Programmation VBA, développement rapide d'interfaces

Hervé Hocquard

Sommaire

- Introduction
- VBA et Excel
  - L'éditeur (VBE)
  - Modèle objet de VBA
  - Procédures Sub
  - Procédure Function
  - Enregistrer des Macros
- Programmation en VBA
  - Variables, instructions
  - Objets Range
  - VBA et fonctions
  - Déroulement d'un programme

Introduction (1)

- VBA: Visual Basic pour Application
  - Langage Visual Basic fortement associé à la suite bureautique MS Office : Word, Powerpoint… Excel.

- VBA et Excel:
  - Automatiser certaines tâches
  - Exécuter des actions en série (traitement par lot ou *batch processing*)
  - Commandes et boutons personnalisées
  - Ajouter des boutons dans le ruban
  - Créer des interfaces graphiques et des applications avec Excel

Avantages / inconvénients du VBA

Avantages

- Automatisation d'une exécution
  - Rapidité
  - Régularité
  - Sans erreurs
- Apprentissage « facile » permettant d'étendre les fonctionnalités d'Excel
- Fortement lié à Office

Inconvénients

- Nécessite Excel
- Pérennité du code?
- Limité: difficile de produire de « grosses » applications (mais ce n'est pas ce qu'on lui demande)

But du module

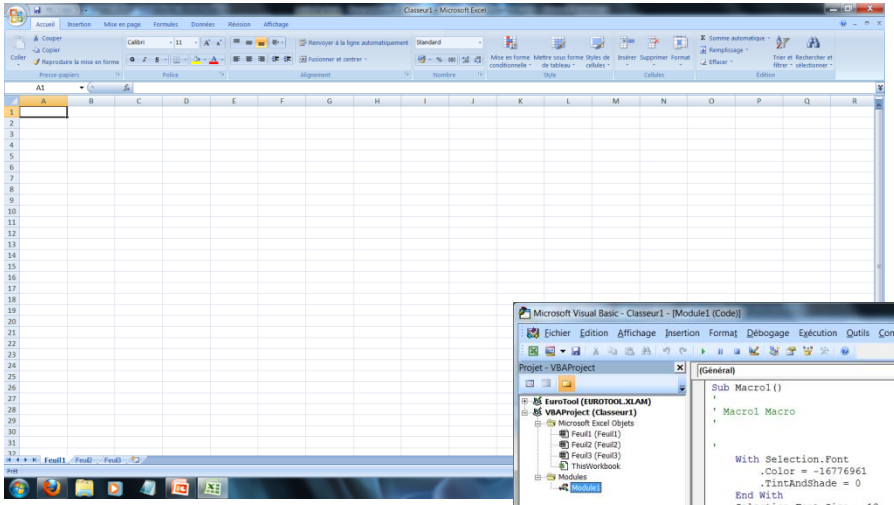
- « Augmenter » les capacités des applications bureautiques (Excel) en proposant vos propres programmes, adaptés aux besoins

- Développer rapidement des interfaces sur un éditeur dédié afin de faciliter la prise en main de vos programmes par d'autres dans l'entreprise

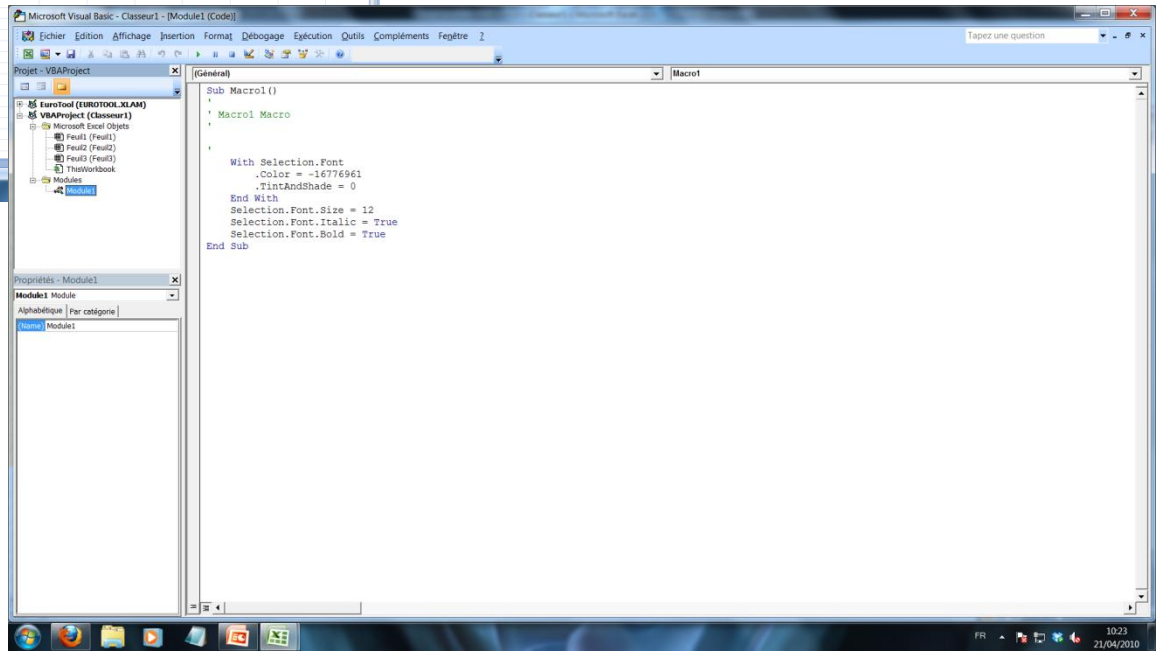
Partie 1 : VBA et Excel

- L'éditeur (VBE)
- Modèle objet de VBA
- Procédures Sub et Function

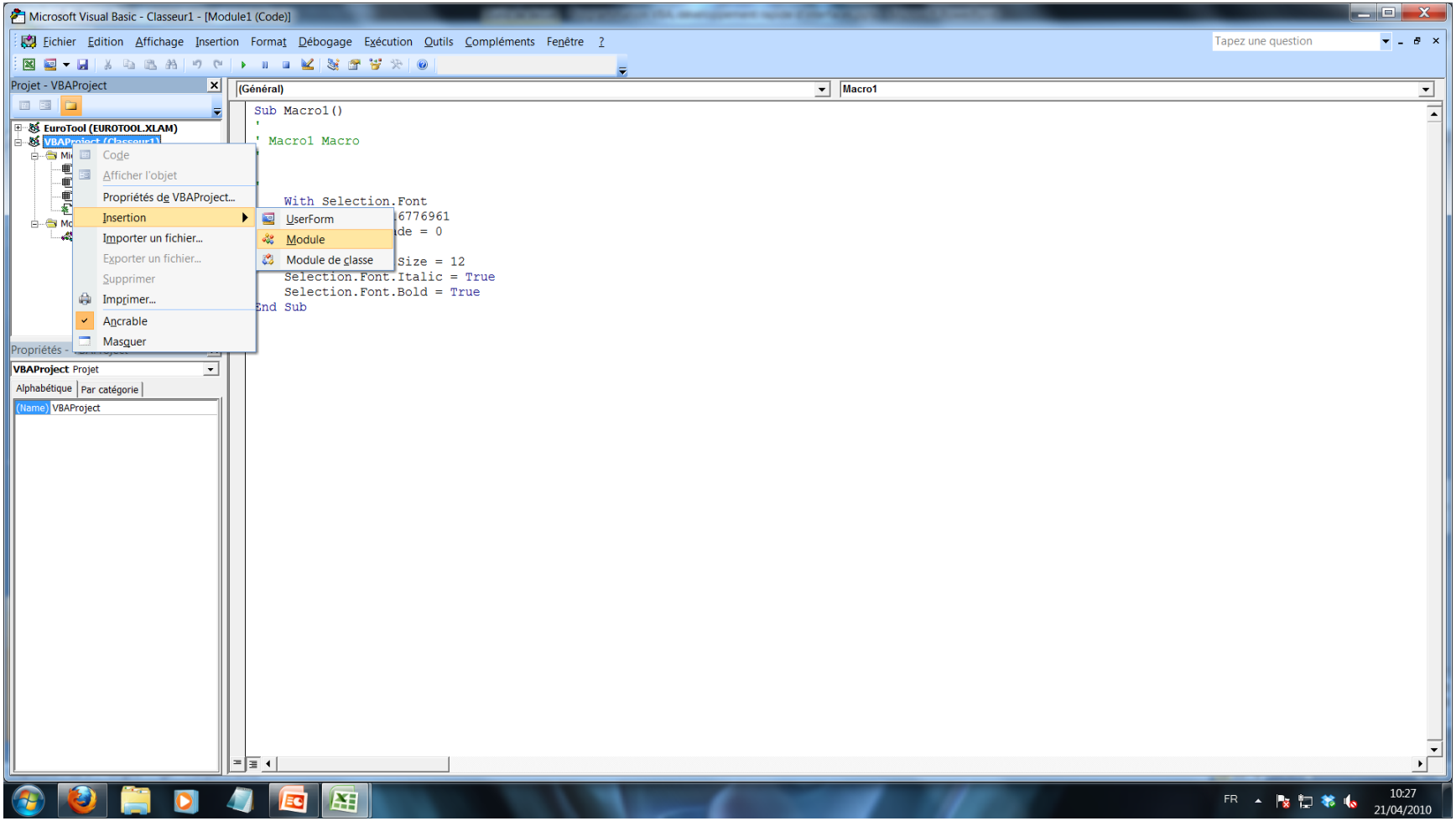
L'éditeur (Visual Basic Editor)



