Les espaces de noms

Le code d'une application peut très vite contenir plusieurs dizaines, centaines ou milliers de types différents. Afin d'organiser le projet, il est tout à fait possible de créer une structure hiérarchique de dossiers regroupant les types par utilité ou par contexte d'utilisation. En plus de cette organisation physique, il est possible de créer une organisation logique à l'aide du concept d'espace de noms (**namespace** en anglais).

1. Nomenclature

Un espace de noms est composé de plusieurs identificateurs séparés par l'opérateur `.`, chacun des identificateurs faisant office de conteneur logique. Voyons quelques exemples d'espaces de noms :

```
System
System.Windows
System.Data.SqlClient
```

Ces trois espaces de noms font partie du framework .NET. `System` contient les types de base de .NET, comme les types primitifs, `System.Windows` est le conteneur logique des types de base pour la création d'applications fenêtrées. Enfin `System.Data.SqlClient` contient les types de la brique ADO.NET spécifiques aux bases de données SQL Server.

En plus de l'aide à la structuration, l'utilisation d'espaces de noms permet d'avoir plusieurs types dont le nom est identique : pour éviter toute ambiguïté entre ces types, il est possible d'utiliser le nom pleinement qualifié du type que l'on souhaite utiliser. Ce nom complet est la concaténation de l'espace de noms, de l'opérateur point (`.`) et du nom du type.

Le framework .NET comprend plusieurs classes nommées `DataGrid`. Celles-ci se trouvant dans des espaces de noms différents, il est possible de faire référence à chacun de ces types sans ambiguïté de la manière suivante :

```
System.Windows.Controls.DataGrid grilleWpf;

System.Windows.Forms.DataGrid grilleWindowsForms;
```

Pour inclure un type dans un espace de noms, il suffit de déclarer le type à l'intérieur d'un bloc `namespace`.

```
namespace MonApplication.Rss
{
    public class LecteurFluxRss
    {
    //...
    }
}
```

2. using

Très souvent, il ne sera pas nécessaire d'utiliser ce nom complet explicitement. Il est en effet possible de déclarer dans le code que nous souhaitons utiliser les classes d'un ou plusieurs espaces de noms particuliers à l'aide du mot-

clé `using`.

Dans ce contexte, ce mot-clé n'a rien à voir avec l'utilisation qui a été détaillée dans le chapitre Les bases du langage à la section Autres structures.

Ainsi, si l'on souhaite utiliser les classes de l'espace de noms `System.Windows.Forms`, on pourra écrire le code suivant :

```
using System.Windows.Forms;

public class MaClasse
{
    private void CreerDataGrid()
    {
          DataGrid maGrille;
          // ...

          //On peut aussi utiliser le type DataGrid de l'espace
          //de noms System.Windows.Controls en spécifiant
          //le nom complet
          System.Windows.Controls.DataGrid maGrilleWpf;
    }
}
```

Depuis l'arrivée de C#6, le mot-clé `using` peut également être utilisé pour simplifier l'utilisation d'éléments statiques. Les membres statiques sont en effet toujours utilisés au travers du nom du type auquel ils appartiennent, ce qui peut amener à de nombreuses répétitions. L'instruction suivante utilise deux membres de la classe `System.Math` pour calculer le cosinus de π.

```
using System;

public class MaClasse
{
    private void ExecuterCalcul()
    {
          int cosPi = Math.Cos(Math.PI);
          ...
    }
}
```

L'utilisation de la combinaison de mots-clés `using static` suivie du nom d'un type statique permet de s'affranchir de l'utilisation du nom de ce type dans le fichier de code. L'exemple précédent pourrait ainsi être réécrit de la manière suivante :

```
using System;
using static Math;

public class MaClasse
{
    private void ExecuterCalcul()
```

```
    {
          int cosPi = Cos(PI);
          ...
    }
}
```

Attention ! L'utilisation de `using static` peut amener à un manque de lisibilité du code, voire à certains conflits empêchant la compilation. Il pourrait en effet être difficile, pour le développeur comme pour le compilateur, de déterminer à quelle classe appartient une fonction utilisée dans ce contexte.