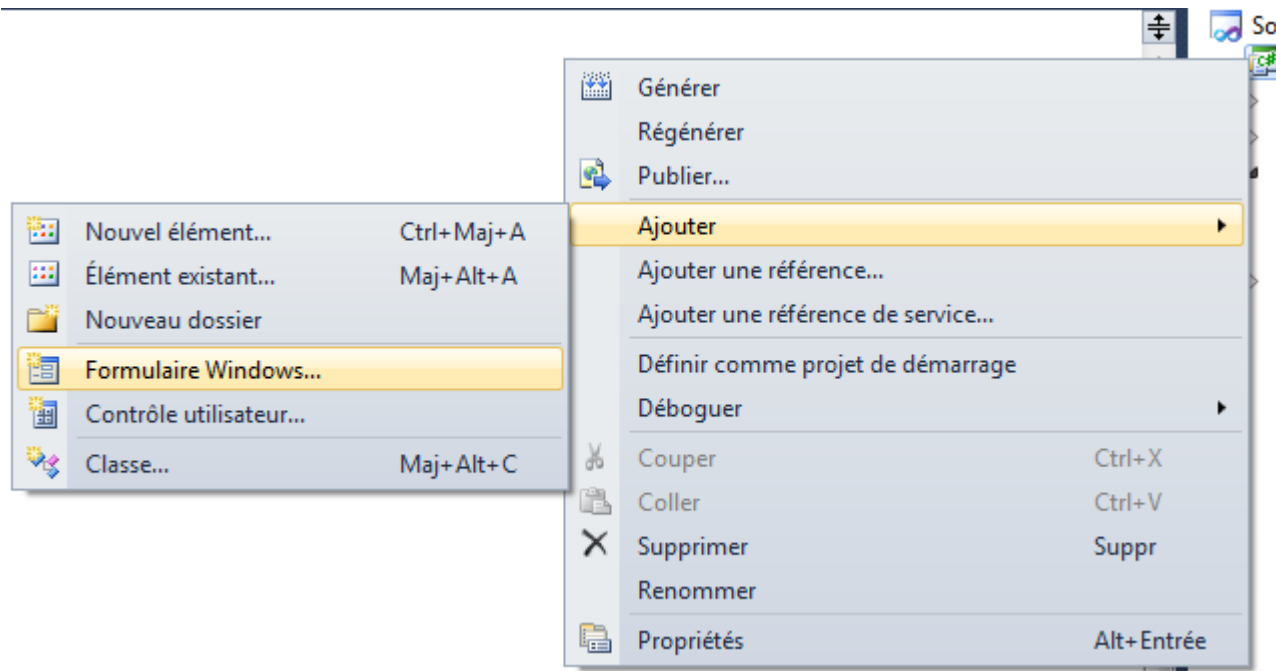


# Application multi-fenêtres

- Il s'agit d'une application mettant en place une certaine navigation entre plusieurs Forms.
- Pour ajouter un autre formulaire au projet :

