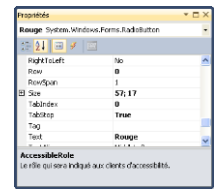
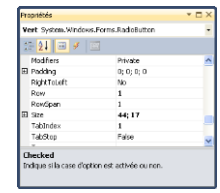
CM4

- Quelques composants
  - Les boutons groupés
  - ListBox, CheckedListBox, ComboBox
  - Liste en arbre (TreeView)
  - Les fenêtres de liste (ListView)
  - La représentation en grille (DataGridView)

Les boutons groupés

- Comment grouper des boutons ?
  - On utilise la propriété TabStop





Les boutons groupés

- Les boutons groupés sont ceux (qui ont des tabstop à faux) compris
  - entre le 1er bouton qui a un TabStop à vrai ,
  - jusqu'au prochain élément qui a un TabStop à vrai
- dans l'ordre des tabulations.
- Cet ordre peut être consulté et modifié par
  - Affichage -> Ordre de tabulation

Les listes standard

- Une boîte de liste standard sert à sélectionner 1 ou plusieurs articles dans une liste.
- Des barres de défilement peuvent être affichées si nécessaire. (par défaut barre verticale)
- Généralement, la liste contient du texte, elle peut être personnalisée et par exemple, contenir des images.

Les listes standard

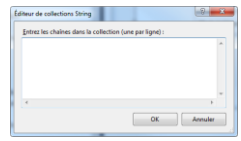
- La hiérarchie de classe est :
- Component -> Control -> ListControl -> ListBox
  - ListControl est une classe abstraite
- Les propriétés d'une liste
  - **MultiColumn (T/F)** vaut vrai si la liste comporte plusieurs colonnes et ColumnWidth est la largeur d'une colonne
  - **DrawMode** indique comment la liste doit être affichée (Normal, OwnerDrawFixed, OwnerDrawVariable)
  - **HorizontalScrollBar (T/F)** si une barre horizontale doit être automatiquement affichée
  - **Sorted (T/F)** indique si la liste est triée

Les listes standard

- Les propriétés d'une liste
  - **Items** collection (des libellés) des articles
  - **SelectionMode** pour fixer le mode de sélection des articles de la liste
    - One, un seul article
    - None, aucun
    - MultiSimple chaque clic sélectionne un nouvel article
    - MultiExtended sélection multiple avec les touches MAJ et CTRL

Les listes standard

- Ajout d'articles dans la liste
  - Par l'éditeur de collections sous Visual



- Par les méthodes
  - int Add(object item);
  - void Add(object []);
  - void AddRange(ObjectCollection)
- Exemple
  - string [] ts={"France", "Italie", …}
  - liste.Items.AddRange(ts);

7

Les listes standard

- Récupération des articles sélectionnés
  - SelectedIndice : indice de l'article sélectionné à partir de 0, -1 si aucun
  - SelectedIndices : collections des indices sélectionnés
  - SelectedItem : article sélectionné
  - SelectedItems : collection des articles
    - foreach (string s in liste.SelectedItems)
      - { …….}
    - string s= (string) liste.Items[i];
    - int c= liste.Items.Count;

8

Les listes standard

- Lés événements usuels
- DoubleClick, pour agir sur un double clic souris sur un article
- KeyPress pour détecter une frappe sur ENTREE sur un article sélectionné

```
private void liste_KeyPress(object sender, KeyPressEventArgs e)
  {
     if (e.KeyChar == 13) {string s = (string) lb2.SelectedItem; …}
  }
```

- SelectedIndexChanged pour détecter le passage d'un article à un autre avec la souris, ou avec les flèches

9

Les listes avec des images

- Pour créer une liste d'images, il faut
  - Spécifier la valeur de DrawMode pour une liste personnalisée
  - Traiter l'événement DrawItem qui assure le dessin de l'article
  - Traiter l'événement DrawMeasure si les images de la liste ont des hauteurs variables.
- Dans tous les cas, il faut remplir la boîte avec les libellés, même si ce libellé n'est pas affiché,
  - il sert à déterminer quelle image doit être affichée

