***Tableaux***

Les tableaux sont des collections ordonnées d'éléments de même type de données auxquels on accède en utilisant le nom du tableau conjointement au décalage de l'élément voulu par rapport au début du tableau. Il existe des différences importantes dans la manière avec laquelle les tableaux sont déclarés et utilisés en C# par rapport à Java.

**Tableaux unidimensionnels**

Un tableau unidimensionnel stocke un nombre fixe d'éléments de manière linéaire et ne nécessite qu'une seule valeur d'index pour identifier un élément. En C#, des crochets doivent suivre le type de données dans la déclaration de tableau et ne peuvent pas être placés après le nom de variable comme cela est autorisé en Java. Ainsi, un tableau de type integers est déclaré à l'aide de la syntaxe suivante :

int[] T;

La déclaration suivante n'est pas valide en C#:

//int T[]; //erreur de compilation

Les crochets permettent de spécifier qu'il s'agit d'un tableau. Toutefois à ce niveau, les tableaux sont déclarés, mais non créés. En fait ils n'existent pas concrètement en mémoire, vous venez de définir une entité virtuelle. Pour pouvoir les manipuler il faut donc les construire: vous utilisez le mot clé **new**.

int[] T;

T = new int[5]; // créer un tableau de 5 entiers

Vous accédez ensuite aux éléments dans un tableau unidimensionnel à l'aide d'une syntaxe identique à Java. Les indices de tableau C# sont également de base zéro. Le code suivant permet d'accéder au dernier élément du tableau précédent :

System.Console.WriteLine(T[4]); // Accès à la 5ème valeur

**Initialisation**

Les éléments de tableau C# peuvent être initialisés au moment de la création à l'aide de la même syntaxe qu'en Java :

int[] T;

T = new int[5] {1, 2, 3, 4, 5};

Le nombre d'initialiseurs C# doit correspondre exactement à la taille de tableau. Vous pouvez utiliser cette fonctionnalité pour déclarer et initialiser un tableau C# dans une seule ligne :

int[] T = {1, 2, 3, 4, 5};

Cette syntaxe crée un tableau de taille égale au nombre d'initialiseurs.

**Initialisation dans une boucle de programme**

L'autre manière d'initialiser un tableau en C# consiste à utiliser la boucle **for**. La boucle suivante affecte la valeur zéro à chaque élément d'un tableau :

int[] T = new int[5];

for (int i=0; i<T.Length; i++) // T.Length donne la taille du tableau T

{

T[i] = 0;

}

**La classe System.Array**

Dans le .NET Framework, les tableaux sont implémentés comme des instances de la classe [Array](http://msdn.microsoft.com/fr-fr/library/system.array(v=vs.80).aspx). Cette classe fournit plusieurs méthodes utiles, telles que Sort et [Reverse](http://msdn.microsoft.com/fr-fr/library/d3877932(v=vs.80).aspx).

L'exemple suivant montre combien ces méthodes sont simples d'utilisation. En premier lieu, vous inversez les éléments d'un tableau à l'aide de la méthode **Reverse**, puis, vous les triez avec la méthode **Sort** :

class ArrayMethods

{

static void Main()

{

// Créer un tableau de string de taille 5

string[] Noms = new string[5];

// Lire les noms de 5 employés

System.Console.WriteLine("Entrer les noms de 5 emplyés:");

for (int i = 0; i < Noms.Length; i++)

{

Noms[i] = Console.ReadLine();

}

// Afficher les éléments du tableau saisi

Console.WriteLine("\nle tableau siasi:");

foreach (string nom in Noms)

{

Console.Write("{0} ", nom);

}

// Inverser le tableau

Array.Reverse(Noms);

// Afficher le tableau inversé

Console.WriteLine("\n\nLe tableau dans l’ordre inverse:");

foreach (string nom in Noms)

{

Console.Write("{0} ", nom);

}

// Trier le tableau

Array.Sort(Noms);

// Afficher le tableau trié

Console.WriteLine("\n\nLe tableau trié:");

foreach (string nom in Noms)

{

Console.Write("{0} ", nom);

}

}

}

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Méthodes publiques**](javascript:void(0)) (http://msdn.microsoft.com/fr-fr/library/e23tz9hz(v=VS.80).aspx)

