

# Les génériques

(J-C Armici [jca.developpez.com](http://jca.developpez.com), [www.unvrai.com](http://www.unvrai.com))

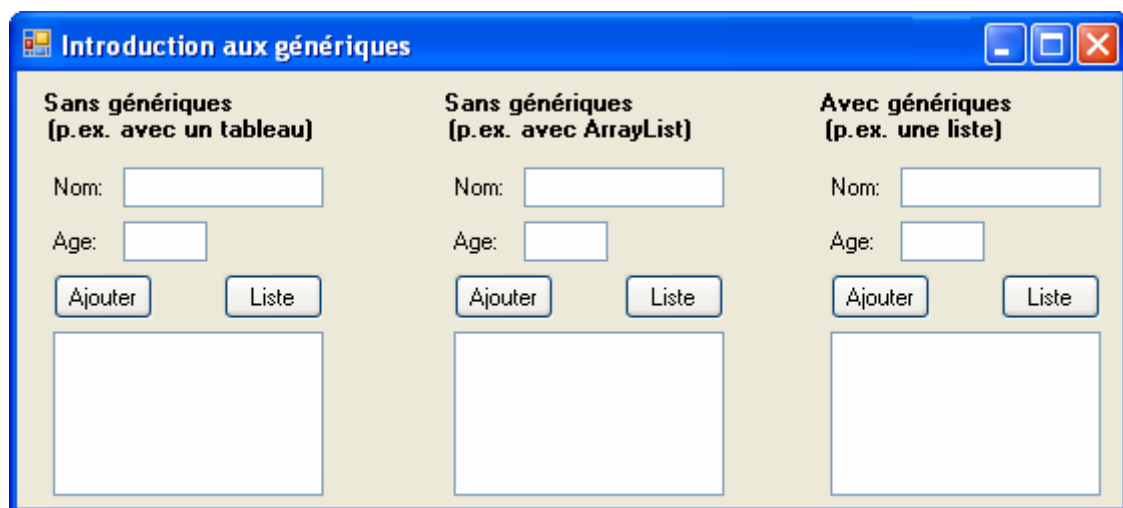
## Introduction

Dans la vie d'un programmeur plusieurs étapes (ou déclics) peuvent survenir lorsqu'il évolue dans le monde de la programmation orientée objet. Une de ces étapes est certainement l'héritage et le polymorphisme, qui ouvrent la voie vers la réutilisabilité du code.

Une étape importante est également la prise de conscience de l'accroissement de l'efficacité du développement en faisant appel aux génériques. Les génériques sont un nouveau concept dont l'utilisation peut faire penser à un développeur: "comment ai-je pu vivre sans ?".

## Exemple

Afin de mieux comprendre pourquoi et comment les génériques s'utilisent considérons l'exemple suivant:



Nous allons examiner successivement les trois parties séparées constituant ce programme de démonstration. Le but est le suivant:

- nous voulons pouvoir ajouter des personnes (caractérisées simplement par un Nom et un Age) dans une structure de données
- nous voulons également pouvoir obtenir (et afficher) la liste des personnes enregistrées

d'où la présence des boutons Ajouter et Liste.

Nous avons défini une classe `PersonneSimple` de la manière suivante (avec des membres `public` pour simplifier):

```
class PersonneSimple
{
    public string Nom = "";
    public int Age = 0;

    public PersonneSimple(string LeNom, int LAge)
    {
        Nom = LeNom;
        Age = LAge;
    }
}
```

Parmi les structures de données à disposition prenons tout d'abord un tableau, déclaré de la manière suivante:

```
const int MAX = 10; // nombre max de personnes
PersonneSimple[] tabPers = new PersonneSimple[MAX];
int nbPers = 0; // compteur de personnes
```

Voici le code derrière le bouton Ajouter:

```
private void btnAJouter0_Click(object sender, EventArgs e)
{
    PersonneSimple p = new PersonneSimple(txtNom0.Text, int.Parse(txtAge0.Text));
    if (nbPers < MAX - 1)
    {
        tabPers[nbPers] = p; // la personne est placée dans le tableau
        nbPers++;
    }
    else
        MessageBox.Show("Limite de 10 personnes atteinte");
}
```

Comme on peut le voir, p étant de type PersonneSimple, l'affectation est sans surprise. Voici comment parcourir le tableau et afficher les informations des personnes dans un listBox.

