Le composant DataGridView

1. Présentation

Le composant représente une collection de lignes comprenant une collection de colonnes autrement dit un tableau.

Date	Nombre de jours	Motif
06/03/2020	1	Arrêt de maladie
10/03/2020	3	Accident du travail
17/03/2020	3	Mariage
23/03/2020	3	Décès conjoint

L'opérateur d'indexation [numColonne, numLigne] donne accès à une cellule du tableau.
La numérotation des colonnes et des lignes commence à 0.

Pour accéder à la valeur d'une cellule :

- dgvA**[numColonne, numLigne]**.Value
- ou unDgv.**Rows[numLigne].Cells[numColonne].**Value = "new value"

Pour remplir le composant on peut lui associer une source de données (propriété dataSource) ou lui ajouter des lignes (méthodes Rows.Add(valeur colonne1, … caleur colonne n)

Pour parcourir un dataGridView

```
foreach (DataGridViewRow uneLigne in unDgv.Rows) {
        uneVariable = (maClasse) uneLigne.Cells[0].Value;
        ….
}
```

2. Propriété et méthodes concernant la gestion des lignes du composant

Propriétés et méthodes	Type	Rôle
Rows.Add()	void	Ajoute une nouvelle ligne vide;
Rows.Insert(p, ligne)	void	Insère une nouvelle ligne (objet DataGridViewRow) en position p;
Rows.Remove(ligne)	void	Supprime une ligne (objet DataGridViewRow)
Rows.RemoveAt(numLigne)	void	Supprime la ligne d'index numLigne
Rows.Clear()	void	Efface toutes les lignes
Rows[n].Selected	bool	Sélectionner ou désélectionner la ligne
SelectedRows	Colection	Retourne la collection des lignes sélectionnées SelectedRows[0] retourne la première ligne séléctionnée.
CurrentRow.Index	int	Retourne le numéro de la ligne courante.
RowCount	int	retourne le nombre de lignes
DataSouce	Collection	Objet de type collection utiliser pour alimenter le composant
HitTest(x, y)		Retourne les informations relatives à l'emplacement, telles que les indices de ligne et de colonne, en fonction des coordonnées x et y de la souris.
Sort(Colonne, ordre)	void	Permet de trier sur une colonne. unDgv.Sort(unDgv.Columns[0], ListSortDirection.Ascending); Nécessite l'espace de nom System.ComponentModel

Liste complète : http://msdn.microsoft.com/fr-fr/library/system.windows.forms.datagridview.rows(v=vs.110).aspx
Pour le dimensionnement des colonnes : https://msdn.microsoft.com/fr-fr/library/74b2wakt(v=vs.110).aspx

3. Les méthodes et propriétés liées à la configuration du composant

Toutes ces propriétés peuvent se trouver dans une méthode de l'application chargée de configurer le composant :

```
private void parametrerDgv(DataGridView unDgv)
{
    paramètrage concernant le datagridview dans son ensemble

    paramètrage concernant la ligne d'entête (les entêtes de chaque colonnes)

    paramètrage concernant l'entête de ligne (la colonne d'entête ou le sélecteur)

    paramètrage au niveau des lignes

    paramètrage au niveau des cellules

    paramètrage au niveau de la zone sélectionnée

    paramètrage des colonnes
}
```

En passant le composant Cela permet d'utiliser au mieux la technique de copier, coller, adapter

Exemple

```
// Accessibilité : doit rester à true si on veut pouvoir utiliser les barres de défilement
 unDgv.Enabled = true;

// style de bordure
unDgv.BorderStyle = BorderStyle.FixedSingle;

// couleur de fond
unDgv.BackgroundColor = Color.White;

// couleur de texte
unDgv.ForeColor = Color.Black;

// police de caractères par défaut
unDgv.DefaultCellStyle.Font = new Font("Georgia", 11);

// mode de sélection dans le composant : FullRowSelect, CellSelect ...
unDgv.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

// sélection multiple
unDgv.MultiSelect = false;

// l'utilisateur peut-il ajouter ou supprimer des lignes ?
unDgv.AllowUserToDeleteRows = false;
unDgv.AllowUserToAddRows = false;

// L'utilisateur peut-il modifier le contenu des cellules ou est-elle réservée à la programmation ?
unDgv.EditMode = DataGridViewEditMode.EditProgrammatically;

// l'utilisateur peut-il redimensionner les colonnes et les lignes ?
unDgv.AllowUserToResizeColumns = false;
unDgv.AllowUserToResizeRows = false;
```

```
// l'utilisateur peut-il modifier l'ordre des colonnes ?
unDgv.AllowUserToOrderColumns = false;

// le composant accepte t'il le 'déposer' dans un Glisser - Déposer ?
unDgv.AllowDrop = false;
```