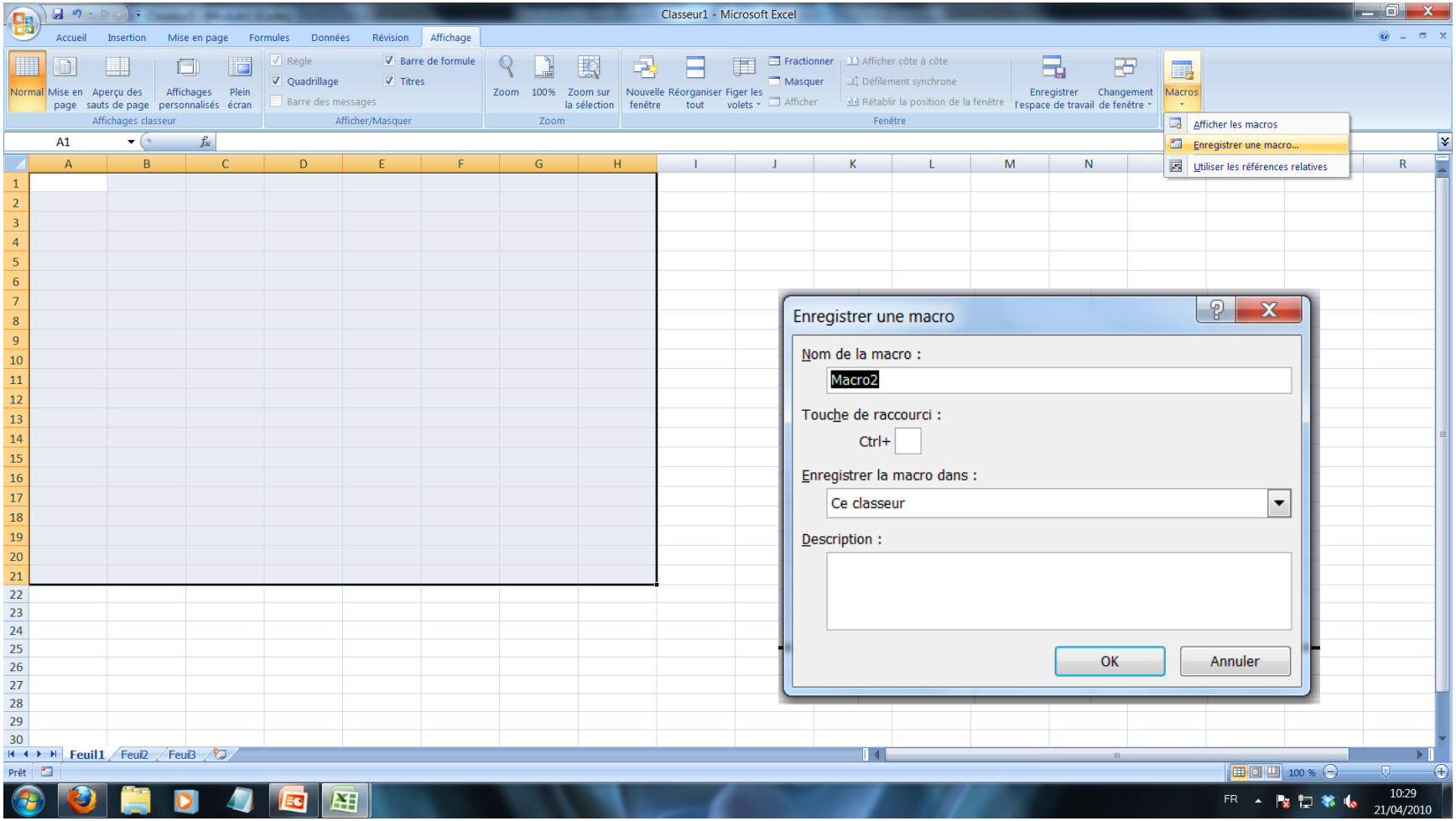
Alt + F11



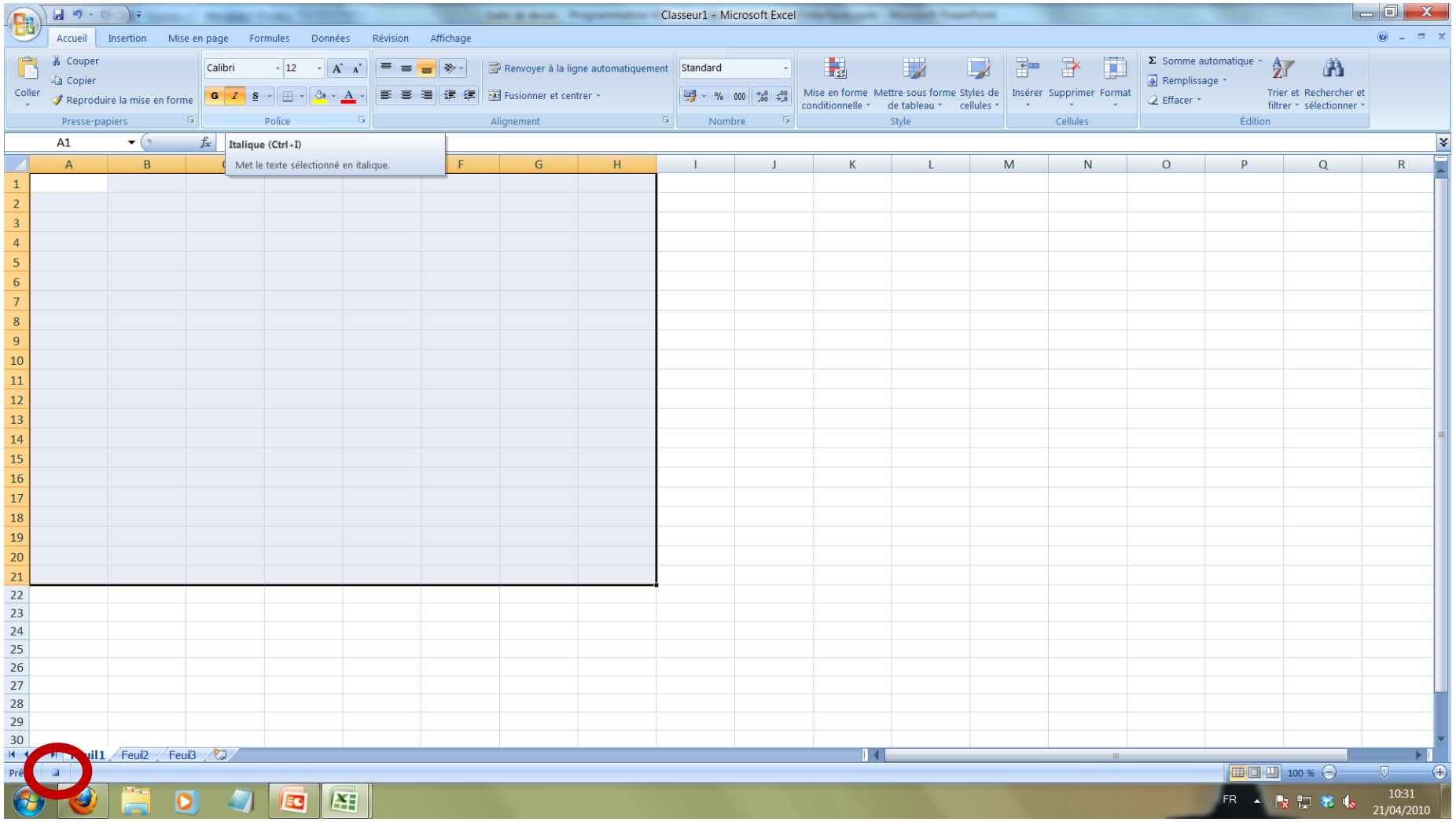
Insertion d'un module dans l'éditeur



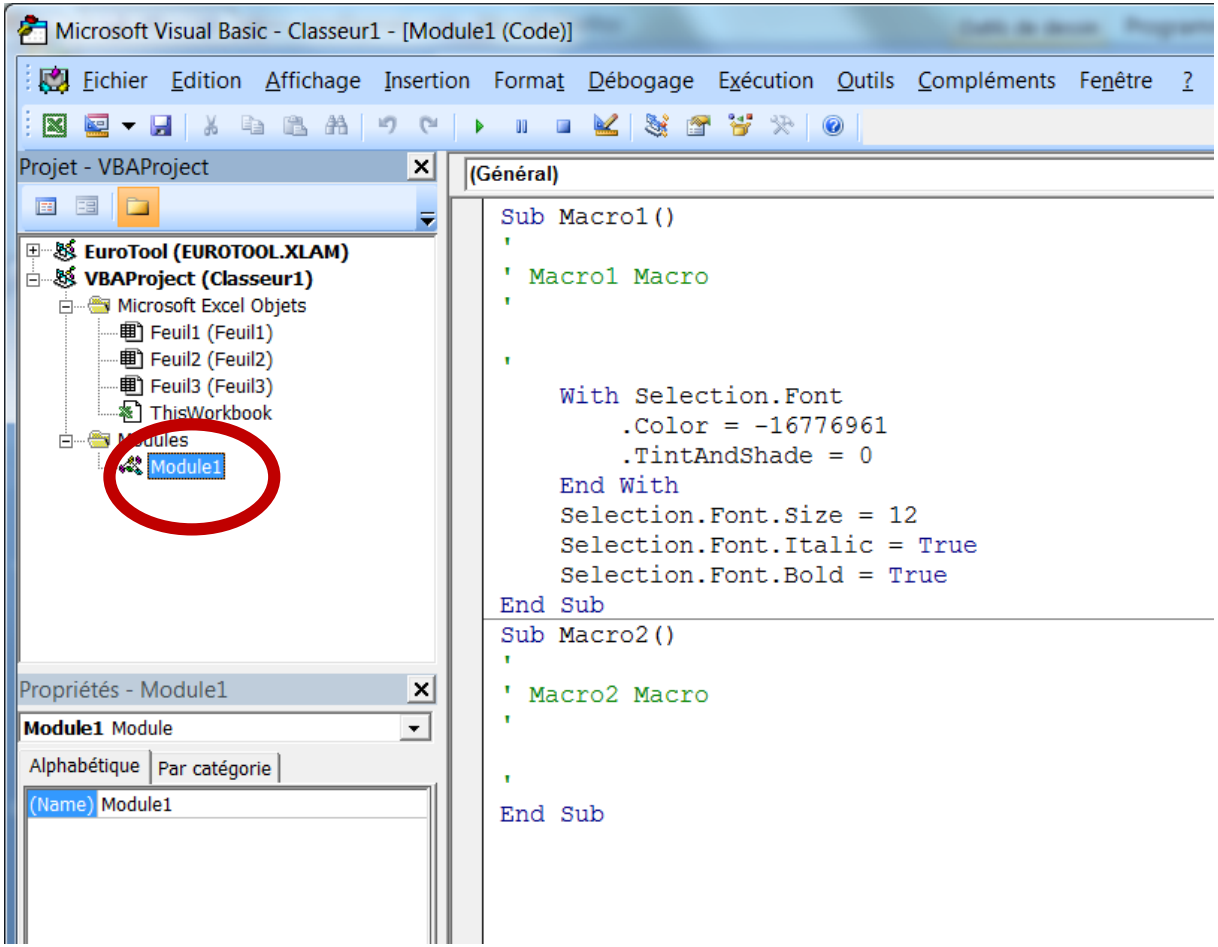
Enregistrer une macro



Enregistrer une macro

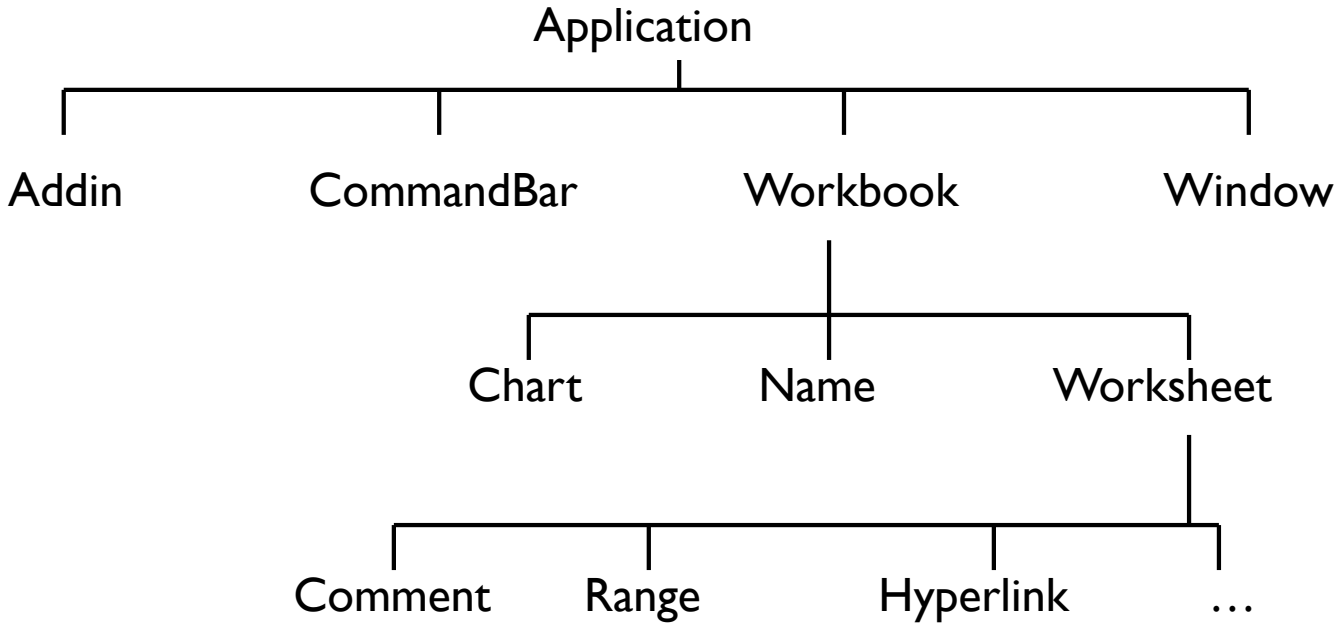


Impact dans l'éditeur



Le modèle objet dans VBA

- Un objet est constitué d'attributs (ou propriétés) et de méthodes qui lui sont associées
- Les objets existants sont constitués en hiérarchie (relation de composition)



Les collections

- Concept clé
- On rajoute un « s »!
  - Workbooks : collection des objets Workbook
  - Worksheets : collection des objets Worksheet
  - … etc.

- Faire appel à un élément d'une collection: 2 méthodes:
  - Appel par le nom de l'élément
    - Ex: `Worksheets("Feuil1")`
  - Appel par l'indice
    - Ex: `Worksheets(1)`

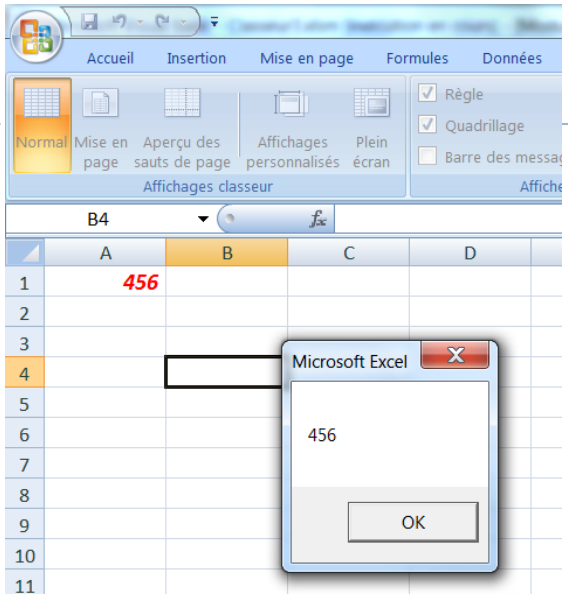
Hiérarchie: Accéder aux objets

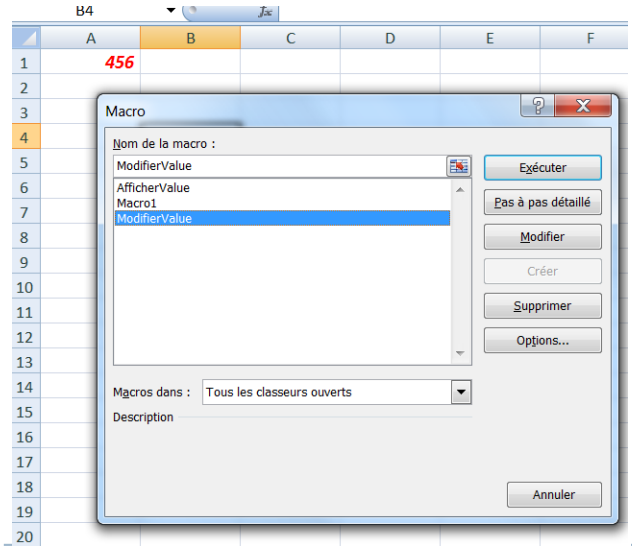
- Opérateur point (.)
  - Ex:
    ```
    Application.Workbooks("Classeur1.xlsx").Worksheets(1).Range("A1").Value=934
    ```
- Simplification: par exemple si Classeur1.xlsx est le classeur actif:
  - `Worksheets(1).Range("A1").Value=934`

Propriétés d'un objet

