Annexe : la bibliothèque de classes .NET Framework

1. Présentation du Framework .Net

La bibliothèque de classes .NET Framework est une bibliothèque de classes, interfaces et types valeur inclus dans le Kit de développement Microsoft .NET Framework SDK. Cette bibliothèque, qui permet d'accéder aux fonctionnalités du système, est le fondement des applications, composants et contrôles du .NET Framework.
La bibliothèque de classes .NET Framework fournit les espaces de noms suivants :

System
Contient des classes fondamentales et des classes de base qui définissent les types de données référence et valeur, les événements et gestionnaires d'événements, les interfaces, les attributs et le traitement des exceptions courants. D'autres classes fournissent des services prenant en charge la conversion des types de données, la manipulation des paramètres de méthodes, les opérations mathématiques, l'appel de programmes distants et locaux, la gestion de l'environnement d'application, ainsi que le contrôle des applications managées et non managées.

System.CodeDOM
Contient des classes qui peuvent être utilisées pour représenter les éléments et la structure d'un document de code source. Ces éléments peuvent être utilisés pour créer la structure d'un document de code source qui peut s'afficher comme code source dans un langage pris en charge à l'aide de la fonctionnalité fournie par l'espace de noms System.CodeDom.Compiler.

System.Collections
Contient des interfaces et des classes qui définissent différentes collections d'objets, telles que des listes, des files d'attente, des tableaux de bits, des tables de hachage et des dictionnaires.

System.ComponentModel
Fournit des classes qui sont utilisées pour implémenter le comportement au moment de l'exécution et au moment du design des composants et des contrôles. Cet espace de noms inclut les classes de base et les interfaces pour l'implémentation des attributs et des convertisseurs de type, pour la liaison à des sources de données et pour la licence des composants.

System.Configuration
Contient les types qui fournissent le modèle de programmation destiné à la gestion des données de configuration.

System.Data
Contient des classes qui constituent la majeure partie de l'architecture ADO.NET. L'architecture ADO.NET vous permet de construire des composants qui gèrent efficacement les données provenant de plusieurs sources. Dans un scénario déconnecté (tel qu'Internet), ADO.NET fournit les outils permettant de demander, mettre à jour et rapprocher les données de systèmes à plusieurs couches. L'architecture ADO.NET est

également implémentée dans les applications clientes, telles que Windows Forms ou les pages HTML créées par ASP.NET.

System.Deployment

Contient les classes que vous devez utiliser pour mettre à jour par programme l'application ClickOnce à l'aide de sa version la plus récente.

System.Diagnostics

Fournit des classes qui vous permettent d'interagir avec des processus système, des journaux des événements et des compteurs de performance. Cet espace de noms fournit également des classes qui vous permettent de déboguer votre application et d'effectuer le suivi de l'exécution de votre code. Pour plus d'informations, consultez les classes Trace et Debug.

System.Drawing

Permet d'accéder aux fonctionnalités graphiques de base de GDI+. Des fonctionnalités plus avancées sont offertes dans les espaces de noms System.Drawing.Drawing2D, System.Drawing.Imaging et System.Drawing.Text.

System.Globalization

Contient des classes qui définissent des informations liées à la culture, notamment la langue, le pays ou la région, les calendriers utilisés, les formats des dates, des monnaies et des nombres, ainsi que l'ordre de tri à respecter pour les chaînes. Ces classes sont utiles pour écrire des applications globalisées (internationalisées).

System.IO

Contient des types qui permettent la lecture et l'écriture synchrone et asynchrone de flux de données et de fichiers.

System.Net

Constitue une interface de programmation simple pour un grand nombre des protocoles réseau employés aujourd'hui. Les classes WebRequest et WebResponse constituent la base des protocoles enfichables, qui sont une implémentation de services réseau vous permettant de développer des applications qui utilisent des ressources Internet sans vous soucier des spécificités de chaque protocole.

System.Reflection

Contient des classes et des interfaces qui fournissent une vue managée des types, des méthodes et des champs chargés, avec la possibilité de créer dynamiquement et d'appeler des types.

System.Resources

Contient des classes et des interfaces qui permettent aux développeurs de créer, de stocker et de gérer différentes ressources spécifiques à la culture utilisées dans une application.

System.Runtime

Contient des types avancés qui prennent en charge des espaces de noms divers tels que System, **Runtime** et **Security**.

System.Security

Fournit la structure sous-jacente du système de sécurité .NET Framework, y compris les classes de base pour les autorisations.

System.Text

Contient des classes représentant le codage de caractères ASCII, Unicode, UTF-7 et UTF-8 ; des classes de base abstraites pour convertir des blocs de caractères en direction et en provenance de blocs d'octets ; et une classe d'assistance qui manipule et formate les objets String sans créer des instances intermédiaires de String.

System.Threading

Fournit des classes et des interfaces qui permettent la programmation multithread. En plus des classes permettant la synchronisation des activités des threads et l'accès aux données (Mutex, Monitor, Interlocked, AutoResetEvent, etc.), cet espace de noms comprend une classe ThreadPool qui vous permet d'utiliser un pool de threads fournis par le système et une classe Timer qui exécute les méthodes de rappel sur des threads du pool de threads.

System.Timers

Fournit le composant Timer qui vous permet de déclencher un événement à un intervalle spécifié.