10

Les listes avec des images

- L'événement DrawItem est appelé à chaque fois qu'un article de la liste doit être affiché,
  - Il comporte un argument de type DrawItemEventArgs qui dérive de EventArgs
  - Cet argument comporte plusieurs attributs dont
    - BackColor, ForeColor, Font
    - Graphics pour dessiner
    - Bounds : le rectangle dans lequel doit s'afficher l'article
    - Index : numéro de l'article concerné
    - State (Focus, Disabled ou Selected)

11

Les listes avec des images

- Création d'une boîte de liste dont les articles sont des images (drapeaux)
  - La liste doit être une liste personnalisée
    - liste.DrawMode= DrawMode.OwnerDrawFixed
      - Si les images sont de la même taille
    - liste.DrawMode= DrawMode.OwnerDrawVariable.
      - Si les images sont de tailles variables
  - (dans l'exemple, de taille fixe)

12

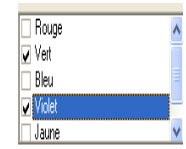
Les listes avec des images

- Création d'une liste d'images (en ressources)
  - ImageList imList = new ImageList();
  - imList.Images.Add(Properties.Resources.France);
  - Etc.
- Ajout des libellés
  - liste.Items.Add("France"),
  - Etc.
- Traitement de l'événement DrawItem qui est appelé lors de l'affichage de la liste (pour chaque article)

```
private void liste_DrawItem(object sender, DrawItemEventArgs e)
{ Point p = e.Bounds.Location // pour dessiner
   imList.Draw(e.Graphics, p, e.Index);                }
// Draw dessine l'article d'indice e.Index, à la position p, avec le
   graphique e.Graphics, e.Index correspond aussi à l'indice dan la liste
```


Listes avec cases à cocher

- La classe CheckedListBox dérive de la classe ListBox
- Elle comporte des propriétés spécifiques qui donnent
  - La collection des indices ou des articles cochés
    - CheckedIndices, CheckedItems
- Des méthodes spécifiques
  - GetItemChecked (int n) -> T/F selon si l'article d'indice n est coché ou non



Listes avec cases à cocher

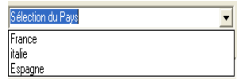
- Exemples de manipulation d'une liste avec cases à cocher
- Création d'une liste
  - CheckedListBox liste = new CheckedListBox();
- Ajout d'un article
  - int n = liste.Items.Add("Rouge");
- Ajout d'un coche pour l'article
  - liste.SetItemCheckedState(n, CheckedState.Checked);
- Récupération des cases cochées
  - CheckedItemCollection coll = liste.CheckedItems;
  - …

Les listes déroulantes

- Une liste déroulante comporte des caractéristiques des zones d'édition, et des listes.
  - La classe ComboBox dérive de la classe ListControl qui dérive de la classe Control
- Elle peut être triée.
- Les propriétés importantes sont
  - Name, Text (qui est le texte de la 1ère ligne affichée)
  - DropDownStyle – style de la liste
    - (Simple, DropDown, *DropDownList*)
  - Sorted – triée ou non
  - Items – collection des articles
  - MaxLength - nombre maximum de caractères dans la zone d'édition

Les listes déroulantes

- Exemple de liste en utilisant les valeurs d'un type énuméré
  - ComboBox liste = new ComboBox();
  - Description du type énuméré
    - public enum Pays {France, Italie, …}
  - Remplissage de la liste
    - liste.DataSource = Enum.GetValues(typeof(Pays));
  - Récupération de l'élément sélectionné
    - Pays p = (Pays) liste.SelectedItem;



Les listes déroulantes

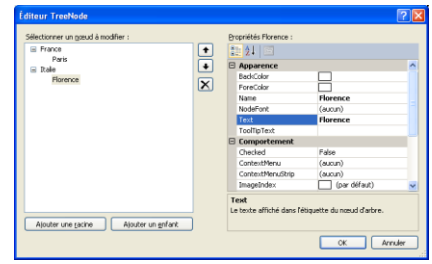
- Les événements usuels relatifs à une liste déroulante :
  - **TextChanged :** le contenu de la zone d'édition a été changé (sélection ou écriture dans la zone)
  - **SelectedIndexChanged** : sélection d'un nouvel article à l'aide de la souris et avec les touches de direction
  - **SelectionChangeCommitted** : changement de sélection, un nouvel article est dans la zone d'édition