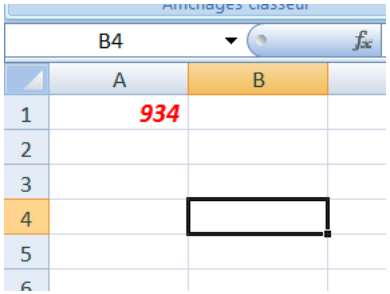
```
Sub AfficherValue()
'
Contents = Worksheets("Feuil1").Range("A1").Value
MsgBox Contents


    ActiveWorkbook.Save
End Sub
```






```
Sub ModifierValue()
    Worksheets("Feuil1").Range("A1").Value = 934
End Sub
```



Méthode d'un objet

- Action relative à un objet

- Exemples:
  - Worksheets("Feuil1").Activate
  - Range("A1").Copy Range("B1")

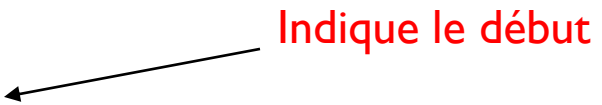
- Une méthode prend en compte 0, 1 ou plusieurs arguments.
  - Le premier argument est séparé de la méthode par un espace, les arguments sont séparés entre eux par des virgules
  - OU utilisation des parenthèses

Procédures

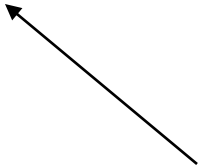
- 2 Types: Sub et Function
  - Une procédure Sub est un groupe d'instructions VBA qui exécute une ou plusieurs actions avec Excel.
  - Une procédure Function est un groupe d'instruction VBA qui exécute un calcul et retourne une seule valeur.
- L'enregistreur de macros produit toujours une procédure Sub.
- Possibilité de lancer une procédure Sub via des raccourcis clavier / des boutons personnalisés…
- Mais une procédure Function n'est appelée que dans une cellule ou dans une autre procédure.