La ligne d'entête (les entêtes de chaque colonne)

```
// visibilité
 unDgv.ColumnHeadersVisible = true;

// bordure
unDgv.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.None;

// style  [à adapter] (ici : noir sur fond transparent sans mise en évidence de la sélection)
unDgv.EnableHeadersVisualStyles = false;
DataGridViewCellStyle style = unDgv.ColumnHeadersDefaultCellStyle;
style.BackColor = Color.WhiteSmoke;
style.ForeColor = Color.Black;
style.SelectionBackColor = Color.WhiteSmoke;    // même couleur que backColor pour ne pas mettre en évidence la colonne sélectionnée
style.SelectionForeColor = Color.Black;
style.Alignment = DataGridViewContentAlignment.MiddleLeft;
style.Font = new Font("Georgia", 12, FontStyle.Bold);

// hauteur
unDgv.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.EnableResizing;
unDgv.ColumnHeadersHeight = 40;
```

La colonne de sélection (première colonne ou les entête de chaque ligne)

```
// visible
unDgv.RowHeadersVisible = false;

// style de bordure
unDgv.RowHeadersBorderStyle = DataGridViewHeaderBorderStyle.None;
```

Au niveau des lignes

```
// Hauteur
unDgv.RowTemplate.Height = 30;
```

Au niveau des cellules

```
// style de bordure
unDgv.CellBorderStyle = DataGridViewCellBorderStyle.None;

// couleur de fond, ne pas utiliser transparent
unDgv.RowsDefaultCellStyle.BackColor = Color.White;

// Couleur alternative appliquée une ligne sur deux
unDgv.AlternatingRowsDefaultCellStyle.BackColor = Color.FromArgb(255, 238, 238, 238);
```

Au niveau de la zone sélectionnée

```
// couleur de fond mettre la même que les cellules si on ne veut pas mettre la zone en évidence
unDgv.RowsDefaultCellStyle.SelectionBackColor = System.Drawing.Color.White;
```

```
// couleur du texte
unDgv.RowsDefaultCellStyle.SelectionForeColor = System.Drawing.Color.Black;

Au niveau des colonnes
// Largeur : à contrôler avec la largeur des colonnes si elle est définie
unDgv.Width = 700;

// Nombre de colonne sans compter les colonnes ajoutées par la méthode Add
unDgv.ColumnCount = 4;

// faut-il ajuster automatiquement la taille des colonnes à leur contenu (commenter la ligne si non)
unDgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

// faut-il ajuster automatiquement la taille des colonnes par un ajustement proportionnel à la
largeur totale (commenter la ligne si non)
unDgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

// description de chaque colonne  [partie à personnaliser] : visibilité, largeur, alignement cellule et
entête si elle ne correspond pas à la valeur par défaut
unDgv.Columns[0].HeaderText = "programmée le";
unDgv.Columns[0].Name = "Date";
unDgv.Columns[0].Width = 200;
unDgv.Columns[0].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;

unDgv.Columns[1].HeaderText = "à";
unDgv.Columns[1].Name = "Heure";
unDgv.Columns[1].Width = 50;
unDgv.Columns[1].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
unDgv.Columns[2].HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

…

// faut-il désactiver le tri sur toutes les colonnes ? (commenter les lignes si non)
for (int i = 0; i < unDgv.ColumnCount; i++)
      unDgv.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;
```

4. Autres opérations sur un DataGridView

Adapter la hauteur du datagridView par rapport à son contenu

```
unDgv.Height = lesRendezVous.Count * (unDgv.RowTemplate.Height) +
unDgv.ColumnHeadersHeight + 10;
if (unDgv.Height > 800) unDgv.Height = 800;
```

Interdire la modification d'une colonne

```
unDgv.Columns[0].ReadOnly = true;
```

Mise en forme multiligne pour une colonne

```
unDgv.Columns[1].DefaultCellStyle.WrapMode = DataGridViewTriState.True;
```

Masquer une colonne du DataGridView

```
unDgv.Columns["nomColonne" ou index].Visible = false;
```

Définir la couleur de fond par défaut pour une ligne et la couleur de fond alternative (1/2)

```
unDgv.AlternatingRowsDefaultCellStyle.BackColor = Color.FromArgb(255, 238, 238, 238);
unDgv.RowsDefaultCellStyle.BackColor = Color.White;
```

Centrer une cellule d'entête

```
unDgv.Columns[2].HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;
```

Alimentation d'une dataGridView ligne par ligne

```
 foreach(Etudiant unEtudiant in lesEtudiants) {
       unDgv.Rows.Add(unEtudiant.Id, unEtudiant.Nom, unEtudiant.Prenom,
unEtudiant.Option);
 }
```