System.Web

Fournit des classes et des interfaces permettant la communication entre le navigateur et le serveur. Cet espace de noms inclut la classe HTTPRequest, qui fournit des informations complètes sur la demande HTTP actuelle, la classe HTTPResponse qui gère la sortie HTTP au client, et la classe HTTPServerUtility qui fournit l'accès aux utilitaires et aux processus côté serveur. **System.Web** inclut également des classes utilisées pour la manipulation des cookies, pour le transfert de fichiers, pour l'obtention d'informations sur les exceptions et pour le contrôle de cache de sortie.

System.Windows.Forms

Contient des classes permettant de créer des applications Windows qui profitent pleinement des fonctionnalités élaborées de l'interface utilisateur disponibles dans le système d'exploitation Microsoft Windows.

System.Xml

Fournit une prise en charge des normes pour le traitement du code XML.

2. L’espace de noms System

L'espace de noms **System** contient des classes fondamentales et des classes de base qui définissent des types de données valeur et référence, des événements et des gestionnaires d'événements, des interfaces, des attributs, ainsi que des exceptions de traitement couramment utilisés.

Classes

Classe	Description
Array	Fournit des méthodes pour la création, la manipulation, la recherche ainsi que le tri des tableaux et sert de classe de base pour tous les tableaux du Common Language Runtime.
BitConverter	Convertit les types de données de base en tableau d'octets et un tableau d'octets en types de données de base.

Buffer	Manipule les tableaux de types primitifs.
Console	Représente les flux d'entrée, de sortie et d'erreur standard pour les applications console. Cette classe ne peut pas être héritée.
Convert	Convertit un type de données de base en un autre type de données de base.
DBNull	Représente une valeur null.
Environment	Fournit des informations concernant l'environnement et la plate-forme en cours ainsi que des moyens pour les manipuler. Cette classe ne peut pas être héritée.
EventArgs	EventArgs est la classe de base des classes contenant des données d'événement.
Exception	Représente les erreurs qui se produisent lors de l'exécution de l'application.
GC	Contrôle le garbage collector (ramasse-miettes) du système, un service qui récupère automatiquement la mémoire inutilisée.
Math	Fournit des constantes et des méthodes statiques pour des fonctions trigonométriques, logarithmiques et d'autres fonctions mathématiques courantes.
Nullable	Prend en charge un type valeur auquel il est possible d'assigner référence Null (**Nothing** en Visual Basic) en tant que type référence. Cette classe ne peut pas être héritée.
Object	Prend en charge toutes les classes de la hiérarchie des classes du .NET Framework et fournit des services de bas niveau à des classes dérivées. Il s'agit de la classe de base fondamentale parmi toutes les classes du .NET Framework. Elle constitue la racine de la hiérarchie des types.
OperatingSystem	Représente des informations relatives à un système d'exploitation, telles que l'identificateur de version et de plate-forme. Cette classe ne peut pas être héritée.
ParamArrayAttribute	Indique que la méthode comportera un nombre variable d'arguments dans son appel. Cette classe ne peut pas être héritée.
Random	Représente un générateur de nombres pseudo-aléatoires. Il s'agit d'un périphérique qui produit une séquence de nombres conformes à certains prérequis statistiques liés à l'aspect aléatoire.
String	Représente du texte sous forme d'une série de caractères Unicode.

TimeZone	Représente un fuseau horaire.
Type	Représente les déclarations de types : types classe, types interface, types tableau, types valeur, types énumération, paramètres de type, définitions de type générique et types génériques construits ouverts ou fermés.
Version	Représente le numéro de version d'un assembly du Common Language Runtime. Cette classe ne peut pas être héritée.

Structures

Structure	Description
ArgIterator	Représente une liste d'arguments de longueur variable, autrement dit les paramètres d'une fonction qui prend un nombre d'arguments variable.
ArraySegment	Délimite une section d'un tableau unidimensionnel.
Boolean	Représente une valeur booléenne.
Byte	Représente un entier non signé 8 bits.
Char	Représente un caractère Unicode.
ConsoleKeyInfo	Décrit la touche de console qui a été enfoncée, y compris le caractère représenté par la touche de console et l'état des touches de modification MAJ, ALT et CTRL.
DateTime	Représente un instant, généralement exprimé sous la forme d'une date ou d'une heure.
Decimal	Représente un nombre décimal.
Double	Représente un nombre à virgule flottante double précision.
Enum	Fournit la classe de base pour les énumérations.
Guid	Représente un GUID (identificateur global unique).
Int16	Représente un entier signé 16 bits.
Int32	Représente un entier signé 32 bits.
Int64	Représente un entier signé 64 bits.
IntPtr	Type spécifique à la plate-forme, utilisé pour représenter un pointeur.
Nullable	Représente un objet dont le type sous-jacent est un type valeur qui peut également être assigné référence Null (**Nothing** en Visual Basic) comme un type référence.

SByte	Représente un entier signé 8 bits.
Single	Représente un nombre à virgule flottante simple précision.
TimeSpan	Représente un intervalle de temps.
UInt16	Représente un entier non signé 16 bits.
UInt32	Représente un entier non signé 32 bits.
UInt64	Représente un entier non signé 64 bits.
UIntPtr	Type spécifique à la plate-forme, utilisé pour représenter un pointeur.
Void	Spécifie un type de valeur de retour pour une méthode qui ne retourne pas de valeur.