#### 1<sup>ère</sup> version

```
private void btnListe0_Click(object sender, EventArgs e)
{
    lbPersonnes0.Items.Clear();
    for (int i=0; i < nbPers; i++)
    {
        lbPersonnes0.Items.Add(tabPers[i].Nom + " "
                               + tabPers[i].Age.ToString());
    }
}
```

#### 2<sup>ème</sup> version (fonctionnellement équivalente)

```
private void btnListe0_Click(object sender, EventArgs e)
{
    lbPersonnes0.Items.Clear();
    for (int i=0; i < nbPers; i++)
    {
        PersonneSimple p = tabPers[i];
        lbPersonnes0.Items.Add(p.Nom + " " + p.Age.ToString());
    }
}
```

La seconde partie du programme de démonstration fait appel à un ArrayList plutôt qu'un tableau. Cette structure est beaucoup plus souple d'utilisation. Il s'agit quasiment d'un tableau dynamique d'objets. Sa déclaration est la suivante:

```
ArrayList alPers = new ArrayList();
```

Le bouton Ajouter comporte cette fois le code suivant:

```
private void btnAjouter_Click(object sender, EventArgs e)
{
    PersonneSimple p = new PersonneSimple(txtNom.Text, int.Parse(txtAge.Text));
    alPers.Add(p);
}
```

Pour obtenir la liste des personnes voici le code du bouton Liste:

```
private void btnListe_Click(object sender, EventArgs e)
{
    lbPersonnes.Items.Clear();
    foreach (PersonneSimple p in alPers)
    {
        lbPersonnes.Items.Add(p.Nom + " " + p.Age.ToString());
    }
}
```

La troisième partie du programme fait appel aux génériques, à une liste générique, dont voici la déclaration:

```
List<PersonneSimple> listePers = new List<PersonneSimple>();
```

Cette notation particulière indique que nous voulons utiliser une liste d'objets PersonneSimple.

L'adjonction (bouton Ajouter) ressemble aux deux versions précédentes:

```
private void btnAjouter2_Click(object sender, EventArgs e)
{
    PersonneSimple p = new PersonneSimple(txtNom2.Text, int.Parse(txtAge2.Text));
    listePers.Add(p);
}
```

Le code permettant d'obtenir a liste des personnes est le suivant:

```
private void btnListe2_Click(object sender, EventArgs e)
{
    lbPersonnes2.Items.Clear();
    foreach (PersonneSimple p in listePers)
        lbPersonnes2.Items.Add(p.Nom + " " + p.Age.ToString());
}
```

Pour terminer voici un exemple à méditer, tiré et adapté du livre "Professional .NET 2.0 Generics"<sup>1</sup>.

```
public class Hello<T>
{
    private T _parleur;

    public T Parleur
    {
        get { return _parleur; }
        set { _parleur = value; }
    }

    public void DisBonjour()
    {
        string bonjour = _parleur.ToString();
        Console.WriteLine(bonjour);
    }
}
```

```
public class ParleurAllemand
{
    public override string ToString()
    {
        return "Guten Tag";
    }
}

public class ParleurFrancais
{
    public override string ToString()
    {
        return "Bonjour";
    }
}

public class ParleurAnglais
{
    public override string ToString()
    {
        return "Hello";
    }
}
```

Voici le programme principal:

```
class Program
{
    static void Main(string[] args)
    {
        Hello<ParleurFrancais> fr = new Hello<ParleurFrancais>();
        fr.Parleur = new ParleurFrancais();
        fr.DisBonjour();

        Hello<ParleurAnglais> en = new Hello<ParleurAnglais>();
        en.Parleur = new ParleurAnglais();
        en.DisBonjour();

        Console.ReadLine();
    }
}
```

et le résultat lors de son exécution:

```
Bonjour
Hello
```

## Classes définissant des collections génériques

Voici les principales classes permettant l'utilisation des génériques:

- Dictionary
- LinkedList
- LinkedListNode
- List
- Queue
- SortedDictionary
- SortedList
- Stack

## Conclusion

L'utilisation des génériques à la manière de notre programme est d'une simplicité biblique comparée aux bénéfices que l'on peut en retirer. Il serait donc dommage de s'en priver.

Toutefois les génériques et leur utilisation ne se cantonnent pas à ce genre de situations simples. Ils représentent un filtre, une manière de faire pouvant s'appliquer de manière plus générale à tous les aspects de la programmation objet en .NET.

Pour aller plus loin dans la découverte et l'utilisation des génériques, il est possible de consulter un livre complètement et uniquement consacré aux génériques:

(1) "Professional .NET 2.0 Generics", *Tod Golding*, Wrox, ISBN: 0-7645-5988-5