Syntaxe de base Sub

- Il faut toujours indiquer où se trouve le début et la fin du programme que l'on écrit.




```
Public Sub nom_du_programme(on peut mettre des arguments ou pas)
        ' séquences d' instructions
End Sub
```



Syntaxe de base Function

- Une fonction encapsule aussi un ensemble d'instructions, mais retourne une valeur (désignée par le nom même de la fonction).
  - Cette valeur doit être affectée au nom de la fonction avant la fin du bloc d'instructions.
  - Syntaxe :

```
Function nom(arg1 As type, …) As Type
    Instructions
    nom = exp_du_bon_type
    Instructions
End Function
```

- Il faut préciser le type de la valeur retournée.

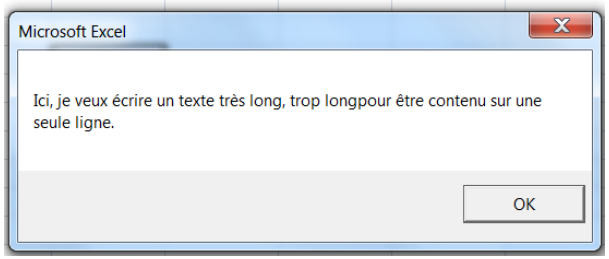
Partie 2 : Programmation en VBA

- Variables, instructions
- Objets Range
- VBA et fonctions
- Déroulement d'un programme
- Evènements automatiques
- Gérer les erreurs

Premiers pas

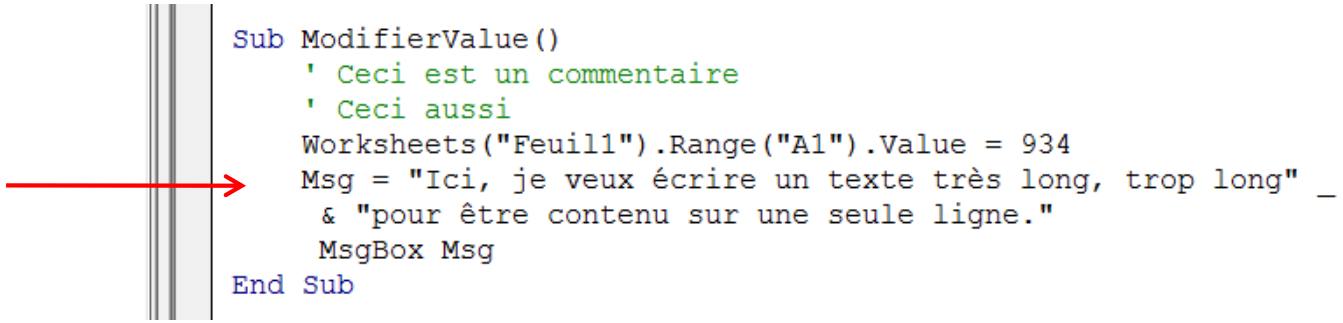
- Pas de point virgule à la fin d'une instruction
- Une instruction par ligne
  - Espace + underscore (" _") pour écrire une instruction sur plusieurs lignes
- Commentaires: commencer la ligne par une apostrophe

```
Sub ModifierValue()
    ' Ceci est un commentaire
    ' Ceci aussi
    Worksheets("Feuil1").Range("A1").Value = 934
    Msg = "Ici, je veux écrire un texte très long, trop long" _
     & "pour être contenu sur une seule ligne."
     MsgBox Msg
End Sub
```



Les Variables

- Nommage:
  - Doit commencer par une lettre, puis lettres, chiffres et quelques caractères spéciaux
  - Pas de différence de casse
  - Caractères invalides: #, $, %, &, !
  - Maximum 254 caractères
- Typage des données
  - Données peuvent être non typées explicitement: type Variant



```
Sub ModifierValue()
    ' Ceci est un commentaire
    ' Ceci aussi
    Worksheets("Feuil1").Range("A1").Value = 934
    Msg = "Ici, je veux écrire un texte très long, trop long" _
     & "pour être contenu sur une seule ligne."
     MsgBox Msg
End Sub
```

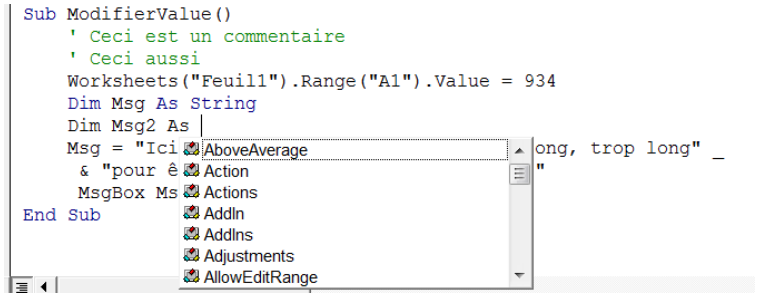
Typage des données

Type	Valeurs
Boolean	Vrai, faux
Integer	Entiers
Long	Entiers
Single	Réels
Double	Réels
Currency	4 chiffres après la ,
Date	1/1/100 à 31/12/9999
String	Chaines de caractères
Object	Tout objet
Variant	N'importe quel type

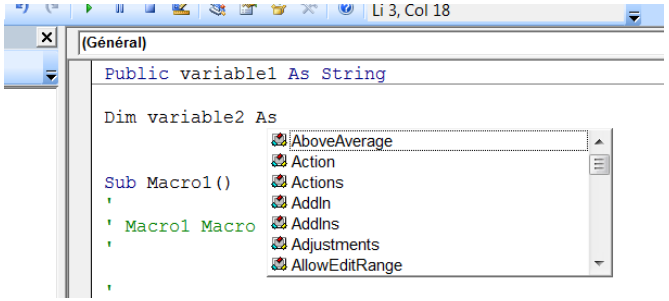
Déclarer des variables, portée

- Forcer la déclaration: ajouter en début de module
  - `Option Explicit`

- Portée: procédure courante:
  - `Dim` ou `Static` dans la procédure
  - `Dim nomVariable As type`



- Portée: module:
  - Dim hors d'une procédure



- Portée : toutes les procédures, tous les modules:
  - Public au tout début du module

Variables particulières

- Les variables Statiques
  - Ne sont pas réinitialisées à la sortie de la procédure (ex, pour des compteurs)
  - Static Compteur As Integer

- Les Constantes
  - la valeur est donnée et ne peut changer
  - Const Pi As Double = 3.1415

- Les dates
  - Doivent être mises entre dièses sous la forme:

```
Dim hoy As Date
hoy = #1/1/200#
today = #1/2/450#
```

Instructions

- Affectation : =
- Opérateurs: +, *, /, -, ^, &, \, Mod
- Opérateur logique: Not, And, Or, Xor, Eqv, Imp

Les tableaux (1)

- Déclaration
  - `Dim MonTableau(1 to 100) As Integer`
  - Index débute à 0 par défaut;
    - `Option Base 1`
- Tableaux multidimensionnels
  - `Dim MonTableau(1 to 10, 1 to 10) As Integer`
- Affectation
  - MonTableau(3,4) = 125

Les tableaux (2)

- Tableaux dynamiques
- Création
  - `Dim MonTableau() As Integer`
- Redimensionnement
  - `ReDim MonTableau(NombreElement)`
- Redimensionner en gardant les données déjà présentes
  - `ReDim Preserve MonTableau(NombreElements)`

Les tableaux avec Array (3)

- Structure pour afficher le contenu:

```
Dim mois As Variant
Dim m As Variant
mois = Array("Janvier", "Mars", "Août", "Décembre")
For Each m In mois
        MsgBox m
Next m
```

- Ou alors…

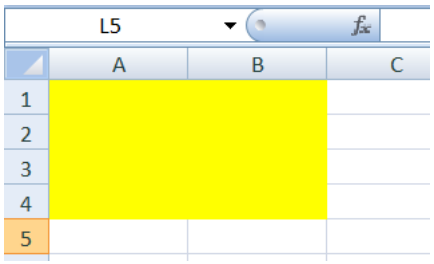
```
Dim mois As Variant
Dim  i As Integer
mois = Array("Janvier", "Mars", "Août", "Décembre")
For i = 0 To 3
        MsgBox mois(i)
Next i
```

L'objet Range (1)

- Plage de cellule
  - `Range("A1:C5")`, `Range("Liste_Prix")` (plage nommée) `Range("3:3")` (ligne entière), `Range("D:D")` (colonne entière).
- Propriétés:

Cells	**Address**
Offset	**hasFormula**
Value	**Font**
Text	**Interior**
Count	**Formula**
Column, Row	**NumberFormat**

```
Sub Test()
    Range("A1:B4").Interior.Color = vbYellow
End Sub
```



L'objet Range (2)

- Méthodes
  - Select : Sélectionne une plage de cellule
    - ~~`Range("A1").Select`~~
      ~~`Selection.Value = "toto"`~~
    - `Range("A1").Value = "toto"`
  - Copy, Paste
    - ```
      Range("A1:A2").Select
      Selection.Copy
      Range("C3").Select
      ActiveSheet.Paste
      ```
  - Clear: efface le contenu et la mise en forme
  - Delete: supprime une plage (et décale les cellules)
      ```

L'objet Range (3)

- Exemples à tester et observer…
  - `Range("A1:H8").Formula = "=Rand()"`
  - `ActiveSheet.Cells(2, 1).Formula = "=Sum(B1:B5)"`
  - `ActiveSheet.Cells(2, 1).FormulaLocal = "=Somme(B1:B5)"`
  - `ActiveSheet.Cells(2, 1) = Application.WorksheetFunction.Sum(Range("B1:B5"))`
  - `Worksheets(1).Range("C5:C10").Cells(1, 1).Formula = "=Rand()"`