Les listes en arbre

- Les listes en arbre (TreeView) sont utilisées
  - notamment pour visualiser les répertoires et les fichiers,
  - mais elles permettent de construire n'importe quelle hiérarchie d'articles
- Les nœuds de l'arbre sont de type TreeNode.
- Les nœuds peuvent :
  - Être reliés par des lignes (ShowLines)
  - Comporter une petite image (icône provenant d'une liste d'images)
  - Comporter des cases à cocher

Les listes en arbre

- Construction d'une liste en arbre sous l'environnement Visual (Editeur TreeNode)
  - En cliquant sur la propriété Nodes de la liste



Les listes en arbre

- La classe TreeView dérive de la classe Control et possède des propriétés comme
  - CheckedBoxes (T/F) indique si des cases à cocher sont affichées
  - Images : collections (ImageList) des images à appliquer à l'arbre
  - Nodes : collection des nœuds de l'arbre
  - SelectedNode : nœud sélectionné
  - Etc.

Les listes en arbre

- Les méthodes de développement ou réduction de l'arbre
  - void CollapseAll() réduction totale
  - void ExpandAll() développement total
- Les événements liés aux listes en arbre
  - AfterSelect
    - void arbre1_AfterSelect(object sender, TreeViewEventArgs e)
    - TreeViewEventArgs est dérivé de EventArgs
  - BeforeSelect
  - BeforeCollapse, AfterCollapse
  - BeforeExpand, AfterExpand

Les listes en arbre

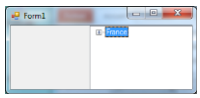
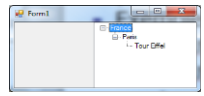
- L'événement TreeViewEventArgs contient 2 attributs
  - **Action** : qui indique le type d'opération qui vient d'être effectuée
    - ByKeyBoard ou ByMouse sélection à partir du clavier ou la souris
    - Collapse ou Expand : réduction ou développement d'un nœud
  - **Node** : qui désigne le nœud concerné par l'action

Les listes en arbre

- La liste TreeView comporte un attribut Nodes qui désigne la collection de ses nœuds
  - Nodes est de type TreeNodeCollection, collection d'objets de type TreeNode
- Cette collection peut être manipulée par les méthodes des collections.

Les listes en arbre

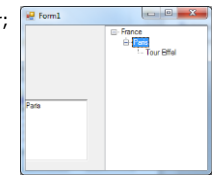
- Exemples
  - arbre.Nodes.Clear();
  - arbre.Nodes.Add("France"); // 1er niveau
  - arbre.Nodes[0].Nodes.Add("Paris"); // 2ème niveau
  - arbre.Nodes[0].Nodes[0].Nodes.Add("Tour Eiffel");

Les listes en arbre

- Il est possible de définir une action lors de la sélection d'un nœud en définissant le gestionnaire :

```
private void arbre_AfterSelect(object sender, TreeViewEventArgs e)
    {
       TreeView tv = (TreeView) sender;
       TreeNode nd = tv.SelectedNode;
       // nd=e.Node
       Edition.Text = nd.Text;
    }
```

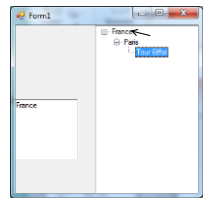


  - sender : objet qui a déclenché l'action
    - Ici le TreeView
  - e : information sur l'évènement
    - Ici le nœud sélectionné

Les listes en arbre

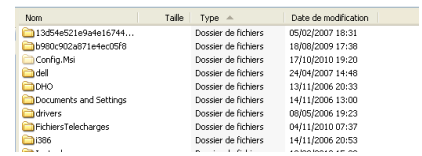
- Pour déterminer quel nœud est survolé par la souris (et faire une action, …)
- On peut écrire dans le gestionnaire MouseMove de la liste (avec un argument MouseEventArgs e)

```
Point p = new Point(e.X, e.Y);
TreeNode nd= arbre.GetNodeAt(p);
if (nd != null)
{ Edition.Text=nd.Text;
    }
```