3. Les conversions de types

Méthodes de la classe statique System.Convert

Nom	Description
ChangeType	Surchargé. Retourne un Object possédant un type spécifié et dont la valeur équivaut à un objet spécifié.
Equals	Surchargé. Détermine si deux instances de **Object** sont égales. (Hérité de Object.)
FromBase64CharArray	Convertit un sous-ensemble d'un tableau de caractères Unicode (qui code les données binaires en chiffres en base 64) en un tableau équivalent d'entiers non signés 8 bits. Les paramètres spécifient le sous-ensemble dans le tableau d'entrée et le nombre d'éléments à convertir.
FromBase64String	Convertit le String spécifié (qui code les données binaires en chiffres en base 64) en un tableau équivalent d'entiers non signés 8 bits.
GetHashCode	Sert de fonction de hachage pour un type particulier. GetHashCode est aproprié à une utilisation dans des algorithmes de hachage et des structures de données telles qu'une table de hachage. (Hérité de Object.)
GetType	Obtient le Type de l'instance en cours. (Hérité de Object.)
GetTypeCode	Retourne le TypeCode de l'objet spécifié.
IsDBNull	Retourne une indication précisant si l'objet spécifié est de type DBNull.

ReferenceEquals	Détermine si les instances de **Object** spécifiées sont identiques. (Hérité de Object.)
ToBase64CharArray	Surchargé. Convertit un sous-ensemble d'un tableau d'entiers non signés 8 bits en un sous-ensemble équivalent d'un tableau de caractères Unicode codé à l'aide de chiffres en base 64.
ToBase64String	Surchargé. Convertit la valeur d'un tableau d'entiers non signés 8 bits en sa représentation **String** équivalente codée à l'aide de chiffres en base 64.
ToBoolean	Surchargé. Convertit une valeur spécifiée en une valeur booléenne équivalente.
ToByte	Surchargé. Convertit une valeur spécifiée en un entier non signé 8 bits.
ToChar	Surchargé. Convertit une valeur spécifiée en caractères Unicode.
ToDateTime	Surchargé. Convertit une valeur spécifiée en DateTime.
ToDecimal	Surchargé. Convertit une valeur spécifiée en un nombre Decimal.
ToDouble	Surchargé. Convertit une valeur spécifiée en un nombre à virgule flottante double précision.
ToInt16	Surchargé. Convertit une valeur spécifiée en un entier signé 16 bits.
ToInt32	Surchargé. Convertit une valeur spécifiée en un entier signé 32 bits.
ToInt64	Surchargé. Convertit une valeur spécifiée en un entier signé 64 bits.
ToSByte	Surchargé. Convertit une valeur spécifiée en un entier signé 8 bits.
ToSingle	Surchargé. Convertit une valeur spécifiée en un nombre à virgule flottante simple précision.
ToString	Surchargé. Convertit la valeur spécifiée en sa représentation **String** équivalente.
ToUInt16	Surchargé. Convertit une valeur spécifiée en un entier non signé 16 bits.
ToUInt32	Surchargé. Convertit une valeur spécifiée en un entier non signé 32 bits.

ToUInt64	Surchargé. Convertit une valeur spécifiée en un entier non signé 64 bits.

4. Les fonctions mathématiques

La classe System.Math Fournit des constantes et des méthodes statiques pour des fonctions trigonométriques, logarithmiques et d'autres fonctions mathématiques courantes.

Champs publics

Nom	Description
E	Représente la base de logarithme naturelle spécifiée par la constante **e**.
PI	Représente le rapport de la circonférence d'un cercle à son diamètre, spécifié par la constante π.

Méthodes publiques

Nom	Description
Abs (x)	Surchargé. Retourne la valeur absolue d'un nombre spécifié.
Acos (x)	Retourne l'angle dont le cosinus est le nombre spécifié.
Asin (x)	Retourne l'angle dont le sinus est le nombre spécifié.
Atan (x)	Retourne l'angle dont la tangente est le nombre spécifié.
Atan2 (x,y)	Retourne l'angle dont la tangente est le quotient de deux nombres spécifiés.
BigMul (x,y)	Génère le produit intégral de deux nombres 32 bits.
Ceiling (x)	Surchargé. Retourne le plus petit entier supérieur ou égal au nombre spécifié.
Cos (x)	Retourne le cosinus de l'angle spécifié.
Cosh (x)	Retourne le cosinus hyperbolique de l'angle spécifié.
DivRem (x,y,reste)	Surchargé. Retourne le quotient de deux nombres et retourne également le reste dans un paramètre de sortie.
Equals	Surchargé. Détermine si deux instances de Object sont égales. (Hérité de Object.)
Exp (x)	Retourne **e** élevé à la puissance spécifiée.
Floor (x)	Surchargé. Retourne le plus grand entier inférieur ou égal au nombre spécifié.