L'objet Range (4)

- Exemples à tester et observer…tout d'abord, à partir de A1 remplir un tableau de données…par exemple, A1=ALEA.ENTRE.BORNES(0;100)

```
MsgBox Range("D4").CurrentRegion.Address
MsgBox Range("D4").CurrentRegion.Columns.Count
MsgBox Range("D4").CurrentRegion.Rows.Count
MsgBox Range("D4").End(xlToLeft).Column
MsgBox Range("D4").End(xlToRight).Column
MsgBox Range("D4").End(xlUp).Row
MsgBox Range("D4").End(xlDown).Row
```

Instructions de contrôle

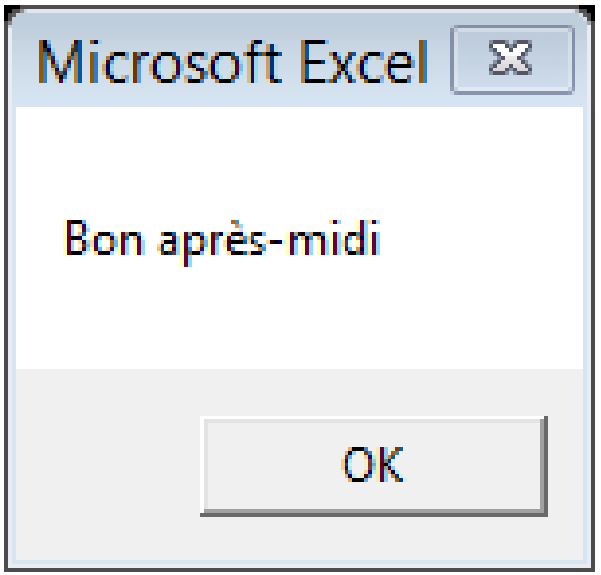
- If – Then : Exécute une action si la condition est vérifiée
- Select Case : Exécute une action parmi plusieurs, selon la valeur retournée
- For – Next : Exécute une série d'instructions en boucle autant de fois que spécifié
- For Each – Next: Parcourir une collection
- Do – While : Exécute une série d'instructions en boucle tant que la condition est vraie (True)
- Do – Until : Exécute une série d'instructions en boucle jusqu'à ce que la condition soit vraie

If / ElseIf / Then

```
Sub Bonjour()
Dim Msg As String
If Time < 0.5 Then
Msg = "jour"
    ElseIf Time < 0.75 Then
        Msg = "après-midi"
        Else
        Msg = "soir"
End If

MsgBox "Bon " & Msg

End Sub
```



Select Case

```
Sub AfficheRabais()
Dim Quantite As Integer
Dim Rabais As Double
Quantite = InputBox("Quelle quantité ?")
Select Case Quantite
    Case 0 To 24
        Rabais = 0.1
    Case 25 To 49
        Rabais = 0.15
    Case 50 To 74
        Rabais = 0.2
    Case Is >= 75
        Rabais = 0.25
    End Select
    MsgBox "Le Rabais est de " & Rabais & "%"
End Sub
```

For Next

```
For compteur = début To Fin [Step intervalle]
   [instructions]
   [Exit For]
   [instructions]
Next [compteur]
```

```
Sub RemplissageCellule()
    Dim Colonne As Integer
    Dim Ligne As Long
    For Colonne = 1 To 5 Step 1
        For Ligne = 1 To 12
            Cells(Ligne, Colonne) = Rnd
        Next Ligne
    Next Colonne
End Sub
```

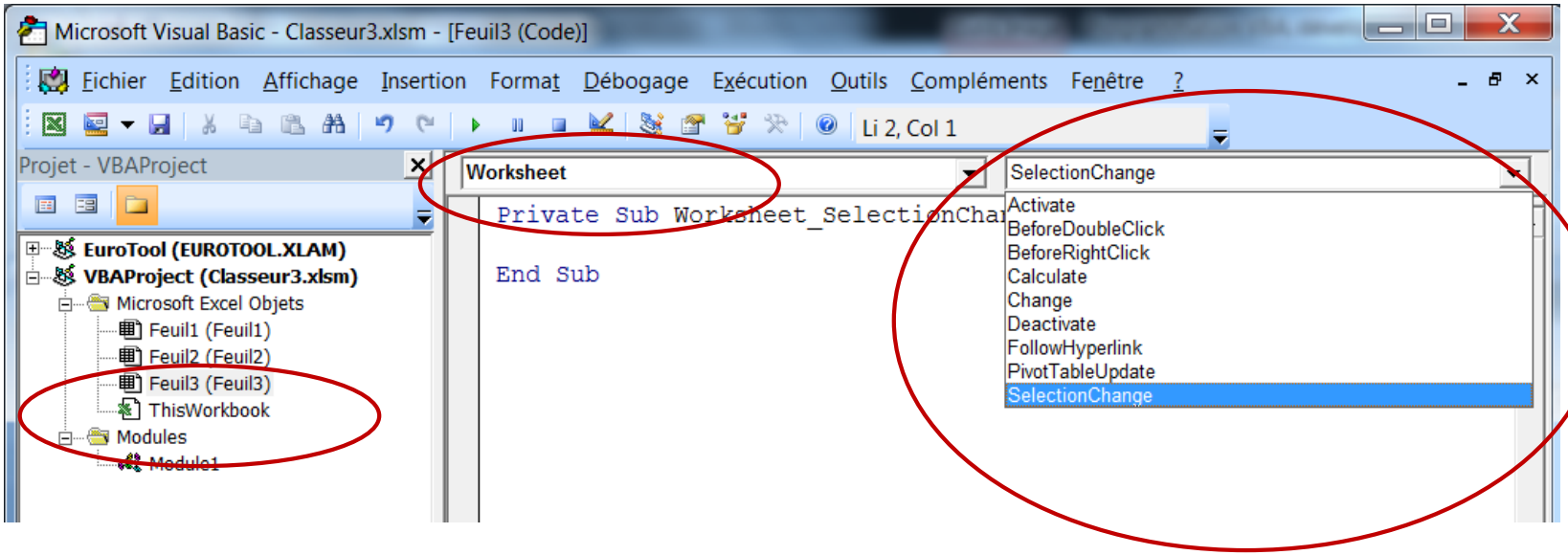
Do While, Do Until

```
Sub DemoDoWhile()
    Do While ActiveCell.Value <> Empty
        ActiveCell.Value = ActiveCell.Value * 2
        ActiveCell.Offset(1, 0).Select
    Loop
End Sub
```

```
Sub DemoDoUntil()
    Do Until IsEmpty(ActiveCell.Value)
        ActiveCell.Value = ActiveCell.Value * 2
        ActiveCell.Offset(1, 0).Select
    Loop
End Sub
```

Evènements

- Action déclenchant l'appel d'une méthode
- Primordial en interfaces graphiques!
- Certains objets disposent d'évènements



Partie 3 Les enregistrements

- Type structuré
- Champs simples ou structurés

Les enregistrements

- Contrairement aux tableaux, ce type structuré permet de regrouper des données de types différents.
- Exemple : on identifie un ouvrage par un code, un titre, un ou plusieurs auteurs, un éditeur et éventuellement la date de parution.
- Ouvrage est une variable de type enregistrement; chacune de ces cinq données est un champ pouvant être simple ou structuré.

Les enregistrements

- Les enregistrements sont déclarés en VB avec le mot Type.
- Syntaxe :

```
Type NomEnregistrement
    Champ1 As type1
    Champ2 As type2
    …
End Type
```

Champs simples

- Exemple :

```
Type ouvrage
    code as Integer
    titre As String*40
    auteur As String*50
    editeur As String*50
    dateparution As Date
End Type
```

Champ structuré

```
Type Date
    jour As Integer
    mois As Integer
    annee As Integer
End Type
```

Les enregistrements

- Exemple :

```
Type ouvrage
    code as Integer
    titre As String*40
    auteur As String*50
    editeur As String*50
    dateparution As Date
End Type
```

```
Type Date
    jour As Integer
    mois As Integer
    annee As Integer
End Type
```

- Pour accéder à un champ :

```
Dim livre  As ouvrage
livre.auteur = "Durand "
livre.dateparution.annee = 1980
 'on s'aperçoit ici que l'on pourrait remplacer livre par un tableau dans le type
 ouvrage…Dim livre(1 To 10000) as ouvrage…
 livre(9).auteur = "Durand" s'il s'agit du neuvième livre de la liste…
…
```

Les enregistrements – Exemple

Un étudiant est défini par son nom, son prénom, sa date de naissance et sa note :

```
Private Type Etudiant
        nom As String * 40
        prenom As String * 40
        dateNaissance As Date
        note As Double
End Type
```

Une classe peut contenir au plus 30 étudiants :

```
Const NbMax = 30                        'pour le nombre limite d'étudiants
Private Type Classe
    liste(NbMax) As Etudiant            'la liste est un tableau d'étudiants
    nbr As Integer                      'le nombre réel des étudiants
End Type
```

On déclare ensuite la classe d'étudiants :

```
Dim c As Classe
```

Les enregistrements – Exercice

- L'exemple précédent sera complété dans le prochain cours sur les interfaces graphiques…
- Comment définir une matrice ?
- Créer un programme qui affiche le nombre de lignes et de colonnes d'une matrice saisie sur la Feuil1 du classeur.