Les fenêtres de liste

- Les fenêtres de liste (ListView) permettent de présenter des données dans une liste compartimentée en colonnes



Les fenêtres de liste

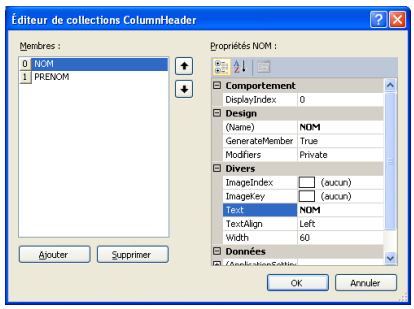
- Une fenêtre de liste a les caractéristiques suivantes :
  - Une petite image peut être affichée pour chaque article (1ère colonne)
  - En cliquant sur un bouton de titre de la colonne, on peut déclencher une action (tri ou autre)
  - Il est possible de modifier la largeur des colonnes avec la souris
  - Le clic sur un article déjà sélectionné permet de modifier son libellé
  - L'utilisateur peut réorganiser les colonnes si cela est autorisé.

Les fenêtres de liste

- Les listes ListView permettent de visualiser les données sous 4 formes
  - Icônes standard (Grandes icônes)
  - Petites icônes
  - Liste limitée à 1 seule colonne
  - En mode Détails
- Spécification des colonnes
  - Sous Visual, à l'aide de l'éditeur de collections (ColumnHeader)
  - Ou de façon dynamique par l'application

Les fenêtres de liste

- Editeur des colonnes (propriété Columns de la liste)

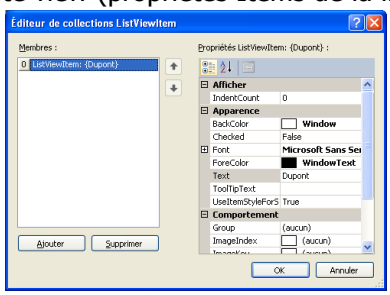


Les fenêtres de liste

- Création dynamique des colonnes
  - ColumnHeader col=new ColumnHeader();
  - col.Text="NOM";
  - Col.Width =100;
  - liste.Columns.Add(col);
  - Etc.
- Ajout des données
  - La liste doit être en mode détails
    - liste.View=View.Details;
  - ListViewItem listeItem= new ListViewItem (new string [ ] {"Dupont", "Pierre"};
  - liste.Items.Add(listeItem);

Les fenêtres de liste

- Editeur de création de données dans une liste view (propriétés Items de la liste)



Les fenêtres de liste

- Un ou plusieurs articles peuvent être sélectionnés selon la propriété MultiSelect.
  - Nombre d'articles sélectionnés
    - liste.SelectedItems.Count
  - Libellé de l'article (principal)
    - liste.SelectedItems[0].Text // 1er article, 1ère colonne
  - Nombre de sous articles (nombre de colonnes)
    - liste.SelectedItems[0].SubItems.Count
  - Libellé du 1er sous-article (2ème colonne)
    - liste.SelectedItems[0].SubItems[0].Text

Les fenêtres de liste

- Pour pré-sélectionner l'article de la ième ligne, il faut
  - Donner le focus à la liste
    - liste.Focus();
  - Remettre à vide la liste des indices sélectionnés
    - liste.SelectedIndices.Clear();
  - Mettre l'indice i dans la sélection
    - liste.SelectedIndices.Add(i);

Les fenêtres de liste

- La classe ListView comporte un grand nombre de propriétés
  - **Activation** : donne la façon de sélectionner un article (Standard, OneClick, TwoClick)
  - **CheckedBoxes** : (T/F) indique s'il y a des cases à cocher
  - **Columns, Items,**
  - **SelectedItems, CheckedItems, CheckedIndices** pour les articles sélectionnés
  - **LargeImagesList** de type ImageList, liste des images pour les grandes icônes
  - **SmallImagesList** de type ImageList, liste des images pour les petite icônes
  - **StateImageList** de type ImageList, liste des images pour le mode détails
  - Etc.