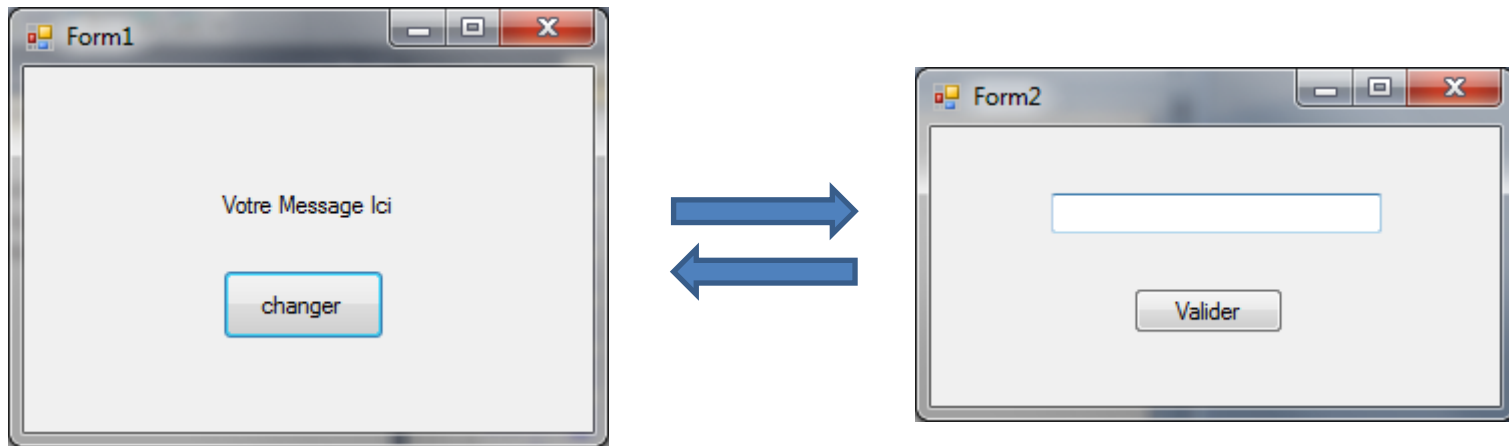


# Application multi-fenêtres

- La classe Form dispose de méthodes permettant un tel scénario :
  - Show : affiche le formulaire désiré (Visible = true)
  - Hide : cache le formulaire (Visible = false)
  - ShowDialog : affiche le formulaire désiré comme une boîte de dialogue. Elle retourne une instance de DialogResult.
  - Activate : active le formulaire et lui donne le focus.

# Application multi-fenêtres

Exemple :



On passera d'un form à l'autre pour modifier le message à afficher


# Application multi-fenêtres

- Exemple

```
public partial class Form2 : Form
{
    public Form2()
    {
        InitializeComponent();
    }


    private void button1_Click(object sender, EventArgs e)
    {
        Form1 f1 = new Form1();
        f1.setText(textBox1.Text);
        this.Hide();
        f1.Show();
    }
}
```

*Mauvaise Implémentation*



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Form2 f2 = new Form2();
        this.Hide();
        f2.Show();
    }
}
```

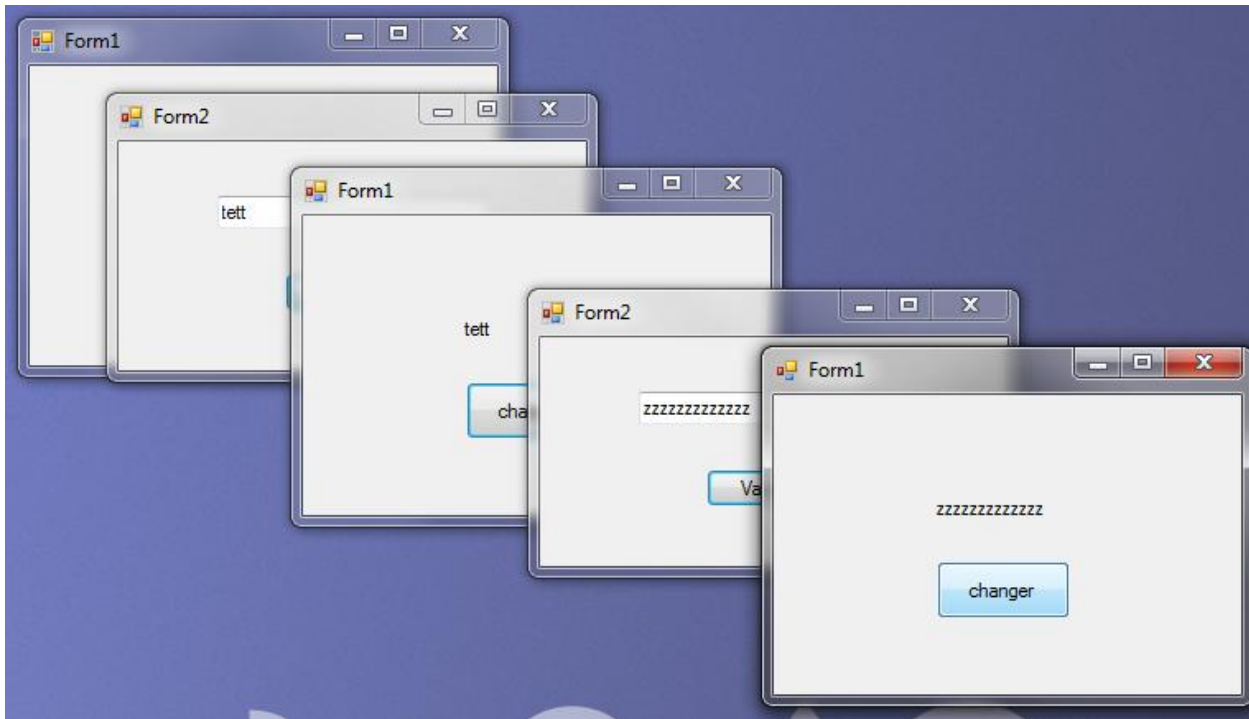


