C# - FICHE PRATIQUE N° 1

Gestion d'une liste déroulante
Projet, formulaire, contrôle, programmation événementielle.

A/ Présentation (très) rapide de DotNet et C#

1. Le framework DotNet

Le framework (cadre de travail) DotNet est constitué d'un ensemble de fonctions de haut niveau permettant de traiter l'ensemble des besoins d'un programmeur : interface graphique, accès aux bases de données, génération de pages Web, …

Ce framework est disponible gratuitement pour toutes les plateformes Windows, des projets de portage vers d'autres systèmes sont en cours.

En fait, il s'agit d'un environnement d'exécution (une machine virtuelle) pour des programmes écrits en CLR (Common Language Runtime). Les programmes ressemblent à des exécutables, mais ils sont en réalité pris en charge par le framework.

Pour générer un programme, il faut compiler un source en langage intermédiaire MSIL (Microsoft intermediate language). Ce source peut être écrit en VB.net, C++.net, C#, etc (le support de 27 langages est prévu).

L'idée est donc d'offrir une indépendance vis à vis du langage lors de la programmation, et une indépendance vis à vis du système d'exploitation lors de l'exécution (ceci n'est actuellement vrai qu'avec les systèmes Microsoft).

2. Le langage C#

Pour pouvoir être « dotnet compatible », un langage de programmation doit satisfaire aux CLS (common language spécifications). Ceci débouche inévitablement par l'adoption de normes communes (types et taille des données manipulées, capacités en matière de POO, etc…). Certains langages implémentés sont un sous-ensemble du CLS (VB par exemple), d'autres se trouvent appauvris par cette transformation (C++ par exemple). Parmi tous ces langages, il en est un nouveau, créé spécialement pour DotNet et répondant exactement aux spécifications de CLS : C# (prononcer C Sharp).

Ce langage est censé être constitué du meilleur de C++ et du meilleur de Java, l'avenir le dira…

3. VisualStudio.net

Pour développer des programme en C#, il suffit de disposer du « .net framework SDK » disponible librement en téléchargement.

La production d'un programme passe alors par les phases suivantes :

- Edition du source en C# à l'aide d'un éditeur de texte (monprog.cs)
- Compilation à l'aide du SDK (csc monprog.cs). On obtient un fichier monprog.exe qui n'est pas un exécutable !
- Exécution en lançant « monprog » : le code intermédiaire est pris en charge par dotnet pour être exécuté.

Evidemment, cette méthode de production de programme nécessite la connaissance du langage (syntaxe, bibliothèque, etc). De plus, la création des interfaces graphiques est un peu laborieuse. VisualStudio.net est l'environnement de développement multi-langages proposé par Microsoft (pas gratuitement cette fois !). Il s'agit d'un outil de développement graphique : on travaille à l'aide d'une interface conviviale et VisualStudio génère un programme exécutable par dotnet.

Remarque : il existe des produits concurrents, notamment CsharpDevelop, totalement libre.

Le développement d'une application s'organise en général de la manière suivante :

