Tp événementiel développement en C#

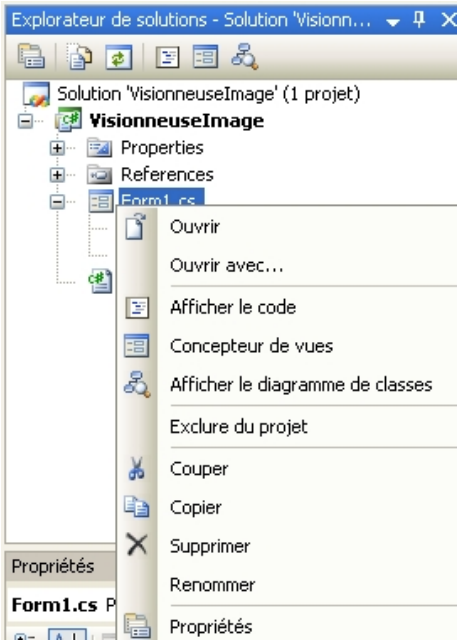
I) TP principes de base

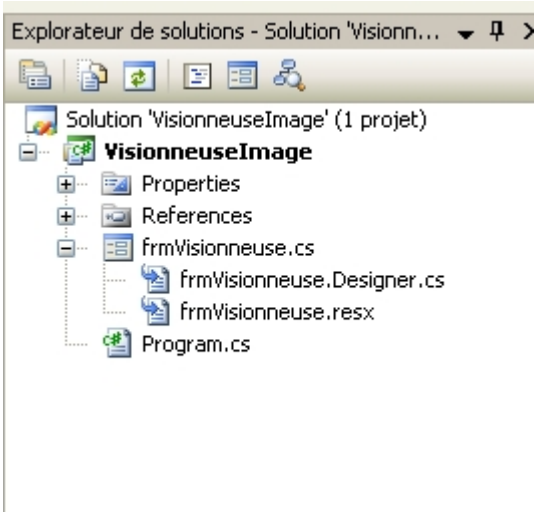
Création du projet

- Sous Visual Studio 2008 ou 2010 créer un nouveau projet et choisir application Windows.
  Donner un nom au projet et choisir l'emplacement de stockage des fichiers, valider par OK.

- Le formulaire étant sélectionné:
  Dans la fenêtre des propriétés (si elle n'est pas présente:menu affichage, propriétés), changer la propriété Name en frmVisionneuse.

- Dans la fenêtre Explorateur de solutions renommer (clic droit) form1.cs en frmVisionneuse.cs





Travail sur l'interface graphique

- Modifier la propriété Text: Visionneuse d'images.

- Enregistrer tout.

- Changer l'icône: propriété Icon développer (clic sur +) puis **…,** sélectionner un fichier .ico

- Changer la propriété Size (clic sur +): Width 400, Height 325

Ajout de contrôles visibles au formulaire.

A partir de la boîte à outils, controles communs:

- Ajouter un bouton (double clic).
- Modifier ses propriétes (voir tableau page suivante).

Propriété	*Valeur*
Name	**btnSelectionImage**
Location	**301; 10***
Size	**85; 23**
Text	**Sélection**

* 301 correspond à la coordonnée x et 10 à la coordonnée y

- Placer un nouveau bouton (par copier coller du précédent) et donner lui les propriétés suivantes:

Propriété	*Valeur*
Name	**btnQuitter**
Location	**301; 40**
Text	**Quitter**

- Placer un contrôle PictureBox et configurer ses propriétés comme suit:

Propriété	**Valeur**
Name	**imgAfficherImage**
BorderStyle	**FixedSingle**
Location	**8;8**
Size	**282;275**

Ajout de contrôles invisibles au formulaire.

- Ajouter par double clic le contrôle OpenFileDialog (catégorie boîtes de dialogue) au formulaire.
- Modifier ses propriétés de la façon suivante:

Propriété	**Valeur**
Name	**ofdSelectionImage**
FileName	**<ne rien mettre>**
Filter	**Bitmaps Windows|*.BMP|Fichiers JPEG|*.JPG|Tous les fichiers|*.***
Title	**Sélectionner une image**

remarque: Filter permet de choisir ce qui sera afficher dans la boîte de dialogue "ouvrir un fichier".

Écriture du code des gestionnaires d'événements.

Écriture du code qui est exécuté en réponse à l'événement clic sur le bouton Sélection.

- après double-clic sur le bouton, saisir le code ci-dessous:

```
private void btnSelectionImage_Click(object sender, EventArgs e)
{
    //Affiche la boîte de dialogue Ouvrir un fichier.
    if (ofdSelectionImage.ShowDialog() == DialogResult.OK)
    {
        //charge l'image dans la zone d'image.
        imgAfficherImage.Image = Image.FromFile(ofdSelectionImage.FileName);
        //Présente le nom du fichier dans le titre du formulaire.
        //modification par pg de la propriété texte du formulaire.
        this.Text = string.Concat("Visionneuse Image("+ofdSelectionImage.FileName+ ")");
    }

}
```

Remarque:

Sender fait référence au contrôle ou composant qui déclenche l'événement, il a un type générique, celui-ci ne nous intéresse pas pour l'instant.

e est un objet de type EventArgs, il possède des propriétés en relation avec l'événement Click (exemple: les coordonnées X, Y du pointeur de souris pour un gestionnaire MouseDown (voir variable msg du cours sur la structure d'un programme Windows et événements)

Écriture du code qui est exécuté en réponse à l'événement clic sur le bouton Quitter.

```
private void btnQuitter_Click(object sender, EventArgs e)
{
    //Ferme la fenêtre et quitte l'application
    this.Close();

}
```

Enregistrer (tout) et Tester le programme avec F5 ou 

Modification des propriétés d'un objet (le formulaire dans ce cas) par programmation.