```
public void setText(string s)
{
    label1.Text = s;
}
```

# Application multi-fenêtres

# Mauvaise implémentation :

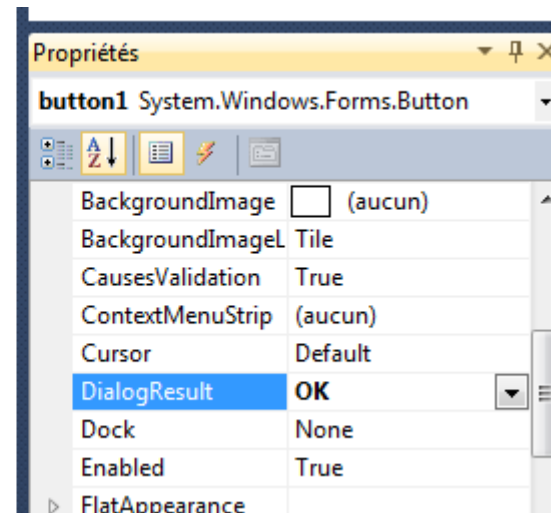
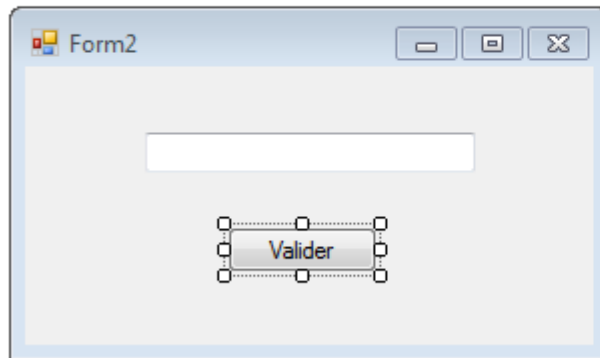
- Plusieurs instantiations inutiles



# Application multi-fenêtres

## Solution : ShowDialog

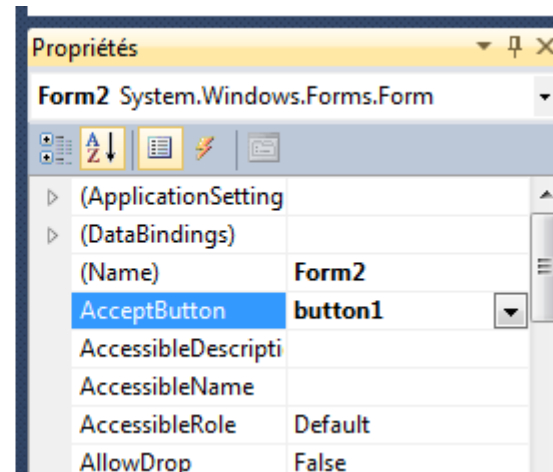
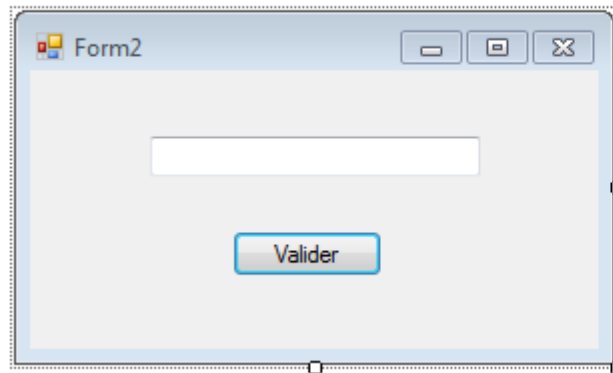
- Préparer Form2 en utilisant AcceptButton
  - associer à la propriété DialogResult du bouton **Valider** la valeur OK



# Application multi-fenêtres

Solution : ShowDialog

- Préparer Form2 en utilisant AcceptButton
  - associer à la propriété AcceptButton du formulaire **Form2** le bouton **Valider** (Button1)



# Application multi-fenêtres

Solution : ShowDialog

- Préparer Form2 en utilisant AcceptButton
  - Ajouter à **Form2** la propriété InputText

```
public string InputText
{
    get { return textBox1.Text; }
    set { textBox1.Text = value; }
}
```



# Application multi-fenêtres

## Solution : ShowDialog

- Au niveau de Form1:

```
public partial class Form1 : Form
{
    Form2 f2;
    public Form1()
    {
        InitializeComponent();
        f2 = new Form2();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        f2.InputText = label1.Text;
        this.Hide();
        if (f2.ShowDialog() == DialogResult.OK)
        {
            label1.Text = f2.InputText;
            this.Show();
        }
    }
}
```

**Testez !**

# MDI : Multiple Document Interface

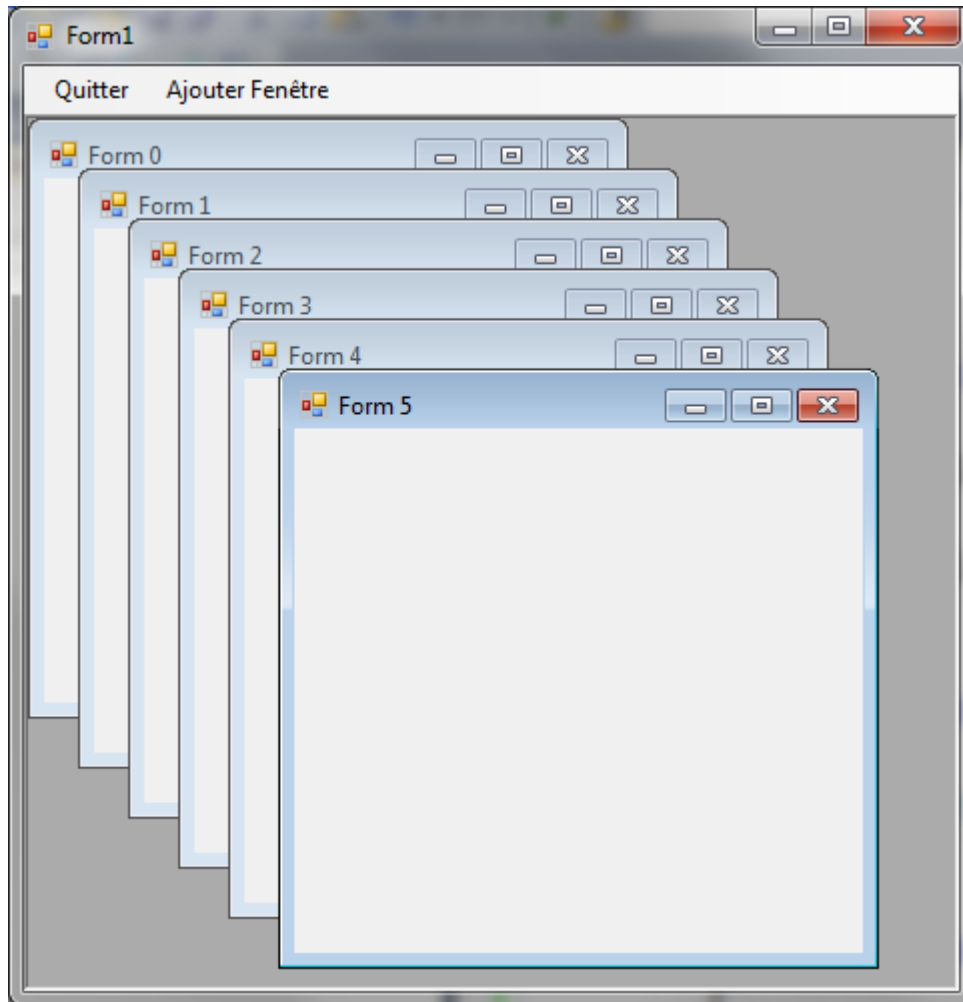
- Il s'agit d'une autre façon de gérer plusieurs fenêtres
- Deux possibilité :
  - Transformer un simple formulaire en MDI
  - Créer un MDI Standard selon le modèle offert par Visual C#
- Attention : une fenêtre au sein d'un MDI ne peut être de type MDI (pas d'imbrication)

# MDI : Multiple Document Interface

Transformer un simple formulaire en MDI

- mettre sa propriété **IsMdiContainer** à true
- créer d'autres formulaires en spécifiant quel est leur formulaire parent avec la propriété **MdiParent**
- Exemple : réaliser une interface qui peut contenir des forms créés par elle-même.

# MDI : Multiple Document Interface



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void quitterToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void ajouterFenêtreToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Form form = new Form();
        form.Text = "Form " + MdiChildren.Length;
        form.MdiParent = this;
        form.Show();
    }
}
```

