Annexe – Conventions de codage en C#.

1. Introduction

Nous examinerons ici les recommandations écrites de Microsoft en ce qui concerne les conventions de codage en langage C#.

Tout langage de développement possède généralement un style de programmation qui lui est propre. Ce style ne fait pas partie du langage lui-même mais représente un ensemble de conventions concernant le langage.
Ces conventions incluent par exemple des règles de nommage des variables, la casse à utiliser selon les cas, et les modes d'utilisation des classes, méthodes et fonctions.

Si la majorité des développeurs suivent les mêmes conventions, la lecture et la maintenance du code est facilitée pour tout le monde et il sera ainsi plus facile pour un développeur de comprendre le code écrit par un autre.

Ces conventions sont consultables dans l'aide MSDN aux adresses suivantes :

Pour les conventions générales de conception :
ms-help://MS.NETFrameworkSDKv1.1/cpgenref/html/cpconnetframeworkdesignguidelines.htm

Pour les conventions de nommage des éléments :
ms-help://MS.NETFrameworkSDKv1.1/cpgenref/html/cpconnamingguidelines.htm

Bien sûr, l'ensemble de toutes les recommandations de codage pour adopter de bonnes pratiques de programmation représente une quantité importante d'informations à "digérer".
Dans le cadre de cette annexe, nous ne ferons que reprendre les points les plus importants de ces recommandations.
Si vous voulez être certains que votre code suit à la lettre les recommandations de codage en vigueur, il vous est conseillé de vous reporter à la documentation MSDN.

2. Conventions de nommage

Les conventions de nommage ont toujours fait l'objet de discussion passionnées entre les développeurs et architectes d'application, chacun débattant longuement des avantages et inconvénients de leurs propres méthodes.

Ainsi, dans Visual Basic 6, les développeurs utilisaient une convention qui consistait à préfixer les noms de variables par une abréviation représentant le type de la variable déclarée. Par exemple, une variable de type `string` nommée "Nom" se serait écrite : `Dim strNom as string.`
Les conventions dépendent cependant également des langages et plateformes utilisées.
Par exemple, les développeurs C++ qui programment sur des plateformes Windows utilisent des préfixes `psz` ou `lpsz` pour indiquer des chaînes de caractères (ex: `char *pszResult;` ou encore `char *lpszMessage;`)
Mais, sur des machines Unix, il n'es pas rare de voir des développeurs n'utilisant pas ces mêmes préfixes (ex: `char *Result;` ou encore `char *Message;`).

*La convention selon laquelle les noms de variables sont précédés de lettres représentant le type de données est connue sous le nom de **notation hongroise**. Il est ainsi possible, pour un développeur, de reconnaître immédiatement le type de la variable à la lecture de son nom.*

En ce qui concerne les conventions de codage en C#, nous n'utiliserons pas la notation hongroise. En effet, dans le Framework .NET, **tout est objet**. On ne peut donc pas préfixer les noms de variables avec un préfixe indiquant le type de données (ou alors, on ne préfixerait que les variables représentant les types de données intrinsèques).

Intuitivement on peut dire que les noms devraient refléter la destination de l'élément et n'entrer en conflit avec aucun autre nom.
Dans le Framework .Net, la philosophie générale de la dénomination des variables suggère que le nom de la variable devrait refléter **l'emploi de l'instance de cette variable et non son type de données** (par exemple `Longueur` est un bon nom de variable alors que `ValeurEntière` ne l'est pas).

3. Casse des noms d'éléments (Pascal Casing et Camel Casing)

Pour résumer, on emploie deux types de casse de caractères appelés :

- **Pascal Casing** (Une majuscule au début de chacun des mots composant le nom d'élément).
- **Camel Casing** (on commence le nom en minuscules, puis, une majuscule au début de chaque mot.

Exemple de Pascal Casing : `WindowHeight, WriteLine`
Exemple de Camel Casing : `windowPosition, integralHeight`

Il est **déconseillé** d'utiliser la syntaxe consistant à séparer les mots par un caractère "underscore" comme dans ("`Window_Position`")

De même, il est **déconseillé** d'écrire les noms de constantes en majuscules, comme c'était le cas dans certains langages précédents (comme dans Visual Basic 4,5 et 6 par exemple).

On peut également prendre en compte la portée des éléments pour savoir quelle casse (Pascal ou Camel) utiliser pour nommer un élément.
En général, tous les éléments publics sont écrits en Pascal Casing alors que les éléments privés seront plutôt écrits en Camel Casing.

Par exemple :

- les noms de tous les paramètres passés à une méthode devraient être en casse Camel :
```
public void RecordSale(string salesName, int quantity)
```

- vous devriez utliser la casse Camel pour établir une distinction entre deux éléments qui, autrement auraient exactement le même nom. Un exemple commun est celui d'une propriété encapsule un champ :

```
private string nomSalarie;

public string NomSalarie
{
      get
      {
            return nomSalarie;
      }
```

```
            set
            {
                  nomSalarie = value;
            }
      }
```

Pour ce qui est des conventions, le code précédent est acceptable.

Il est important de rester attentif à la casse des éléments que vous définissez, et, dans tous les cas de figure, d'adopter une règle pour la casse et de s'y tenir. Il est pratique par exemple de savoir immédiatement que la variable `nomSalarie` est une variable privée, juste en analysant sa casse.
En tout état de cause, le langage C# est sensible à la casse et il est donc important de porter une attention particulière aux conventions de nommage des éléments.

4. Style des Noms d'éléments.

Essayez d'être cohérents dans le style que vous donnez aux noms des éléments que vous définissez. Par exemple, essayez de toujours garder dans le même ordre les noms et les verbes.
Si vous appelez une méthode `RechercherFichiersTexte()`, nommez une autre méthode `RechercherFichierImages()` au lieu de `FichiersImageRechercher()`.

5. Nommage des Namespaces (espaces de noms).

Le nom des namespaces est particulièrement important dans l'organisation de votre code et peut grandement faciliter et normer la manière dont les objets communiquent entre eux.

De plus, une grande attention est nécessaire (surtout si vous développez des composants réutilisables) afin d'éviter toute collision possible entre le nom complet (c.à.d. le nom de tous les namespaces contenant + le nom de l'élément) de l'un de vos éléments et le nom d'un autre élément que vous n'avez pas développé.

Par exemple, si vous utilisez le même nom de namespace pour un logiciel alors que ce nom est déjà utilisé dans un autre logiciel, des problèmes de collisions de noms risquent de se poser !
De ce fait, la **création d'un namespace de haut niveau** (comme par exemple le nom de votre société) représente toujours une bonne solution de ce point de vue.
Vous pourrez par la suite créer à l'intérieur du namespace de haut niveau, des namespaces successifs représentant la technologie, le département, ou encore la couche (couche d'accès aux données, couche métier, etc..) à laquelle sont destinées les classes contenues.
Pour information, Microsoft recommande les namespaces commençant par :

```
<NomDeSociété>.<NomLogiciel>
```

Comme par exemple : `ratp.métropolitain.personnel` ou
`ratp.métropolitain.matériel`