GetHashCode (x)	Sert de fonction de hachage pour un type particulier. GetHashCode est aproprié à une utilisation dans des algorithmes de hachage et des structures de données telles qu'une table de hachage. (Hérité de Object.)
GetType (x)	Obtient le Type de l'instance en cours. (Hérité de Object.)
IEEERemainder (x,y)	Retourne le reste de la division d'un nombre spécifié par un autre.
Log (x)	Surchargé. Retourne le logarithme d'un nombre spécifié.
Log10 (x)	Retourne le logarithme de base 10 d'un nombre spécifié.
Max (x,y)	Surchargé. Retourne le plus grand de deux nombres spécifiés.
Min (x,y)	Surchargé. Retourne le plus petit de deux nombres.
Pow (x,y)	Retourne un nombre spécifié x élevé à la puissance spécifiée y.
ReferenceEquals	Détermine si les instances de **Object** spécifiées sont identiques. (Hérité de Object.)
Round (x)	Surchargé. Arrondit une valeur à l'entier le plus proche ou au nombre spécifié de décimales.
Sign (x)	Surchargé. Retourne une valeur indiquant le signe d'un nombre.
Sin (x)	Retourne le sinus de l'angle spécifié.
Sinh (x)	Retourne le sinus hyperbolique de l'angle spécifié.
Sqrt (x)	Retourne la racine carrée d'un nombre spécifié.
Tan (x)	Retourne la tangente de l'angle spécifié.
Tanh (x)	Retourne la tangente hyperbolique de l'angle spécifié.
ToString	Retourne un String qui représente le **Object** en cours. (Hérité de Object.)
Truncate (x)	Surchargé. Calcule la partie entière d'un nombre.

5. Les chaînes de caractères

La classe **System.String** est identique au type simple **string**. Elle présente de nombreuses propriétés et méthodes. En voici quelques-unes :

public int Length { get; }	nombre de caractères de la chaîne
public bool EndsWith(string value)	rend vrai si la chaîne se termine par *value*

public bool StartsWith(string value)	rend vrai si la chaîne commence par *value*
public virtual bool Equals(object obj)	rend vrai si la chaînes est égale à *obj* – équivalent chaîne==obj
public int IndexOf(string value, int startIndex)	rend la première position dans la chaîne de la chaîne *value* - la recherche commence à partir du caractère n° *startIndex*
public int IndexOf(char value, int startIndex)	idem mais pour le caractère *value*
public string Insert(int startIndex, string value)	insère la chaîne *value* dans chaîne en position *startIndex*
public static string Join(string separator, string[] value)	méthode de classe - rend une chaîne de caractères, résultat de la concaténation des valeurs du tableau *value* avec le séparateur *separator*
public int LastIndexOf(char value, int startIndex, int count)	idem *indexOf* mais rend la dernière position au lieu de la première
public int LastIndexOf(string value, int startIndex, int count)	idem *indexOf* mais rend la dernière position au lieu de la première
public string Replace(char oldChar, char newChar)	rend une chaîne copie de la chaîne courante où le caractère *oldChar* a été remplacé par le caractère *newChar*
public string[] Split(char[] separator)	la chaîne est vue comme une suite de champs séparés par les caractères présents dans le tableau *separator*. Le résultat est le tableau de ces champs
public string Substring(int startIndex, int length)	sous-chaîne de la chaîne courante commençant à la position *startIndex* et ayant *length* caractères
public string ToLower()	rend la chaîne courante en minuscules
public string ToUpper()	rend la chaîne courante en majuscules
public string Trim()	rend la chaîne courante débarrassée de ses espaces de début et fin

On notera un point important : lorsqu'une méthode rend une chaîne de caractères, celle-ci est une **chaîne différente** de la chaîne sur laquelle a été appliquée la méthode. Ainsi *S1.Trim()* rend une chaîne *S2*, et *S1* et *S2* sont deux chaînes différentes.
Une chaîne C peut être considérée comme un tableau de caractères. Ainsi **C[i]** est le caractère i de C et **C.Length** est le nombre de caractères de C.

6. La classe Object

Prend en charge toutes les classes de la hiérarchie des classes du .NET Framework et fournit des services de bas niveau à des classes dérivées. Il s'agit de la classe de base fondamentale parmi toutes les classes du .NET Framework. Elle constitue la racine de la hiérarchie des types.

Méthodes publiques

Nom	Description
Equals	Surchargé. Détermine si deux instances de **Object** sont égales.

GetHashCode	Sert de fonction de hachage pour un type particulier. GetHashCode est approprié à une utilisation dans des algorithmes de hachage et des structures de données telles qu'une table de hachage.
GetType	Obtient le Type de l'instance en cours.
ReferenceEquals	Détermine si les instances de **Object** spécifiées sont identiques.
ToString	Retourne un String qui représente le **Object** en cours.

7. La classe ArrayList (System.Collections)

Implémente l'interface IList à l'aide d'un tableau dont la taille augmente dynamiquement selon les besoins.

Propriétés publiques

Nom	Description
Capacity	Obtient ou définit le nombre d'éléments que **ArrayList** peut contenir.
Count	Obtient le nombre d'éléments réellement contenus dans **ArrayList**.
IsFixedSize	Obtient une valeur indiquant si **ArrayList** est de taille fixe.
IsReadOnly	Obtient une valeur indiquant si **ArrayList** est en lecture seule.
IsSynchronized	Obtient une valeur indiquant si l'accès à **ArrayList** est synchronisé (thread-safe).
Item	Obtient ou définit l'élément situé à l'index spécifié.
SyncRoot	Obtient un objet qui peut être utilisé pour synchroniser l'accès au **ArrayList**.