Changer la couleur de fond d'une ligne en fonction d'une condition

```
int ligne = 0;
bool surligner;
foreach(Etudiant unEtudiant in lesEtudiants) {
   unDgv.Rows.Add(unEtudiant.Id, unEtudiant.Nom, unEtudiant.Prenom, unEtudiant.Option);
   surligner =   unEtudiant.Option= "SLAM"
    if (surligner)
       for (int i = 0; i <= 3; i++)
           unDgv.Rows[ligne].Cells[i].Style.BackColor = Color.Red;
```

Attention aux incompatibilité avec le style appliqué sur la ligne sélectionnée

Ajouter une colonne sous la forme d'une zone de liste

```
DataGridViewComboBoxColumn uneColonneListe = new DataGridViewComboBoxColumn();
uneColonneListe.Items.Add("SLAM");
uneColonneListe.Items.Add("SISR");
uneColonneListe.Name = "Option";
uneColonneListe.AutoComplete = true;
unDgv.Columns.Add(uneColonneListe);
```

Ajouter une colonne contenant une case à cocher

```
DataGridViewCheckBoxColumn uneColonneCase = new DataGridViewCheckBoxColumn();
uneColonneBouton.HeaderText = "Conduit";
uneColonneCase.Name = "Conduit";
uneColonneCase.Width = 100;
uneColonneCase.ReadOnly = true;
uneColonneCase.FillWeight = 10;
unDgv.Columns.Add(uneColonneCase);
```

Ajouter une colonne contenant un bouton

```
DataGridViewButtonColumn uneColonneBouton =  new DataGridViewButtonColumn();
uneColonneBouton.HeaderText = "Action";
uneColonneBouton.Name = "Action";
uneColonneBouton.Text = "Modifier";
uneColonneBouton.UseColumnTextForButtonValue = true;
unDgv.Columns.Add(uneColonneBouton);
```

Ajouter une colonne contenant une image

```
DataGridViewImageColumn uneColonne = new DataGridViewImageColumn();
uneColonne.HeaderText = "Supprimer";
uneColonne.Image = new Bitmap("supprimer.png");
unDgv.Columns.Add(uneColonne);
```

Le fichier supprimer.png se trouve dans le répertoire debug de l'application

Si ce type de colonne ne doit pas être placé à la fin (.Add) il est possible de l'insérer :

```
DataGridViewImageColumn colonne = new DataGridViewImageColumn();
colonne.HeaderText = "";
colonne.Image = new Bitmap("supprimer.png");
unDgv.Columns.Insert(1, colonne);
unDgv.Columns[1].Width = 50;
```

5. Les principaux événements

Le clic sur une cellule : CellClick

Cet événement est souvent utilisé pour pouvoir déclencher l'action associée à un bouton placé dans une cellule.
Puisque l'événement se déclenche sur le clic de n'importe quelle cellule, il faut commencer par vérifier que la cellule ayant intercepté le click correspond bien à une cellule contenant un bouton.
Pour cela deux propriétés de l'objet 'e' reçu en paramètre sont utiles :

- RowIndex qui retourne le numéro de la ligne correspondante
- ColumnIndex qui retourne le numéro de la colonne correspondante

```
void dgvA_CellClick(object sender, DataGridViewCellEventArgs e)  {
    if (e.RowIndex < 0 || e.ColumnIndex != 4) return;
    string id = dgvA[0, e.RowIndex].Value.ToString();
    string uneOption = dgvA[3, e.RowIndex].Value.ToString();
    modifierOption(id, uneOption);
  }
```

4 représente ici la cinquième colonne qui contient le bouton modifier
RowIndex peut retourner -1 si l'utilisateur clique sur une entête de colonne

La détection d'un changement de sélection : SelectionChanged

Cet événement peut être utilisé pour mettre à jour l'interface à chaque fois que l'utilisateur sélectionne une ligne du datagridView

```
private void dgvPraticien_SelectionChanged(object sender, EventArgs e) {
      lePraticien = (Praticien)dgvPraticiens.SelectedRows[0].Cells[0].Value;
      remplirPraticien();
}
```

6. Alimentation d'un dataGridView à partir d'une collection

Il est possible de lier un datagridView à une collection.

Il faut cependant que les objets de cette collection possèdent des propriétés. Chaque propriété alimente une colonne du dataGridView

```
unDgv.DataSource = lesEtudiants;
```

La liaison s'effectue par la propriété DataSource. Cependant cette liaison n'est pas dynamique.
Si les données de la collection sont modifiées il faut changer la valeur de la propriété DataSource et la remettre pour l'obliger à relire les données de la source :

```
unDgv.DataSource = "";
unDgv.DataSource = lesEtudiants;
```

Cela oblige aussi à refaire une partie de la mise en forme du composant, notamment la taille des colonnes, la hauteur des lignes.