Partie 4 Développement Rapide d'interfaces

- Boîtes de dialogue de base
- UserForm et éditeur graphique
- Les différents contrôles

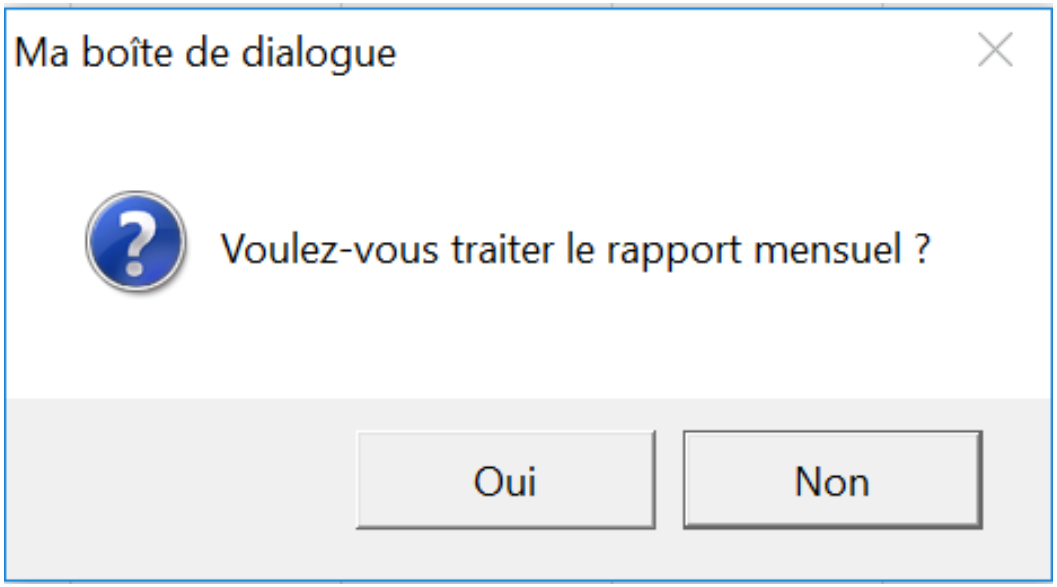
Ma MsgBox

- Boite de dialogue de base, "personnalisable"

vbOKOnly	0	N'affiche que le bouton ok
vbOKCancel	1	Ok et Annuler
vbAbortRetryIgnore	2	Abandonner, Recommencer, Ignorer
vbYesNoCancel	3	Oui, Non, Annuler
vbYesNo	4	Oui, Non
vbRetryCancel	5	Recommencer, Annuler
vbCritical	16	Icône message critique
vbQuestion	32	Icône Question
vbExclamation	48	Icône exclamation
vbInformation	64	Icône Information
vbDefaultButton1	0	Le premier bouton est par défaut
vbDefaultButton2	256	Le 2ième bouton est par défaut
vbDefaultButton3	512	Le 3ième bouton est par défaut
vbDefaultButton4	768	Le 4ième bouton est par défaut
vbSystemModal	4096	Suspend tout jusqu'à une réponse de l'utilisateur

Exemple de MsgBox

```
Sub MaMsgBox()
Dim Config As Integer
Dim Reponse As Integer
Config = vbYesNo + vbQuestion + vbDefaultButton2
Reponse = MsgBox("Voulez-vous traiter le rapport mensuel ?", Config, "Ma boîte de dialogue")
If Reponse = vbYes Then
    MsgBox ("Youpi")
Else
    MsgBox ("Ohhhh")
End If
End Sub
```

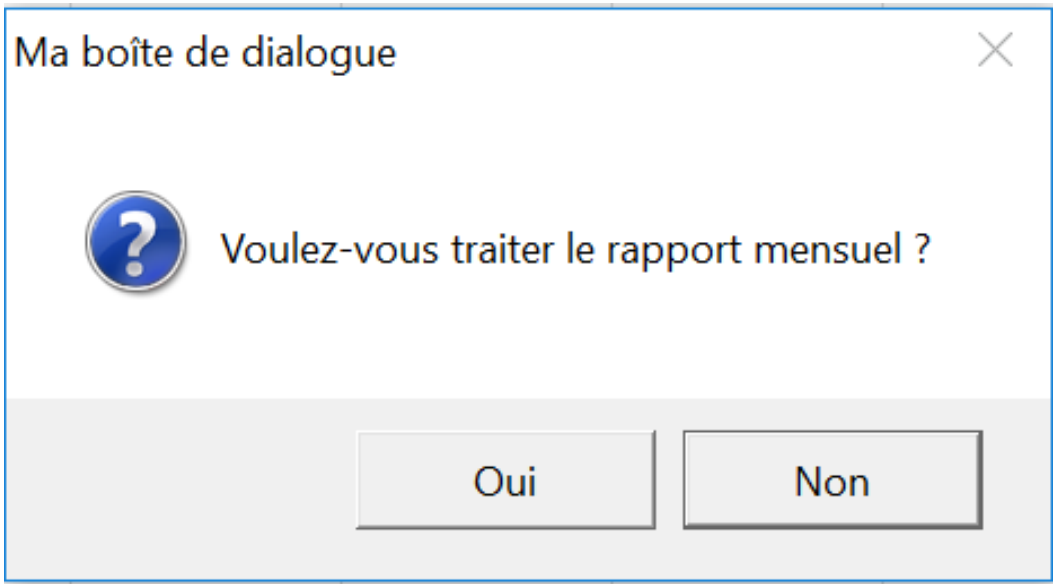


Exemple de MsgBox (2)

```
Sub MaMsgBox()

If MsgBox("Voulez-vous traiter le rapport mensuel ?", 292, "Ma boîte de dialogue") = vbYes Then
    MsgBox ("Youpi")
Else
    MsgBox ("Ohhh")
End If


End Sub
```



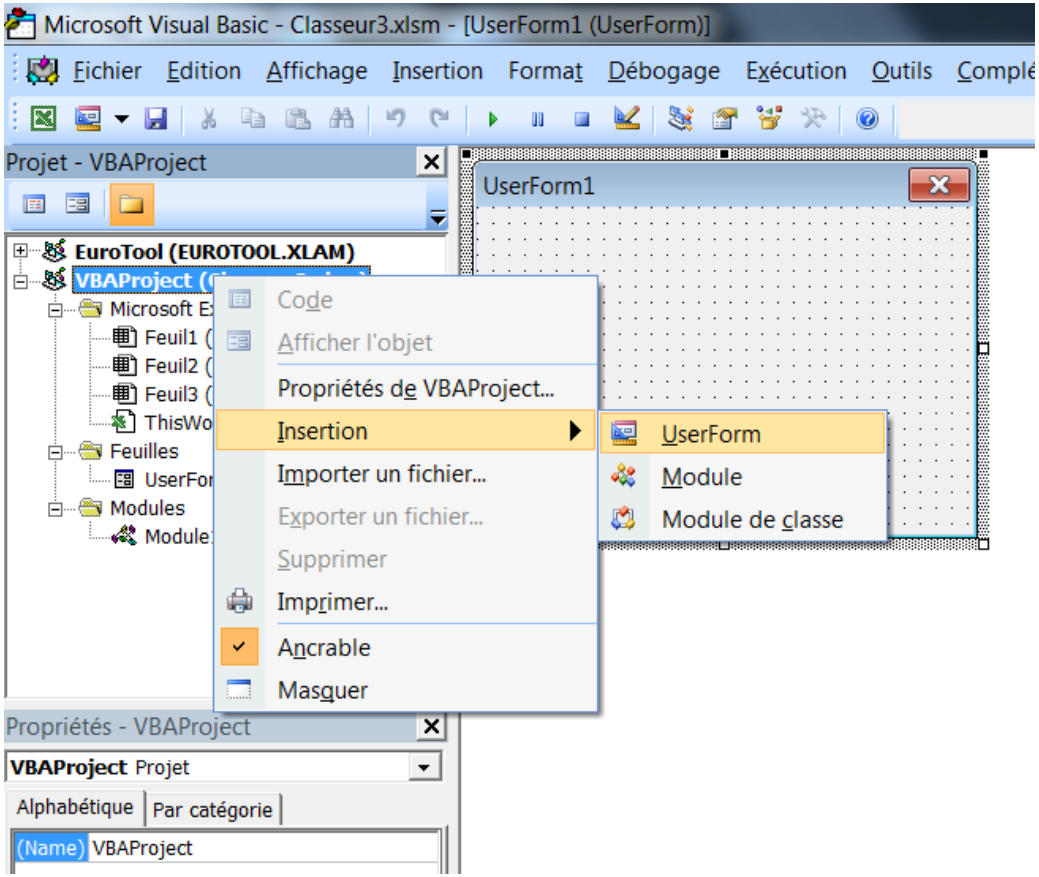
Autres fenêtres classiques

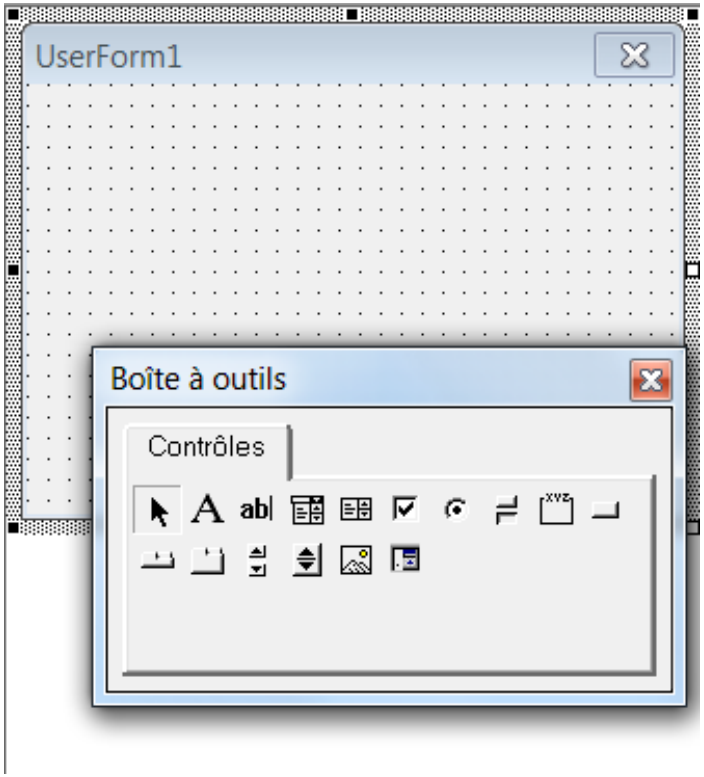
- InputBox
  - Permet de récupérer une valeur entrée par l'utilisateur
- GetOpenFileName
  - Ouvre une boite de dialogue permettant de sélectionner un fichier sur le disque dur
- GetSaveAsFileName
  - Boîte de dialogue pour enregistrer un fichier

