**Les Collections**

Quand vous développez des applications, vous pouvez avoir besoin de traiter et stocker de nombreuses données en continu. L’utilisation des tableaux statiques pouvant être fastidieuse et peu optimisée, le Framework .NET implémente les collections.

**La classe ArrayList**

La classe ArrayList est la collection la plus basique. Sa taille varie au cours de l'exécution du programme, et elle estcapable de stocker des références sur des objets quelconques

[**Constructeurs publics**](javascript:void(0))

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [ArrayList](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.arraylist(v=VS.80).aspx) | Surchargé. Initialise une nouvelle instance de la classe [ArrayList](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist(v=VS.80).aspx). |

[**Propriétés publiques**](javascript:void(0))

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [Capacity](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.capacity(v=VS.80).aspx) | Obtient ou définit le nombre d'éléments que **ArrayList** peut contenir. |
| [Count](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.count(v=VS.80).aspx) | Obtient le nombre d'éléments réellement contenus dans **ArrayList**. |

La capacité de ArrayList correspond au nombre d'éléments que peut contenir ArrayList. La capacité initiale par défaut de ArrayList est 0. Lorsque des éléments sont ajoutés à ArrayList, la capacité augmente automatiquement par réallocation. La capacité peut être diminuée en appelant [TrimToSize](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.trimtosize(v=VS.80).aspx) (Capacity va être égale à Count en appelant TrimToSize) ou en définissant explicitement la propriété [Capacity](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.capacity(v=VS.80).aspx).

[**Méthodes publiques**](javascript:void(0))

***Ajouter des informations***

Une collection de type ArrayList étant désordonné et acceptant tout type d’objets, ajouter et supprimer des données à l’intérieur est très aisé. Pour ajouter des données, nous avons deux méthodes à notre disposition : Add et AddRange. ***Add*** permet de rajouter un objet à la suite des autres dans la liste tandis qu’***AddRange*** permet de rajouter des tableaux d’objets.

static void Main(string[] args)

{

ArrayList liste = new ArrayList();

liste.Add("lamiae");

string[] noms = new string[] { "ali", "sami", "Paul" };

liste.AddRange(noms);

for (int i = 0; i < liste.Count;i++ ) Console.WriteLine(liste[i]);

}

Add et AddRange ajoutent les données à la fin de la liste, si vous souhaitez insérer des données à n’importe quel endroit, vous devez utiliser les méthodes ***Insert*** (qui prend en arguments la position où vous voulez insérer la donnée ainsi que la donnée à insérer) ou ***InsertRange*** (qui prend en arguments la position où vous voulez insérer les données et le tableau de données à insérer):

static void Main(string[] args)

{

ArrayList liste = new ArrayList();

liste.Add("lamiae");

string[] noms = new string[] { "ali", "sami", "Paul" };

liste.InsertRange(0, noms);

liste.Insert(2, "yassine");

foreach (string s in liste) Console.WriteLine(s);

Enfin, si vous souhaitez remplacer une donnée, vous pouvez utiliser l’indexeur. Nous allons dans cet exemple remplacer la première valeur (index 1) par "badr".

liste[1] = "badr";

**Note :** Vous ne pouvez pas insérer des objets sortant des limites de la liste (par exemple, insérer une donnée à la position 5 alors que votre liste ne contient que 2 éléments lèvera une exception de type ArgumentOutOfRangeException).

***Supprimer des données***

Pour supprimer des données, nous disposons de 3 méthodes : Remove, RemoveAt, RemoveRange. L’utilisation des méthodes de suppression est quasi similaire à celle de l’ajout. ***Remove*** tente de supprimer un objet dans la collection, s’il ne trouve pas l’objet, il ne renvoie aucune erreur. ***RemoveAt*** permet de supprimer un objet à une certaine position en utilisant l’index. Enfin ***RemoveRange*** permet de supprimer plusieurs objets en définissant le point de départ avec l’index et le nombre d’objets à supprimer. Voici un exemple de leur utilisation :

static void Main(string[] args)

{

ArrayList liste = new ArrayList();

string[] noms = new string[] { "badr", "sami", "Paul", "lamiae", "yassine", "salma" };

liste.AddRange(noms);

liste.Remove("sami");

liste.RemoveAt(0);

for (int i = 0; i < liste.Count; i++) Console.WriteLine(liste[i].ToString());

liste.RemoveRange(0, 2);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*liste finale\*\*\*\*\*\*\*\*\*\*\*\*");

for (int i = 0; i < liste.Count; i++) Console.WriteLine(liste[i].ToString());}

**Note :** Vous ne pouvez pas utiliser RemoveAt et RemoveRange en dehors des limites de la liste.

***Autres manipulations***

Enfin, la classe ArrayList implémente également des méthodes plus puissantes afin de modifier le contenu de la collection. La méthode ***Clear*** qui permet de vider totalement la collection. La méthode ***IndexOf*** permet d’obtenir l’index d’un objet. La méthode ***Contains*** qui permet de vérifier qu’un objet existe dans la collection. La méthode ***Sort*** permet de trier notre collection. Enfin la méthode ***Reverse*** Inverse l'ordre des éléments dans la collection. Voici un exemple de leur utilisation :

static void Main(string[] args)