Les fenêtres de liste

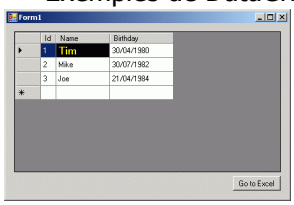
- L'ajout de nombreuses lignes peut provoquer un effet visuel de scintillement (si tri par exemple)
  - On peut entourer les mises à jour avec
    - liste.BeginUpdate();
    - liste.EndUpdate();
- Chaque ligne de la liste est de type ListViewItem, dérivé de Object
  - Il est donc possible d'utiliser les propriétés et les méthodes de cette classe
    - Text : le libellé de l'item
    - SubItems : collections des sous-items de l'item
      - i.e. les autres valeurs de la ligne

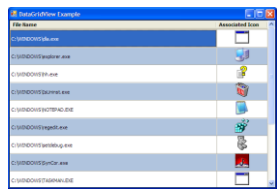
La représentation en grille

- Une grille (DataGridView) permet d'implémenter une grille (façon excel)
  - dont les données peuvent éventuellement être issues d'une base de données
  - Ou bien des informations de l'application
- La classe DataGridView comporte un nombre important de propriétés et de méthodes.

La représentation en grille

- Exemples de DataGridView





La représentation en grille

- On s'intéresse à la façon de :
  - Remplir la grille (avec des données d'un BD, d'une collection ou d'une table de données (DataTable)
  - Influer sur la présentation de la grille
    - Polices, couleurs d'affichage, etc.
  - Modifier les entêtes des colonnes et redimensionner colonnes et rangées (lignes)
  - Modifier l'apparence des cellules
  - Connaître ou modifier le contenu des cellules
  - Modifier le style d'une cellule particlière
  - Dessiner une cellule par exemple si elle contient une image)

La représentation en grille

- Il existe différentes représentations d'une cellule
  - Une cellule peut contenir du texte mais également d'autres choses comme
    - Une case à cocher, une liste déroulante, une image ou n'importe quelle représentation
  - Une cellule est de type DataGridViewCell mais cette classe est une classe abstraite avec des sous-classes
    - DataGridViewTextBoxCell, DataGridViewButtonCell,
    - DataGridViewLinkCell, DataGridViewCheckedBoxCell
    - DataGridViewComboBoxCell, DataGridViewImageCell

La représentation en grille

- Remplissage d'une grille à partir d'une table de données (DataTable)
  - Définition des colonnes de la DataTable
    - DataTable dt= new DataTable();
    - // 1ère colonne NOM
    - DataColumn dc = new DataColumn("NOM", typeof(string));
    - dt.Add(dc); // ajout de la colonne
    - Etc.
  - Ajout d'une ligne de données
    - dt.Rows.Add(new object[ ] {"Dupont", "Pierre"});
    - Rows est la collection des lignes (sans l'entête)

La représentation en grille

- Remplissage d'une grille à partir d'une table de données (DataTable)
  - Association des données de la table (DataTable) avec les données (propriété DataSource) de la grille (grille de type DataGridView)
    - grille.DataSource = dt; // dt est la DataTable
- La grille est alors affichée.
  - Les données de type bool sont affichées par des cases à cocher
  - Les données de type Image ont l'image affichée
  - Une nouvelle ligne peut être ajoutée par l'utilisateur
  - Toutes les cellules peuvent être modifiées (par défaut, clic sur la cellule et frappe d'une touche pour passer en mode modification)

La représentation en grille

- Remplissage d'une grille à partir d'une collection ou d'un tableau, par exemple avec une liste de personnes
- Classe personne : on définit des primitives pour les colonnes en public
  - class Personne
    - { private string nom;
    - private string prenom;
    - public string **Nom** { get {return nom;}
    - public string **Prenom** {get {return prenom,}
    - …
  - Les primitives "ajoutées" ne correspondent pas obligatoirement exactement aux attributs internes private.