8. Méthodes publiques

Nom	Description
Adapter	Crée un wrapper **ArrayList** pour un **IList** spécifique.
Add	Ajoute un objet à la fin de **ArrayList**.
AddRange	Ajoute les éléments de ICollection à la fin de **ArrayList**.
BinarySearch	Surchargé. Utilise un algorithme de recherche binaire pour trouver un élément spécifique dans le **ArrayList** trié ou une partie de celui-ci.
Clear	Supprime tous les éléments de l'objet **ArrayList**.
Clone	Crée une copie partielle de **ArrayList**.
Contains	Détermine si un élément est dans **ArrayList**.
CopyTo	Surchargé. Copie **ArrayList** ou une partie de celui-ci dans un tableau unidimensionnel.
Equals	Surchargé. Détermine si deux instances de Object sont égales. (Hérité de Object.)
FixedSize	Surchargé. Retourne un wrapper de liste de taille fixe, où les éléments peuvent être modifiés, mais ni ajoutés ni supprimés.
GetEnumerator	Surchargé. Retourne un énumérateur qui parcourt **ArrayList**.
GetHashCode	Sert de fonction de hachage pour un type particulier. GetHashCode est approprié à une utilisation dans des algorithmes de hachage et des structures de données telles qu'une table de hachage. (Hérité de Object.)
GetRange	Retourne **ArrayList** qui représente un sous-ensemble des éléments dans le **ArrayList** source.
GetType	Obtient le Type de l'instance en cours. (Hérité de Object.)
IndexOf	Surchargé. Retourne l'index de base zéro de la première occurrence d'une valeur dans **ArrayList** ou dans une partie de celui-ci.
Insert	Insère un élément dans **ArrayList** à l'index spécifié.
InsertRange	Insère les éléments d'une collection **ArrayList** à l'index spécifié.
LastIndexOf	Surchargé. Retourne l'index de base zéro de la dernière occurrence d'une valeur dans **ArrayList** ou dans une partie de celui-ci.
ReadOnly	Surchargé. Retourne un wrapper de liste en lecture seule.
ReferenceEquals	Détermine si les instances de **Object** spécifiées sont identiques. (Hérité de Object.)
Remove	Supprime la première occurrence d'un objet spécifique de **ArrayList**.
RemoveAt	Supprime l'élément au niveau de l'index spécifié de **ArrayList**.

RemoveRange	Supprime une plage d'éléments de **ArrayList**.
Repeat	Retourne **ArrayList** dont les éléments sont des copies de la valeur spécifiée.
Reverse	Surchargé. Inverse l'ordre des éléments dans **ArrayList** ou dans une partie de celui-ci.
SetRange	Copie les éléments d'une collection sur une plage d'éléments dans **ArrayList**.
Sort	Surchargé. Trie les éléments dans **ArrayList** ou dans une partie de celui-ci.
Synchronized	Surchargé. Retourne un wrapper de liste qui est synchronisé (thread-safe).
ToArray	Surchargé. Copie les éléments de **ArrayList** vers un nouveau tableau.
ToString	Retourne un String qui représente le **Object** en cours. (Hérité de Object.)
TrimToSize	Définit la capacité au nombre réel d'éléments dans **ArrayList**.

9. La classe List (System.Collections.Generic)

Représente une liste fortement typée d'objets accessibles par index. Fournit des méthodes de recherche, de tri et de manipulation de listes.

Propriétés publiques

Nom	Description
Capacity	Obtient ou définit le nombre total des éléments que la structure de données interne peut contenir sans redimensionnement.
Count	Obtient le nombre d'éléments réellement contenus dans **List**.
Item	Obtient ou définit l'élément situé à l'index spécifié.

Méthodes publiques

Nom	Description
Add	Ajoute un objet à la fin de **List**.
AddRange	Ajoute les éléments de la collection spécifiée à la fin du **List**.
AsReadOnly	Retourne un wrapper IList en lecture seule pour la collection actuelle.
BinarySearch	Surchargé. Utilise un algorithme de recherche binaire pour trouver un

	élément spécifique dans le **List** trié ou une partie de celui-ci.
Clear	Supprime tous les éléments de **List**.
Contains	Détermine si un élément est dans **List**.
ConvertAll	Convertit les éléments du **List** actuel en un autre type et retourne une liste qui contient les éléments convertis.
CopyTo	Surchargé. Copie **List** ou une partie de celui-ci dans un tableau.
Equals	Surchargé. Détermine si deux instances de Object sont égales. (Hérité de Object.)
Exists	Détermine si le **List** contient des éléments qui correspondent aux conditions définies par le prédicat spécifié.
Find	Recherche un élément qui correspond aux conditions définies par le prédicat spécifié et retourne la première occurrence dans le **List** entier.
FindAll	Récupère tous les éléments qui correspondent aux conditions définies par le prédicat spécifié.
FindIndex	Surchargé. Recherches un élément qui correspond aux conditions définies par un prédicat spécifié et retourne l'index de base zéro de la première occurrence dans le **List** ou une partie.
FindLast	Recherche un élément qui correspond aux conditions définies par le prédicat spécifié et retourne la dernière occurrence dans le **List** entier.
FindLastIndex	Surchargé. Recherches un élément qui correspond aux conditions définies par un prédicat spécifié et retourne l'index de base zéro de la dernière occurrence dans le **List** ou une partie de celui-ci.
ForEach	Exécute l'action spécifiée sur chaque élément du **List**.
GetEnumerator	Retourne un énumérateur qui parcourt **List**.
GetHashCode	Sert de fonction de hachage pour un type particulier. GetHashCode est aproprié à une utilisation dans des algorithmes de hachage et des structures de données telles qu'une table de hachage. (Hérité de Object.)
GetRange	Crée une copie superficielle d'une plage d'éléments dans la source **List**.
GetType	Obtient le Type de l'instance en cours. (Hérité de Object.)
IndexOf	Surchargé. Retourne l'index de base zéro de la première occurrence d'une valeur dans **List** ou dans une partie de celui-ci.
Insert	Insère un élément dans **List** à l'index spécifié.
InsertRange	Insère les éléments d'une collection **List** à l'index spécifié.
LastIndexOf	Surchargé. Retourne l'index de base zéro de la dernière occurrence d'une valeur dans **List** ou dans une partie de celui-ci.