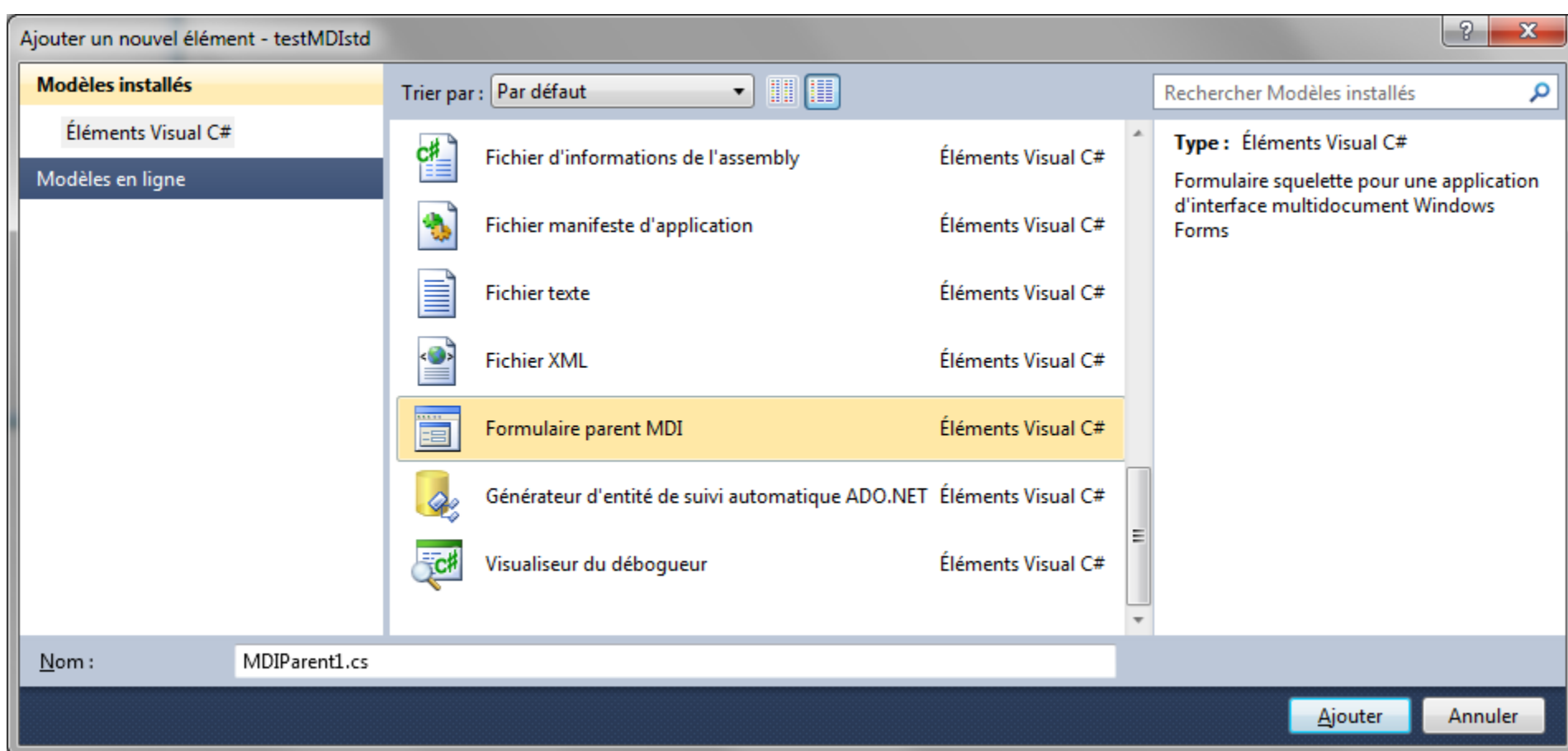
# MDI : Multiple Document Interface

Propriétés en rapport avec les MDI :

- `ActiveMdiChild` : obtient la fenêtre enfant MDI active.
- `IsMdiChild` : obtient une valeur indiquant si le formulaire est un formulaire enfant MDI.
- `IsMdiContainer` : obtient ou définit une valeur indiquant si le formulaire est un conteneur de formulaires enfants d'interface multidocument (MDI).
- `MdiChildren` : obtient un tableau de formulaires représentant les formulaires enfants MDI qui sont apparentés à ce formulaire.
- `MdiParent` : obtient ou définit le formulaire parent MDI en cours de ce formulaire.
- `ActivateMdiChild` : active l'enfant MDI d'un formulaire.

# MDI : Multiple Document Interface

## Créer un MDI Standard



# MDI : Multiple Document Interface