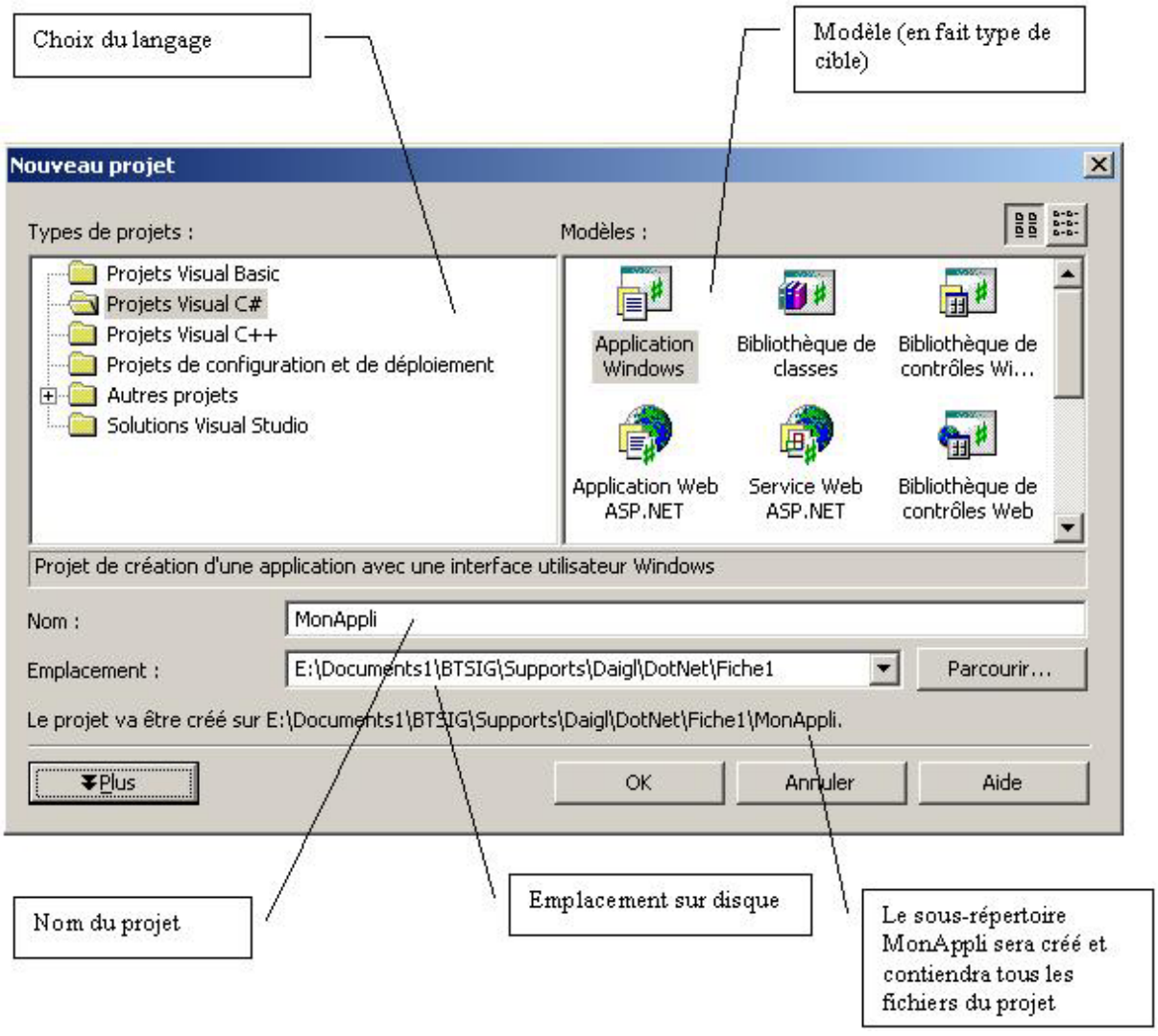
- construction de l'interface,
- validation de cette interface,
- ajout du code nécessaire,
- mise au point.

4. Notion de formulaire (ou fenêtre ou form)

Chaque application consiste en un ensemble de formulaires. Chaque formulaire créé est une classe héritant de la classe générique Form (System.Windows.Forms.Form). Il est possible d'y placer un certain nombre d'objets : texte, zone de saisie… Chacun de ces objets devient un membre de la classe correspondant au formulaire. C'est l'environnement VisualStudio qui se charge de créer une instance de cette classe de formulaire.

5. Création d'un projet

Cliquer sur « nouveau projet » à partir de la page de démarrage.



La barre d'outils principale permet d'exécuter les tâches d'ordre général (enregistrer, exécuter, …).