ReferenceEquals	Détermine si les instances de **Object** spécifiées sont identiques. (Hérité de Object.)
Remove	Supprime la première occurrence d'un objet spécifique de **List**.
RemoveAll	Supprime tous les éléments qui correspondent aux conditions définies par le prédicat spécifié.
RemoveAt	Supprime l'élément au niveau de l'index spécifié de **List**.
RemoveRange	Supprime une plage d'éléments de **List**.
Reverse	Surchargé. Inverse l'ordre des éléments dans **List** ou dans une partie de celui-ci.
Sort	Surchargé. Trie les éléments dans **List** ou dans une partie de celui-ci.
ToArray	Copie les éléments de **List** vers un nouveau tableau.
ToString	Retourne un String qui représente le **Object** en cours. (Hérité de Object.)
TrimExcess	Affecte à la capacité le nombre réel d'éléments dans **List**, si ce nombre est inférieur à une valeur de seuil.
TrueForAll	Détermine si chaque élément de **List** correspond aux conditions définies par le prédicat spécifié.

Exemple :

```
using System;
using System.Collections.Generic;

public class Example
{
    public static void Main()
    {
        List<string> dinosaurs = new List<string>();

        Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);

        dinosaurs.Add("Tyrannosaurus");
        dinosaurs.Add("Amargasaurus");
        dinosaurs.Add("Mamenchisaurus");
        dinosaurs.Add("Deinonychus");
        dinosaurs.Add("Compsognathus");

        Console.WriteLine();
        foreach(string dinosaur in dinosaurs)
        {
            Console.WriteLine(dinosaur);
        }

        Console.WriteLine("\nCapacity: {0}", dinosaurs.Capacity);
        Console.WriteLine("Count: {0}", dinosaurs.Count);

        Console.WriteLine("\nContains(\"Deinonychus\"): {0}",
            dinosaurs.Contains("Deinonychus"));
```

```
            Console.WriteLine("\nInsert(2, \"Compsognathus\")");
            dinosaurs.Insert(2, "Compsognathus");

            Console.WriteLine();
            foreach(string dinosaur in dinosaurs)
            {
                Console.WriteLine(dinosaur);
            }

            Console.WriteLine("\ndinosaurs[3]: {0}", dinosaurs[3]);

            Console.WriteLine("\nRemove(\"Compsognathus\")");
            dinosaurs.Remove("Compsognathus");

            Console.WriteLine();
            foreach(string dinosaur in dinosaurs)
            {
                Console.WriteLine(dinosaur);
            }

            dinosaurs.TrimExcess();
            Console.WriteLine("\nTrimExcess()");
            Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
            Console.WriteLine("Count: {0}", dinosaurs.Count);

            dinosaurs.Clear();
            Console.WriteLine("\nClear()");
            Console.WriteLine("Capacity: {0}", dinosaurs.Capacity);
            Console.WriteLine("Count: {0}", dinosaurs.Count);
        }
}
```

10. La classe Random

La classe System.Random permet de générer des nombres aléatoires.

- La méthode *Next*( ) retourne un Integer positif entre 0 et 2 147 483 647
- La méthode *NextDouble*( ) retourne un Double entre 0 et 1.
- La méthode *NextBytes*( ) retourne un Byte (octet)

On peut surcharger ces méthodes pour définir des bornes.

Exemple:

J'instancie un objet à partir de la classe.

Random Al = new Random() ;

L'objet Al est initialisé avec une valeur probablement liée au temps, à l'horloge interne, aussi l'initialisation est 'aléatoire'.

- Pour obtenir un nombre (un double) entre 0 et 1 (toujours inférieur à 1), j'écris:
  MonNombrealeatoire=Al.NextDouble() ;

Ensuite chaque *NextDouble* génère le nombre aléatoire suivant (à partir d'une formule).

Noter bien que dans ce qui précède, si on fait plusieurs fois *Al= New Random()* , le nombre obtenu par *NextDouble* n'est jamais le même.

Par contre si on fait:

```
Random Al = New Random(1);
MonNombrealeatoire=Al.NextDouble();
MonNombrealeatoire=Al.NextDouble() ;
```

On obtient toujours:
'0.248668584157093'
'0.110743977181029'

On obtient donc la même série car on a imposé avec *Random(1)* une valeur de départ qui est fonction de (1) et non du temps.

- Pour obtenir un nombre aléatoire entre 0 et 10, on utilise *Next*:

```
Random Al= New Random( );
MonNombrealeatoire=Al.Next(10);
```

- Pour obtenir un nombre aléatoire entre 5 et 10 (mais < à 10), on utilise Next:

```
Random Al =New Random( );
MonNombrealeatoire=Al.Next(5,10) ;
```

11. Heure et date

On peut récupérer la valeur de l'heure et de la date grâce à la structure DateTime définie dans l'espace de noms System.