Créer un UserForm personnalisé

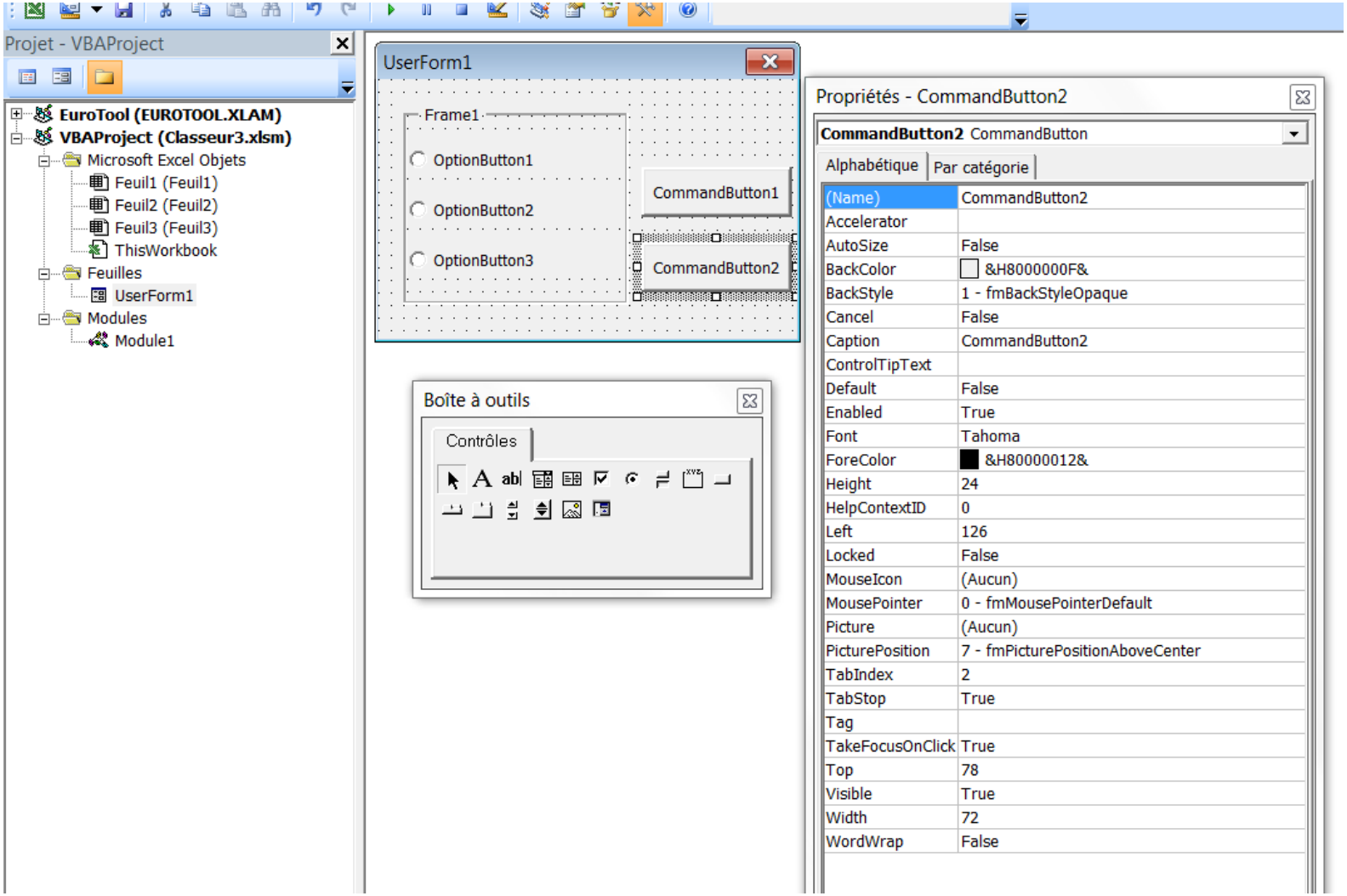
- 1) Imaginer la boîte de dialogue: à quoi sert-elle, où sera-t-elle utilisée?
- 2) Créer un nouvel objet userForm dans l'éditeur VBE
- 3)Ajouter des contrôles
  - Zones de textes
  - Boutons radio
  - Cases à cocher
  - Listes
- 4) Modifier les propriétés des éléments
- 5) Ecrire les procédures d'évènements des différents contrôles
- 6)Ecrire la procédure affichant la boîte de dialogue.

L'éditeur graphique de USerForm



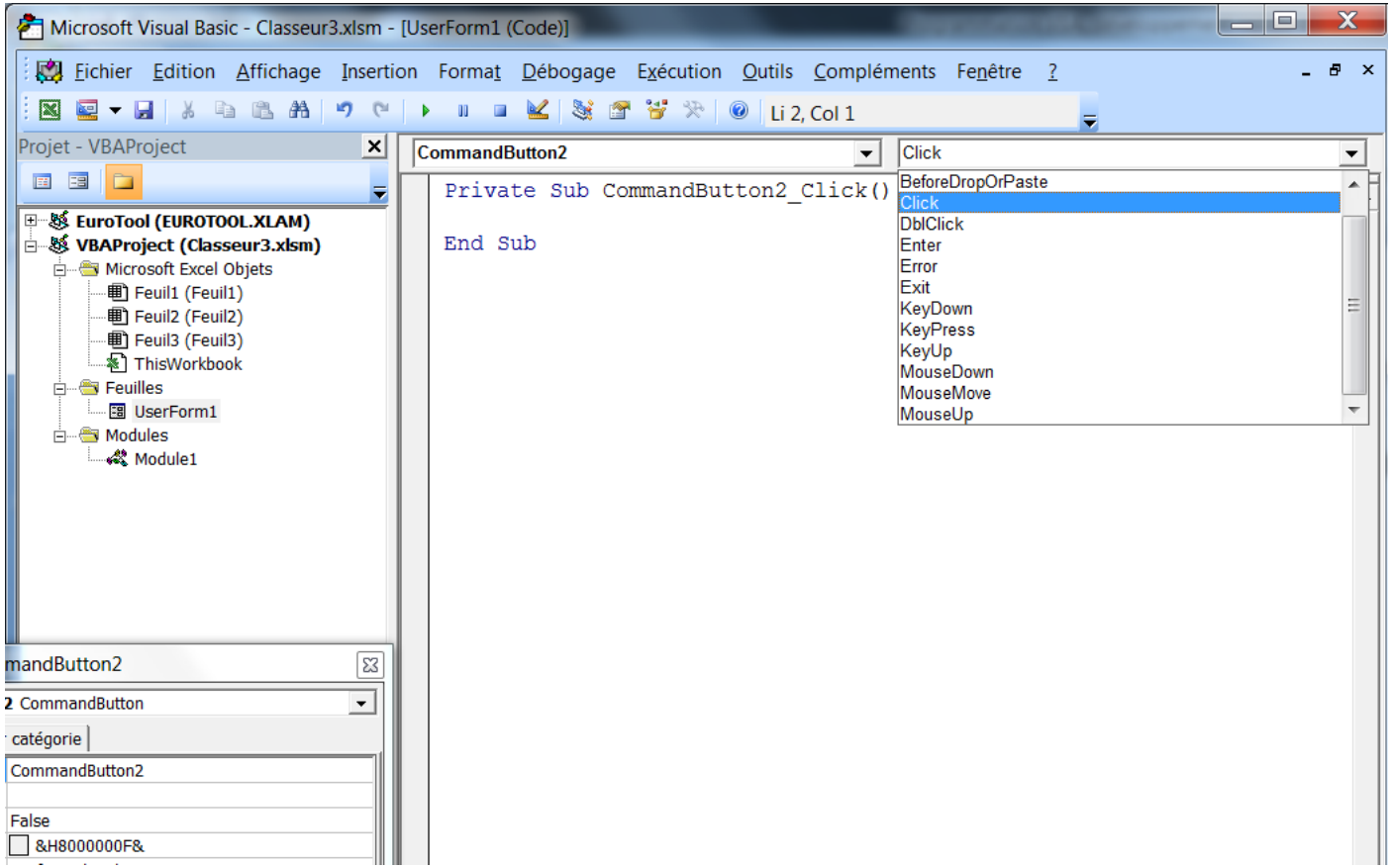


Editer les propriétés des contrôles



Editer les procédures d'évènements

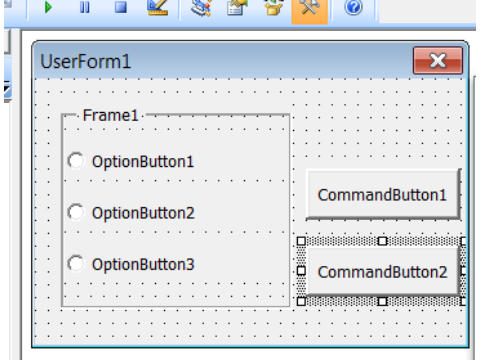
- Double-cliquer sur le contrôle dont on veut éditer les évènements