{

ArrayList liste = new ArrayList();

string[] noms = new string[] { "badr", "sami", "Paul", "lamiae", "yassine", "salma" };

liste.AddRange(noms);

Console.WriteLine(liste.IndexOf("yassine"));

Console.WriteLine(liste.Contains("Paul"));

liste.Reverse();

for (int i = 0; i < liste.Count; i++) Console.WriteLine(liste[i]);

}

[***Méthodes publiques***](javascript:void(0)) ***de la classe ArrayList***

|  |  |
| --- | --- |
| **Nom** | **Description** |
| [Add](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.add(v=VS.80).aspx) | Ajoute un objet à la fin de **ArrayList**. |
| [AddRange](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.addrange(v=VS.80).aspx) | Ajoute les éléments de [ICollection](http://msdn.microsoft.com/fr-fr/library/system.collections.icollection(v=VS.80).aspx) à la fin de **ArrayList**. |
| [BinarySearch](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.binarysearch(v=VS.80).aspx) | Surchargé. Utilise un algorithme de recherche binaire pour trouver un élément spécifique dans le **ArrayList** trié ou une partie de celui-ci. |
| [Clear](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.clear(v=VS.80).aspx) | Supprime tous les éléments de l'objet **ArrayList**. |
| [Clone](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.clone(v=VS.80).aspx) | Crée une copie partielle de **ArrayList**. |
| [Contains](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.contains(v=VS.80).aspx) | Détermine si un élément est dans **ArrayList**. |
| [CopyTo](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.copyto(v=VS.80).aspx) | Surchargé. Copie **ArrayList** ou une partie de celui-ci dans un tableau unidimensionnel. |
| [Equals](http://msdn.microsoft.com/fr-fr/library/system.object.equals(v=VS.80).aspx) | Surchargé. Détermine si deux instances de [Object](http://msdn.microsoft.com/fr-fr/library/system.object(v=VS.80).aspx) sont égales. (Hérité de [Object](http://msdn.microsoft.com/fr-fr/library/system.object(v=VS.80).aspx).) |
| [GetType](http://msdn.microsoft.com/fr-fr/library/system.object.gettype(v=VS.80).aspx) | Obtient le [Type](http://msdn.microsoft.com/fr-fr/library/system.type(v=VS.80).aspx) de l'instance en cours. (Hérité de [Object](http://msdn.microsoft.com/fr-fr/library/system.object(v=VS.80).aspx).) |
| [IndexOf](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.indexof(v=VS.80).aspx) | Surchargé. Retourne l'index de base zéro de la première occurrence d'une valeur dans **ArrayList** ou dans une partie de celui-ci. |
| [Insert](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.insert(v=VS.80).aspx) | Insère un élément dans **ArrayList** à l'index spécifié. |
| [InsertRange](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.insertrange(v=VS.80).aspx) | Insère les éléments d'une collection **ArrayList** à l'index spécifié. |
| [LastIndexOf](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.lastindexof(v=VS.80).aspx) | Surchargé. Retourne l'index de la dernière occurrence d'une valeur dans **ArrayList** ou dans une partie de celui-ci. |
| [Remove](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.remove(v=VS.80).aspx) | Supprime la première occurrence d'un objet spécifique de **ArrayList**. |
| [RemoveAt](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.removeat(v=VS.80).aspx) | Supprime l'élément au niveau de l'index spécifié de **ArrayList**. |
| [RemoveRange](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.removerange(v=VS.80).aspx) | Supprime une plage d'éléments de **ArrayList**. |
| [Reverse](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.reverse(v=VS.80).aspx) | Surchargé. Inverse l'ordre des éléments dans **ArrayList** ou dans une partie de celui-ci. |
| [Sort](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.sort(v=VS.80).aspx) | Surchargé. Trie les éléments dans **ArrayList** ou dans une partie de celui-ci. |
| [TrimToSize](http://msdn.microsoft.com/fr-fr/library/system.collections.arraylist.trimtosize(v=VS.80).aspx) | Définit la capacité au nombre réel d'éléments dans **ArrayList**. |