Exemple :

DateTime.Now.ToShortDateString(); : Retourne la date sous forme d'un String
DateTime.Now.ToLongTimeString();: Retourne l'heure sous forme d'un String
…
DateTime présente plusieurs fonctions de manipulation des dates et du temps.

12. L'opérateur typeof

L'opérateur typeof est utilisé pour obtenir un objet System.Type pour un type.

```
typeof-expression:
        typeof ( type )
        typeof ( void )
```

L'exemple suivant

```
using System;
class Test
{
        static void Main()
 {
                Type[] t = {

                        typeof(int),
```

```
                    typeof(System.Int32),
                    typeof(string),
                    typeof(double[]),
                    typeof(void)
                };
                for (int i = 0; i < t.Length; i++) {

                    Console.WriteLine(t[i].FullName);
                }
            }
        }
```

donne le résultat suivant :

```
System.Int32
System.Int32
System.String
System.Double[]
System.Void
```

13. La fonction MessageBox.Show

Description

Affiche un message dans une boîte de dialogue, attend que l'utilisateur clique sur un bouton et retourne une valeur indiquant le bouton choisi par l'utilisateur.

Syntaxe

returnValue = **MessageBox.Show**(*text*, *caption*, *buttons*, *icon*, *defaultButton*, *options*)

élément	Description
Text AS String	Le contenu du message
Caption As String	Titre du message
Buttons	Définit les boutons qui apparaîtront dans un message
Icon	Définit l'icône du message
DefaultButton	Bouton par défaut
Options	Options d'affichage du message

Buttons :

On choisit un des membres de la collection ***MessageBoxButtons***

Member name	Description
AbortRetryIgnore	The message box contains Abort, Retry, and Ignore buttons.
OK	The message box contains an OK button.

OKCancel	The message box contains OK and Cancel buttons.
RetryCancel	The message box contains Retry and Cancel buttons.
YesNo	The message box contains Yes and No buttons.
YesNoCancel	The message box contains Yes, No, and Cancel buttons.

Icon :
On choisit un des membres de la collection ***MessageBoxIcon***

Member name	Description
Asterisk	The message box contains a symbol consisting of a lowercase letter i in a circle.
Error	The message box contains a symbol consisting of white X in a circle with a red background.
Exclamation	The message box contains a symbol consisting of an exclamation point in a triangle with a yellow background.
Hand	The message box contains a symbol consisting of a white X in a circle with a red background.
Information	The message box contains a symbol consisting of a lowercase letter i in a circle.
None	The message box contain no symbols.
Question	The message box contains a symbol consisting of a question mark in a circle.
Stop	The message box contains a symbol consisting of white X in a circle with a red background.
Warning	The message box contains a symbol consisting of an exclamation point in a triangle with a yellow background.

DefaultButton :
On choisit un des membres de la collection ***MessageBoxDefaultButton***

Member name	Description
Button1	The first button on the message box is the default button.
Button2	The second button on the message box is the default button.
Button3	The third button on the message box is the default button.

Options :
On choisit un des membres de la collection ***MessageBoxOptions***

Member name	Description

DefaultDesktopOnly	The message box is displayed on the active desktop. This constant is the same as **ServiceNotification** except that the system displays the message box only on the default desktop of the interactive window station.
RightAlign	The message box text is right-aligned.
RtlReading	Specifies that the message box text is displayed with right to left reading order.
ServiceNotification	The message box is displayed on the active desktop. The caller is a service notifying the user of an event. The function displays a message box on the current active desktop, even if there is no user logged on to the computer.

Valeurs retournées :
On choisit un des membres de la collection ***DialogResult***

Member name	Description
Abort	The dialog box return value is **Abort** (usually sent from a button labeled Abort).
Cancel	The dialog box return value is **Cancel** (usually sent from a button labeled Cancel).
Ignore	The dialog box return value is **Ignore** (usually sent from a button labeled Ignore).
No	The dialog box return value is **No** (usually sent from a button labeled No).
None	**Nothing** is returned from the dialog box. This means that the modal dialog continues running.
OK	The dialog box return value is **OK** (usually sent from a button labeled OK).
Retry	The dialog box return value is **Retry** (usually sent from a button labeled Retry).
Yes	The dialog box return value is **Yes** (usually sent from a button labeled Yes).

Par exemple :
DialogResult.OK
MessageBoxButtons.YesNo

14. Gestion des formulaires

Pour gérer un formulaire, il est possible d'appeler les méthodes suivantes :

Méthode	Rôle	Exemple

Show	Afficher une forme non modale	Form1.Show
ShowDialog	Afficher une forme modale	Form2.ShowDialog
Hide	Masquer une forme	Form1.Hide
Close	Fermer une forme	Form1.Close

15. Les évènements

14.1 La notion de Focus

Le focus, dans une application Windows, désigne le curseur, au sens le plus général du terme. C'est lorsqu'un contrôle possède le "focus" qu'il devient concerné par la frappe d'une touche au clavier (la touche **Entrée** produisant l'enfoncement d'un bouton, par exemple). Selon les contrôles, le focus se matérialise à l'écran par un curseur clignotant (dans une **Textbox**), ou par un liseré sombre (sur un **Button**).

Du point de vue de l'utilisateur, il y a deux moyens de déplacer le focus :

- en cliquant directement avec la souris sur le contrôle désiré
- en appuyant sur la **touche de tabulation**, qui fait circuler le focus d'un contrôle à l'autre.