Détails sur les contrôles (1)

- La case à cocher
  - Accelerator
  - Value
- Zone de liste modifiable
  - ListRow
  - RowSource
  - Value
- Bouton
  - Annuler
  - Default
- Image
  - picture

Détail sur les contrôles (2)

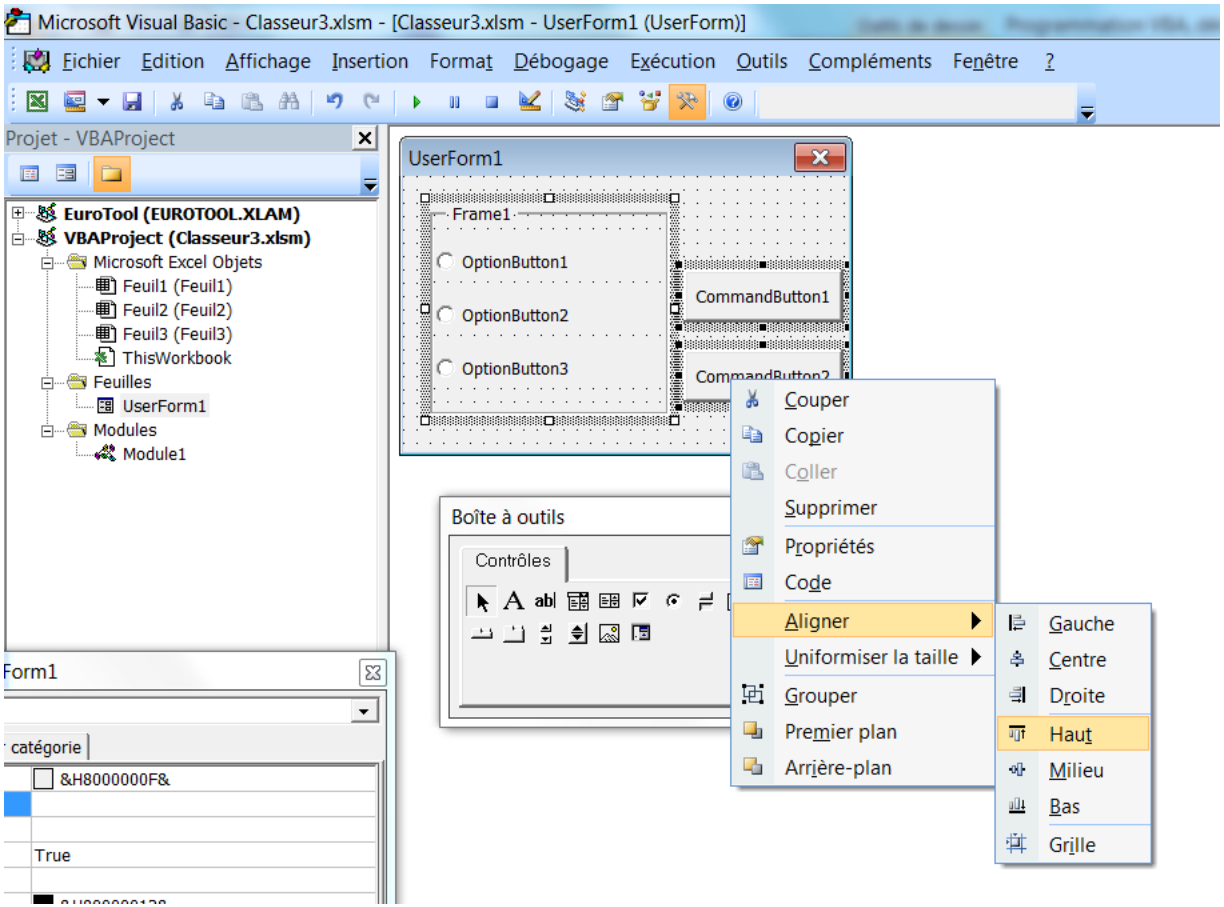


- Multipage: faire des onglets.
- Bouton d'option (bouton radio): sélection d'UNE option parmi plusieurs.
  - Un groupe est défini par tous les boutons ayant la même propriété GroupName ou si tous les boutons sont dans un même cadre.
- RefEdit: permettre à l'utilisateur de sélectionner une plage dans une feuille de calcul
- Barre de défilement: ascenceur permettant de définir/ afficher une valeur

Détails sur les contrôles (3)

- Contrôle Toupie: 2 boutons fléchés permettant d'incrémenter / décrémenter une valeur
- Contrôle zone de texte: insérer du texte!
- Bouton bascule: similaire à la case à cocher

Dimensionner / Aligner les contrôles



Partie 5 Les structures complexes

- Les piles
- Les files

Une pile

- Analogie de la pile d'assiettes
  - Last In First Out (LIFO)
- Opérations possibles
  - Insérer un élément dans une pile
  - Supprimer un élément d'une pile
  - Élément du sommet de la pile
  - Création d'une pile vide
  - Tester si une pile est vide

Mise en œuvre d'une pile

- Plusieurs façons de faire :
  - En particulier, à l'aide d'un tableau :
    - Le nombre max d'éléments dans la pile
    - Le contenu de la pile
    - Un indice pour pointer sur le sommet de la pile

Mise en œuvre d'une pile : exemple

- Type de données :

```
Const NMAX=30

Type TPile
        contenu(NMAX) as Integer
        sommet As Integer
End Type
```

Mise en œuvre d'une pile : exemple

```
Function PileVide(p As TPile) As Boolean
        If (p.sommet = -1) Then
                PileVide = True
        Else
                PileVide = False
        End If
End Function
```

Mise en œuvre d'une pile : exemple

```
Function PilePleine(p As TPile) As Boolean
        If (p.sommet = NMAX - 1) Then
                PilePleine = True
        Else
                PilePleine = False
        End If
End Function
```

Mise en œuvre d'une pile : exemple

```
Sub Empiler(p As TPile,elt As Integer)
   If (PilePleine(p) = False) Then
         p.sommet = p.sommet + 1
         p.contenu(p.sommet) = elt
   Else
         MsgBox("La pile est pleine !")
   End If
End Sub
```

Mise en œuvre d'une pile : exemple

```
Sub Depiler(p As TPile)
   If (PileVide(p) = False) Then
         p.sommet = p.sommet - 1
   Else
         MsgBox("La pile est vide !")
   End If
End Sub
```

Mise en œuvre d'une pile : exemple

```
Function sommet(p As TPile) As Integer
    If (PileVide(p) = False) Then
        sommet = p.contenu(p.sommet)
    Else
        MsgBox("La pile est vide !")
    End If
End Function
```

Une File

- Analogie de la file d'attente
  - First In First out (FIFO)
- Opérations principales
  - Insertion d'un élément
  - Suppression d'un élément (le plus ancien de la file)
  - Quel est l'élément le plus ancien de la file ?
  - Création d'une file vide
  - Est-ce qu'une file est vide ?

Mise en œuvre d'une file

- Plusieurs façons de faire :
  - En particulier, à l'aide d'un tableau :
    - Le nombre max d'éléments dans la file
    - Le contenu de la file
    - Un indice début qui pointe sur l'élément le plus ancien de la file
    - Un indice fin qui pointe sur le dernier élément inséré dans la file

Mise en œuvre d'une file : exemple

- Type de données :

```
Const NMAX=30

Type TFile
        contenu(NMAX) as Integer
        debut As Integer
        fin As Integer
End Type
```

Mise en œuvre d'une file : exemple

```
Function FileVide(f As TFile) As Boolean
        If (f.debut = f.fin) Then
                FileVide = True
        Else
                FileVide = False
        End If
End Function
```

Mise en œuvre d'une file : exemple

```
Function FilePleine(f As TFile) As Boolean
        If (f.debut=(f.fin + 1) mod NMAX) Then
                FilePleine = True
        Else
                FilePleine = False
        End If
End Function
```

Mise en œuvre d'une file : exemple

```
Sub Enfiler(f As TFile,elt As Integer)
   If (FilePleine(p) = False) Then
         f.contenu(f.fin) = elt
         f.fin = (f.fin +1) mod NMAX
   Else
         MsgBox("La file est pleine !")
   End If
End Sub
```

Mise en œuvre d'une file : exemple

```
Sub Defiler(f As TFile)
   If (FileVide(f) = False) Then
         f.debut = (f.debut+1) mod NMAX
   Else
         MsgBox("La file est vide !")
   End If
End Sub
```

Mise en œuvre d'une file : exemple

```
Function Tete(f As TFile) As Integer
    If (FileVide(f) = False) Then
        Tete = f.contenu(f.debut)
    Else
        MsgBox("La file est vide !")
    End If
End Function
```

Mise en œuvre d'une pile : exercice

- Créer un module pour simuler les piles.
- Votre module doit contenir la procédure suivante :

```
Sub main()
        Dim p As TPile
        Dim i As Integer
        i = 1
        While PilePleine(p)=False
                Call Empiler(p,i)
                i = i + 1
        WEnd
        While PileVide(p)=False
                MsgBox(Sommet(p))
                Call Depiler(p)
        Wend
End Sub
```

Mise en œuvre d'une file : exercice

- Une personne est définie par un numéro, un nom et un prénom.
- Définir une structure PFile correspondant à une file de personne.
- Adapter les différentes procédures et fonctions pour qu'elles manipulent des files de personnes.

Conclusion…

- A pratiquer en TP et sur le projet !

Merci

Hervé Hocquard (hocquard@labri.fr)

Alexis Clay-Hervé Hocquard