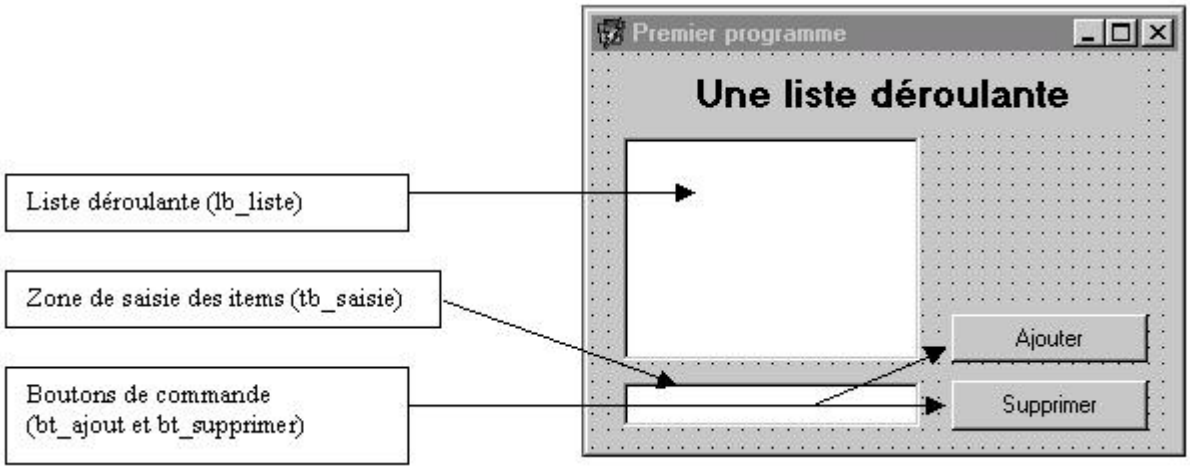
Les autres fenêtres importantes sont la « boîte à outils » et la « fenêtre des propriétés ». Vous les découvrirez très vite…

Remarque : VS crée une « solution », cette « solution » contenant un « projet ». Une solution est en fait un ensemble de projets destiné à résoudre un problème particulier. Vous aurez sans doute un seul projet par solution la plupart du temps.

B/ Une première application

1. Réalisation de l'interface

Nous allons réaliser une petite application de gestion d'une liste déroulante. L'utilisateur peut ajouter et supprimer des items à la liste.



- Lancez VS (Visual Studio) et créez un nouveau projet si ce n'est déjà fait. Nommez-le « fiche1 » et enregistrez-le dans votre répertoire personnel.
- Le premier formulaire de l'application est créé automatiquement. Nommez-le « Fm_principal » (propriété name)
- ATTENTION : IL FAUT SYSTEMATIQUEMENT NOMMER LES COMPOSANTS DES LEUR CREATION. LES NOMS UTILISES DOIVENT ETRE « CORRECTS »… Ne confondez pas les propriétés « Text » et « Name » !!!
- Ajouter un composant "ListBox", éditez les propriétés de la liste créée à l'aide de la fenêtre des propriétés et nommez-la "lb_liste". Demandez une liste triée.
- Ajoutez un composant "TextBox" et nommez-le "tb_saisie".
- Ajoutez deux composants "Button", nommez-les "bt_ajout" et "bt_supprimer", indiquez le texte des boutons et faites du bouton "bt_ajout" le bouton par défaut (il s'agit d'une propriété du formulaire).
- Ajoutez un composant "Label" : texte "Une liste déroulante", fonte Ms Sans sérif 14 gras.
- Indiquez le texte de la barre de titre de la fenêtre : "Premier programme" (propriété Text du formulaire).
- Testez votre interface en cliquant sur le bouton "Démarrer" de la barre d'outils.

2. Ajout du code

Bouton "Ajouter"

Editer la méthode correspondant à un clic sur ce bouton de commande (Evénement Click du bouton) :

```
private void bt_ajout_Click(object sender, System.EventArgs e)
{
    lb_liste.Items.Add(tb_saisie.Text);
    tb_saisie.Text="";    // ou tb_saisie.Clear()
    tb_saisie.Focus();
}
```

En appuyant sur F1 après avoir sélectionné un élément du formulaire, on obtient une aide sur le composant concerné. N'hésitez pas à parcourir cette aide.