- **Ajouter au formulaire deux nouveaux boutons avec les propriétés suivantes:**

Propriété	Valeur
Name	BtnAgrandir
Location	342;261
Size	21;23
Text	^ (Alt Gr 9)

Propriété	Valeur
Name	BtnReduire
Location	365;261
Size	21;23
Text	v

- **Ecrire le code associé à chaque gestionnaire**

```
private void BtnAgrandir_Click(object sender, EventArgs e)
{
    this.Width = this.Width + 20;
    this.Height = this.Height + 20;
}

private void BtnReduire_Click(object sender, EventArgs e)
{
    this.Width = this.Width - 20;
    this.Height = this.Height - 20;
}
```

Enregistrer (tout) et Tester le programme avec F5 ou

Dessin d'une bordure colorée autour de l'image

- **Ajouter au formulaire un nouveau bouton avec les propriétés suivantes:**

propriété	valeur
Name	btnDessinerBord
location	301;69
Size	85;23
Text	Dessiner

- **Code de l'événement click du bouton:**

```
private void btnDessinerBord_Click(object sender, EventArgs e)
  {
        Graphics objGraphics = null;
        objGraphics = this.CreateGraphics();
        objGraphics.Clear(SystemColors.Control);
        objGraphics.DrawRectangle(Pens.Blue, imgAfficherImage.Left - 1, imgAfficherImage.Top - 1,
           imgAfficherImage.Width + 1, imgAfficherImage.Height + 1);
        objGraphics.Dispose(); //Détruit l'objet objGraphics
  }
```

II) Les événements

Un événement peut être déclenché de trois façons différentes:

- **Par l'action de l'utilisateur, exemple un clic sur un bouton de l'interface.**
- **Par un objet , exemple un timer va déclencher un événement à intervalle régulier.**
- **Par le système d'exploitation, exemple Windows va déclencher un événement afin qu'une application redessine sa fenêtre.**
-

Création de gestionnaire d'événements

1) Evénements liés à la souris

MouseDown: se déclenche lorsque le pointeur de souris se trouve sur un contrôle (le contrôle a le focus) et que l'utilisateur appuie sur un bouton de la souris.

MouseUp:	**l'utilisateur relâche un bouton de la souris.**
MouseLeave:	**le curseur sort de la zone de survol.**
MouseEnter:	**le curseur entre dans la zone de survol**
Click:	**l'utilisateur clique (appui et relâchement d'un bouton de la souris).**
DoubleClick:	**l'utilisateur double clique.**
MouseMove: "	**le curseur se déplace sur un contrôle.**
MouseHover:	**Le curseur marque un court temps d'arrêt après entrée dans la zone de survol.**

- Ajouter 2 labels à l'application précédente.

Name	labelx
location	300; 110
Text	X:

Name	labely
location	300; 125
text	Y:

- Sélectionner la zone d'image, afficher ses événements, rechercher l'événement **MouseMove**, double cliquer pour créer le gestionnaire d'événement MouseMove.

- Compléter le gestionnaire de la façon suivante:

```
private void imgAfficherImage_MouseMove(object sender, MouseEventArgs e)
{
    labelx.Text = "X:" + e.X.ToString();
    labely.Text = "Y:" + e.Y.ToString();
}
```

La méthode ToString() permet de convertir l'int X en chaîne de caractères.

Sender fait référence au contrôle ou composant qui déclenche l'événement, il a un type générique, celui-ci ne nous intéresse pas pour l'instant.

e est un objet de type EventArgs, il possède des attributs en relation avec l'événement (voir tableau page suivante)

Propriétés de la classe `MouseEventArgs`

`MouseEventArgs` ← `EventArgs` ← `Object`

`Button`	`enum`	Indique quel bouton de la souris est enfoncé. `Button` peut prendre l'une des valeurs suivantes de l'énumération `MouseButtons` : `Left`, `Middle`, `None` et `Right`, mais aussi `XButton1` `XButton2` pour les deux boutons additionnels de la souris IntelliMouse Explorer.
`Delta`	`int`	Nombre de détentes sur la molette (n'a évidemment de signification que pour les souris équipées d'une molette).
`X`	`int`	Coordonnée `X`. Les axes sont relatifs à l'aire client et par rapport au coin supérieur gauche de cette aire client (axe des `X` horizontal et dirigé de gauche à droite tandis que l'axe des `Y` est vertical et dirigé de haut en bas).
`Y`	`int`	Coordonnée `Y`.

- Compiler et tester le projet.
- Réaliser une modification afin que les cordonnées de X et Y ne soient plus affichées lorsque le curseur de la souris quitte la zone image (événement MouseLeave).
- Réaliser une modification afin qu'au lancement du formulaire (événement load du formulaire) X, Y et les valeurs des coordonnées ne s'affichent pas.

Autres exemples:

- Pour tester si l'utilisateur a cliqué avec le bouton droit: if (e.Button == MouseButtons.Right)….