivant une suggestion de Dieudoné N'TAMACK, il me paraissait intéressant d'en parler aussi dans cet article. Donc on va simplement reprendre l' exemple avec quelques mots d'explication.

```
protected override void CreateChildControls()
```

```
{
  SimpleGrid = new SPGridView();
  DataTable Donnees = new DataTable();
  Donnees.Columns.Add("Colonne 1");
  Donnees.Columns.Add("Colonne 2");
  Donnees.Columns.Add("Colonne 3");
  Donnees.Rows.Add("valeur 1", "valeur 2", "valeur 3");
  SimpleGrid.AutoGenerateColumns = false;
  BoundField Colonne1 = new BoundField();
  Colonne1.DataField = "Colonne 1";
  Colonne1.HeaderText = "Intitulé colonne 1";
  SimpleGrid.Columns.Add(Colonne1);
  BoundField Colonne2 = new BoundField();
  Colonne2.DataField = "Colonne 2";
  Colonne2.HeaderText = "Intitulé colonne 2";
  SimpleGrid.Columns.Add(Colonne2);
  BoundField Colonne3 = new BoundField();
  Colonne3.DataField = "Colonne 3";
  Colonne3.HeaderText = "Intitulé colonne 3";
  SimpleGrid.Columns.Add(Colonne3);

  SimpleGrid.DataSource = Donnees;
  SimpleGrid.DataBind();
  //ici l'ajout de la barre d'outils
  ToolBarButton Bouton =
                (ToolBarButton)Page.LoadControl("~/_controltemplates/ToolBarButton.ascx");
  Bouton.Text = "Un Bouton";
  Bouton.ImageUrl = "/_layouts/images/TABGEN.gif";
  Bouton.Click += new EventHandler(Bouton_Click);
  ToolBar BarreOutils = (ToolBar)Page.LoadControl("~/_controltemplates/ToolBar.ascx");
  BarreOutils.Buttons.Controls.Add(Bouton);
  Controls.Add(BarreOutils);
  Controls.Add(SimpleGrid);
  base.CreateChildControls();
}
```

Les contrôles utilisateurs **ToolBarButton.ascx** et **ToolBar.ascx** sont inclus dans SharePoint. Ils représentent bien sûr respectivement un bouton et une barre d'outils. Ils sont tous deux présents dans le répertoire 12\TEMPLATE \CONTROLTEMPLATES. En chargeant ces contrôles dans la méthode **CreateChildControls()** du SPGridView mais également de n'importe quel webpart, SharePoint accole la barre d'outils automatiquement au webpart en question. En fonction de l'ordre dans lequel vous ajoutez les contrôles, la barre d'outils sera en haut ou en bas du webpart.

Vous pouvez facilement créer vos propres contrôles utilisateurs si vous souhaitez disposer de barres d'outils ou de boutons particuliers. C'est d'ailleurs ce que nous allons faire :). Pour illustrer cet exemple, imaginons le scénario suivant :

Vous souhaitez disposer d'un bouton très similaire au bouton standard mais celui-ci ne doit s'afficher que pour les utilisateurs authentifiés. Donc, en mode anonyme, il ne doit jamais s'afficher.

Tout d'abord, créons la classe dérivant de ToolbarButton :

```
public class MyToolbarButton : ToolBarButton
{
  public override void RenderControl(HtmlTextWriter writer)
  {
    if (SPContext.Current.Web.CurrentUser != null)
      base.RenderControl(writer);
  }

}
```

Ensuite, Il suffit de procéder à ces quelques étapes :

- Localisez le fichier ToolbarButton.ascx présent dans 12\TEMPLATE\CONTROLTEMPLATES
- Copiez-le et nommez la copie MyToolbarButton.ascx
- Editez-le et modifiez la classe dont il dérive

Dans le cas de notre exemple, il faut remplacer la valeur de l'attribut **Inherits** par :

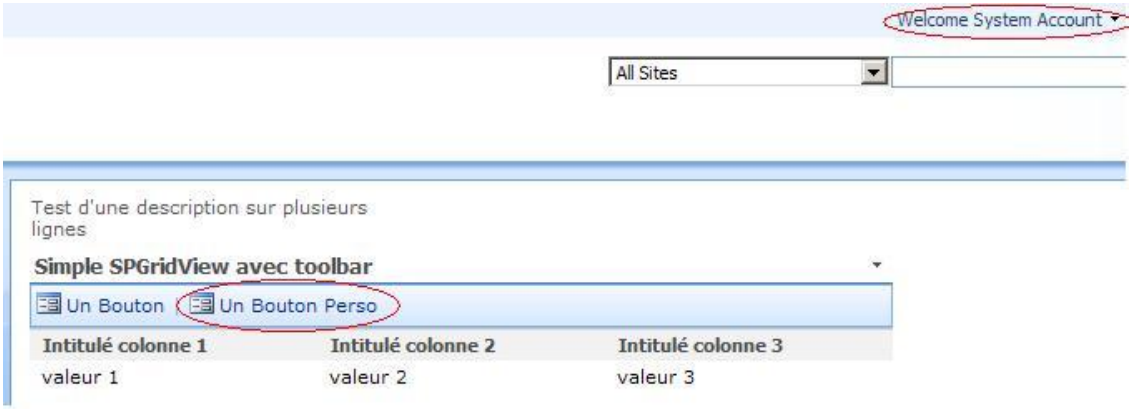
```
DVP.ExemplesSPGridView.MyToolbarButton, DVP.ExemplesSPGridView, Version=1.0.0.0, Culture=neutral,
 PublicKeyToken=59e820c1f1877d1c
```

Ensuite, en se basant sur l'exemple précédent, vous pouvez ajouter les quelques lignes de code suivantes :

```
MyToolbarButton BoutonPerso =
  (MyToolbarButton)Page.LoadControl("~/_controltemplates/MyToolBarButton.ascx");
BoutonPerso.Text = "Un Bouton Perso";
BoutonPerso.Click += new EventHandler(Bouton_Click);
BoutonPerso.ImageUrl = "/_layouts/images/TABGEN.gif";
BarreOutils.Buttons.Controls.Add(BoutonPerso);
```

Ceci vous donnera le résultat suivant :

En mode authentifié :



En mode anonyme :

X - Conclusion

Le SPGridView est un contrôle vraiment très intéressant. Pour faciliter la compréhension des exemples, j'ai isolé toutes les fonctionnalités dans des classes séparées. Il va sans dire que vous pouvez bien sûr toutes les utiliser en même temps. En plus de ce qui est décrit dans ce tutoriel, il est possible de l'utiliser dans des contrôles utilisateurs et d'utiliser la pagination et de le coupler à l'AJAX, cela fera peut-être l'objet d'un futur article.

XI - Téléchargement

Le projet est disponible en téléchargement **ici**.