Bouton "Supprimer"

Voici le code correspondant à un clic sur ce bouton (n'oubliez pas de consulter l'aide en ligne) :

```
private void bt_supprimer_Click(object sender, System.EventArgs e)
{
    lb_liste.Items.RemoveAt(lb_liste.SelectedIndex);
}
```

Messages d'erreur

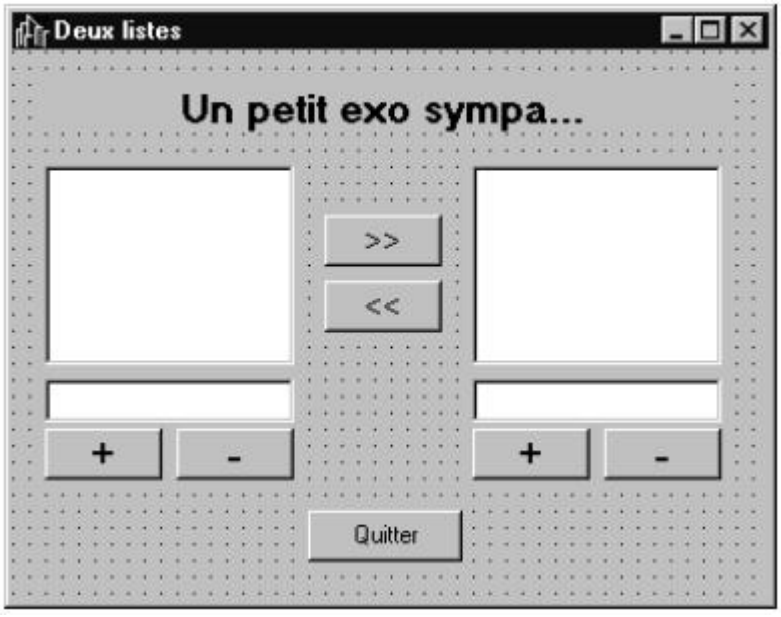
Vous allez ajouter un message d'erreur dans le cas où l'utilisateur clique sur "Ajouter" avec une zone de saisie vide et un autre dans le cas où il clique sur "Supprimer" sans item sélectionné dans la liste. Utilisez pour cela la classe MessageBox, consultez l'aide…

3. Dernières retouches

Mettez au point les derniers petits détails : problèmes de focus au lancement (événement Activated du formulaire) et raccourcis clavier pour les boutons (il suffit d'indiquer "&Ajouter" et "&Supprimer" dans leur propriété "Text").

C/ A vous de jouer…

Il s'agit de réaliser une application gérant deux listes déroulantes dont l'interface est la suivante :



- Les boutons + et – permettent d'ajouter et de supprimer des éléments aux listes déroulantes.
- Le bouton >> fait passer l'élément sélectionné dans la liste de gauche vers la liste de droite.
- Le bouton << a l'effet inverse.
- Un double-clic sur un élément de la liste de gauche permet de le faire passer dans celle de droite.
- Un double-clic sur un élément de la liste de droite permet de le faire passer dans celle de gauche.
- Le bouton Quitter permet de quitter l'application après une demande de confirmation.

Quelques indications utiles

- Vous pouvez créer un nouveau projet dans la solution « fiche1 », ou créer une nouvelle solution.
- Pour mettre fin à l'exécution, on peut utiliser la méthode statique « exit » de la classe « Application ».
- Consultez l'aide de la classe « MessageBox » pour la demande de confirmation de l'utilisateur.
- Il est possible de définir des méthodes (fonctions) personnelles dans le code d'un formulaire. A titre d'exemple, une fonction ajoutant le contenu d'une zone d'édition à une zone de liste serait bien utile ici. Le mot clé « this » désigne en permanence l'objet concerné par un appel de méthode (un peu le « me » d'Access). Cela permet entre autres d'obtenir la fonctionnalité de « code completion » avec les fonctions définies par le programmeur.