La représentation en grille

- Remplissage d'une grille à partir d'une collection ou d'un tableau, par exemple une liste générique de personne
  - List<Personne> liste = new Liste<Personne>
  - liste.Add (new Personne("Dupont", "Pierre");
  - grille.DataSource = liste;
- Les colonnes proviennent des propriétés/primitives publiques de la classe
  - Nom et Prenom
- La grille est en lecture seule (pas d'ajout possible)

La représentation en grille

- Présentation de la grille
  - Police, couleur d'affichage, etc.
- Les propriétés
  - BackGroundColor change la couleur de fond de l'espace
  - GridColor influe sur la couleur des lignes de séparation des cellules

La représentation en grille

- Modification des entêtes des colonnes
  - Les entêtes des colonnes ne sont affichées que si la propriété ColumnHeaderVisible vaut vrai.
  - Les colonnes ne peuvent être réorganisées que si la propriété AllowUserToOrderColumns vaut vrai
  - grille.Columns["Nom"].Text="Name";
  - grille.Columns["Nom"].HeaderCell.Style.BackColor = Color.Yellow;
  - grille.Columns["Nom"].HeaderCell.Style.ForeColor = Color.Red;
  - grille.Columns["Nom"].HeaderCell.Style.Font = new Font (…);

La représentation en grille

- Redimensionnement des colonnes et rangées
  - La partie données de la grille n'occupe pas forcément toute la surface du DataGridView
  - Il est possible de modifier la taille des colonnes
    - grille.Columns.["Nom"].Width= (grille.Width – grille.RowHeaderWidth)/2;
    - où RowHeaderWidth est la largeur de l'entête de la rangée
  - La propriété AutoSizeColumnMode permet de contrôler comment les largeurs des colonnes sont gérées par défaut
    - None, aucun calcul
    - ColumnHeader, largeur ajustée au libellé de l'entête
    - AllCell, largeur ajustée au plus grand libellé (y compris de l'entête)
    - Etc.

La représentation en grille

- Modification de l'apparence des cellules
- La propriété RowsDefaultCellStyle permet de modifier le styles d'une cellule
  - Alignement, BackColor, ForeColor, Format, SelectionBackColor, SelectionFore Color, Wrap
- La propriété AlternatingRowsDefaultCellStyle fait la même chose mais est appliquée une colonne sur 2

La représentation en grille

- Le contenu des cellules
  - grille.Rows.Count donne le nombre de rangées
  - string s= (string) grille.Rows[i].Cells["Nom"].Value;
  - string s= (string) grille.Rows["Nom", i].Value;
- Si la grille a été remplie par un DataTable, cela correspond aux données
  - string s= (string) dt.Rows[i]["Nom"];
- Pour modifier le contenu d'une rangée
  - grille.Rows[i].Cells["Nom"].Value="Durant";

La représentation en grille

- Modification du style d'une cellule
  - grille.Rows[i].Cells["Nom"].Style.ForeColor = Color.Red;
- Modification du style d'une rangée
  - grille.Rows[i].DefaultCellStyle.BackColor= Color.Green;
- Modification d'une cellule en fonction de son contenu, à l'initialisation et lors de l'introduction de nouvelles données par l'utilisateur,
  - Il est nécessaire de gérer l'événement CellFormatting de la grille

```
private void grille_CellFormatting(Object sender,
   DataGridViewCellFormattingEventArgs e)
{  e.Value désigne le contenu de la cellule,
   e.CellStyle désigne son style,
etc.
}
```


La représentation en grille

- Dessin dans une cellule
  - Il est possible de dessiner dans une cellule en traitant l'événement CellPainting de la grille

```
private void grille_CellPainting(object sender,
DataGridViewCellPaintingEventArgs e)
{ Graphics g = e.Graphics;
 g.FillRectangle (new Brush(Color.Red), e.CellBounds);
…}
```

  - e.CellBounds désigne le rectangle dans de la cellule
  - e.RowIndex vaut -1 pour les entêtes de colonnes
  - e.ColumnIndex vaut -1 pour les entêtes de rangées

Les composants - Conclusion

- La bibliothèque "windows Forms" propose un ensemble riche de listes
  - Allant des listes simples, de type liste déroulante
  - À des représentations complexes par les grilles.
- Les arbres et les grilles sont des composants complexes très utilisés.