|  |  |  |
| --- | --- | --- |
|  | Nom | Description |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [Clear](http://msdn.microsoft.com/fr-fr/library/system.array.clear(v=VS.80).aspx) | Affecte à une plage d'éléments de **Array** la valeur zéro, **false** ou référence Null (**Nothing** en Visual Basic), selon le type d'élément. |
| Méthode publiquePris en charge par le .NET Compact Framework | [Clone](http://msdn.microsoft.com/fr-fr/library/system.array.clone(v=VS.80).aspx) | Crée une copie partielle de **Array**. |
| Méthode publiqueStatique | [ConvertAll](http://msdn.microsoft.com/fr-fr/library/exc45z53(v=VS.80).aspx) | Convertit un tableau d'un type en un tableau d'un autre type. |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [Copy](http://msdn.microsoft.com/fr-fr/library/system.array.copy(v=VS.80).aspx) | Surchargé. Copie une plage d'éléments d'un **Array** vers un autre **Array** et effectue éventuellement le casting de type et la conversion boxing. |
| Méthode publiquePris en charge par le .NET Compact Framework | [CopyTo](http://msdn.microsoft.com/fr-fr/library/system.array.copyto(v=VS.80).aspx) | Surchargé. Copie tous les éléments du **Array** unidimensionnel en cours dans le **Array** unidimensionnel spécifié. |
| Méthode publiquePris en charge par le .NET Compact Framework | [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).) |
| Méthode publiqueStatique | [Exists](http://msdn.microsoft.com/fr-fr/library/yw84x8be(v=VS.80).aspx) | Détermine si le tableau spécifié contient des éléments qui correspondent aux conditions définies par le prédicat spécifié. |
| Méthode publiqueStatique | [Find](http://msdn.microsoft.com/fr-fr/library/d9hy2xwa(v=VS.80).aspx) | Recherche un élément qui correspond aux conditions définies par le prédicat spécifié et retourne la première occurrence dans le **Array** entier. |
| Méthode publiqueStatique | [FindAll](http://msdn.microsoft.com/fr-fr/library/1kkxfxdd(v=VS.80).aspx) | Récupère tous les éléments qui correspondent aux conditions définies par le prédicat spécifié. |
| Méthode publiqueStatique | [FindIndex](http://msdn.microsoft.com/fr-fr/library/system.array.findindex(v=VS.80).aspx) | Surchargé. |
| Méthode publiqueStatique | [FindLast](http://msdn.microsoft.com/fr-fr/library/2857xz7b(v=VS.80).aspx) | Recherche un élément qui correspond aux conditions définies par le prédicat spécifié et retourne la dernière occurrence dans le **Array** entier. |
| Méthode publiqueStatique | [FindLastIndex](http://msdn.microsoft.com/fr-fr/library/system.array.findlastindex(v=VS.80).aspx) | Surchargé. |
| Méthode publiqueStatique | [ForEach](http://msdn.microsoft.com/fr-fr/library/zecdkyw2(v=VS.80).aspx) | Exécute l'action spécifiée sur chaque élément du tableau spécifié. |
| Méthode publiquePris en charge par le .NET Compact Framework | [GetLength](http://msdn.microsoft.com/fr-fr/library/system.array.getlength(v=VS.80).aspx) | Obtient un entier 32 bits qui représente le nombre d'éléments dans la dimension spécifiée de **Array**. |
| Méthode publique | [GetLongLength](http://msdn.microsoft.com/fr-fr/library/system.array.getlonglength(v=VS.80).aspx) | Obtient un entier 64 bits qui représente le nombre d'éléments dans la dimension spécifiée de **Array**. |
| Méthode publiquePris en charge par le .NET Compact Framework | [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).) |
| Méthode publiquePris en charge par le .NET Compact Framework | [GetValue](http://msdn.microsoft.com/fr-fr/library/system.array.getvalue(v=VS.80).aspx) | Surchargé. Obtient la valeur de l'élément spécifié dans le **Array** en cours. |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [IndexOf](http://msdn.microsoft.com/fr-fr/library/system.array.indexof(v=VS.80).aspx) | Surchargé. Retourne l'index de la première occurrence d'une valeur dans un **Array** unidimensionnel ou dans une partie de **Array**. |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [LastIndexOf](http://msdn.microsoft.com/fr-fr/library/system.array.lastindexof(v=VS.80).aspx) | Surchargé. Retourne l'index de la dernière occurrence d'une valeur dans un **Array** unidimensionnel ou dans une partie de **Array**. |
| Méthode publiqueStatique | [Resize](http://msdn.microsoft.com/fr-fr/library/1ffy6686(v=VS.80).aspx) | Modifie la taille d'un tableau avec la nouvelle taille spécifiée. |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [Reverse](http://msdn.microsoft.com/fr-fr/library/system.array.reverse(v=VS.80).aspx) | Surchargé. Inverse l'ordre des éléments dans un **Array** unidimensionnel ou dans une partie de **Array**. |
| Méthode publiquePris en charge par le .NET Compact Framework | [SetValue](http://msdn.microsoft.com/fr-fr/library/system.array.setvalue(v=VS.80).aspx) | Surchargé. Affecte la valeur spécifiée à l'élément déterminé dans le **Array** en cours. |
| Méthode publiqueStatiquePris en charge par le .NET Compact Framework | [Sort](http://msdn.microsoft.com/fr-fr/library/system.array.sort(v=VS.80).aspx) | Surchargé. Trie les éléments dans les objets **Array** unidimensionnels. |

Exemple :

class Program

{

static void Main(string[] args)

{

Client[] t = new Client[3];

t[0] = new Client("Alami", "ali", 2, 16);

t[1] = new Client("Alami", "fati", 2, 16);

t[2] = new Client("salmi", "ahmed", 4, 17);

// afficher tous les éléments du tableau t

foreach (Client a in t) Console.WriteLine(a.ToString());

Client c = new Client("Alami", "fati", 2, 16);

// chercher la position de la première occurrence de c dans le tableau t

int i = Array.IndexOf(t, c);

Console.WriteLine("la position de c est: " + i);

// chercher la position de la dernière occurrence de c dans le tableau t

i = Array.LastIndexOf(t, c);

Console.WriteLine("la position de c est: " + i);

//Récuppérer la longueur de la dimension 0 du tableau t

Console.WriteLine("longueur=: " + t.GetLength(0));

Client[] T = new Client[3];

// copier 3 éléments du tableau t dans T

Array.Copy(t, T, 3);

//copier t dans T à partir de la position 0

t.CopyTo(T, 0);

// tester si t et T sont egaux

bool res= Array.Equals(t, T);

Console.WriteLine("res= " + res);

// affecte null à la case 2, le dernier parametre (1) désigne le nombre de cases auxquelles on veut affecter null

Array.Clear(T, 2, 1);

// redimensioner le tableau T

Array.Resize(ref T, 2);

foreach (Client a in T) Console.WriteLine(a.ToString());

// recupperer le type

Console.WriteLine("type: " + t.GetType());

Console.Read();

}

}