L'ordre de passage du focus est régi par la propriété **TabIndex** de chaque contrôle : le contrôle qui reçoit le focus par défaut, au lancement de la Form, est celui dont le **TabIndex** vaut zéro.

Voyons maintenant le point de vue du programmeur. On peut tout aussi bien placer d'autorité le focus sur un contrôle par du code, en lui appliquant la méthode... **Focus**.
Mais le code permet également de détecter l'arrivée du focus sur un contrôle, ou son départ. Il suffit pour cela de gérer respectivement les événements **Enter** et **Leave**, eux aussi disponibles pour la quasi-totalité des contrôles.

14.2 Les événements clavier

Keypress : cet événement détecte le fait qu'un **caractère** a été frappé au clavier.
Keydown et **Keyup** : ces deux événements, qui fonctionnent de pair, se déclenchent lorsqu'une touche du clavier est enfoncée (**Keydown**) ou relâchée (**Keyup**). La caractéristique de ces deux événements est qu'ils détectent **l'état physique** du clavier.

Remarque:

Les touches ne produisant pas de caractères, telles les touches de fonction, ou les touches de direction, ne génèrent pas l'événement **Keypress**, mais génèrent les événements **KeyDown** et **KeyUp**

Lors d'un **KeyPress**, nous trouvons pour le paramètre en entrée **e** la propriété caractère **KeyChar**, contenant le caractère généré par la touche pressée.

14. 3 Événements Souris

- **MouseHover** : qui détecte le passage de la souris sur un contrôle
- **MouseEnter** : qui détecte l'arrivée de la souris sur un contrôle

- **MouseLeave** : qui détecte la sortie de la souris d'un contrôle
- **MouseDown** : événement produit par le fait qu'un bouton de la souris vient d'être enfoncé au-dessus d'un contrôle
- **MouseUp** : événement produit par le fait qu'un bouton de la souris vient d'être relâché au-dessus d'un contrôle **MouseMove** : événement produit par le déplacement de la souris au-dessus d'un contrôle

Ces trois événements génèrent donc un objet **e** comportant plusieurs propriétés, dont les plus intéressantes sont :

- **Button** : qui indique quel est le bouton de la souris à l'origine de l'événement, parmi la liste suivante : **Left**, **Right**, **Middle**, **None**.
- **X** et **Y** : qui désignent les coordonnées de la souris **par rapport au contrôle qui reçoit l'événement** (et non par rapport à la Form, attention, piège à... étudiants, et aux autres aussi)

16. Le code des touches

Jeu de caractères

Code	Touche	Code	Touche	Code	Touche	Code	Touche
0	&127;	**32**	[space]	**64**	@	**96**	`
1	&127;	**33**	!	**65**	A	**97**	a
2	&127;	**34**	"	**66**	B	**98**	b
3	&127;	**35**	#	**67**	C	**99**	c
4	&127;	**36**	$	**68**	D	**100**	d
5	&127;	**37**	%	**69**	E	**101**	e
6	&127;	**38**	&	**70**	F	**102**	f
7	&127;	**39**	'	**71**	G	**103**	g
8	Backspace	**40**	(	**72**	H	**104**	h
9	Tabulation	**41**	)	**73**	I	**105**	i
10	Saut de ligne	**42**	*	**74**	J	**106**	j
11	&127;	**43**	+	**75**	K	**107**	k
12	&127;	**44**	,	**76**	L	**108**	l
13	Enter	**45**	-	**77**	M	**109**	m
14	&127;	**46**	.	**78**	N	**110**	n
15	&127;	**47**	/	**79**	O	**111**	o

16	&127;	**48**	0	**80**	P	**112**	p
17	&127;	**49**	1	**81**	Q	**113**	q
18	&127;	**50**	2	**82**	R	**114**	r
19	&127;	**51**	3	**83**	S	**115**	s
20	&127;	**52**	4	**84**	T	**116**	t
21	&127;	**53**	5	**85**	U	**117**	u
22	&127;	**54**	6	**86**	V	**118**	v
23	&127;	**55**	7	**87**	W	**119**	w
24	&127;	**56**	8	**88**	X	**120**	x
25	&127;	**57**	9	**89**	Y	**121**	y
26	&127;	**58**	:	**90**	Z	**122**	z
27	&127;	**59**	;	**91**	[	**123**	{
28	&127;	**60**	<	**92**	\	**124**	|
29	&127;	**61**	=	**93**	]	**125**	}
30	&127;	**62**	>	**94**	^	**126**	~
31	&127;	**63**	?	**95**	_	**127**	&127;

Codes de touches:

Pour récupérer les codes des touches du clavier, il faut utiliser les membres de la collection Keys. (Par exemple : Keys.F1 pour la touche F1)

Member name	Description
A	The A key.
Add	The add key.
Alt	The ALT modifier key.
B	The B key.
Back	The BACKSPACE key.
BrowserBack	The browser back key (Windows 2000 or later).
BrowserFavorites	The browser favorites key (Windows 2000 or later).
BrowserForward	The browser forward key (Windows 2000 or later).

BrowserHome	The browser home key (Windows 2000 or later).
BrowserRefresh	The browser refresh key (Windows 2000 or later).
BrowserSearch	The browser search key (Windows 2000 or later).
BrowserStop	The browser stop key (Windows 2000 or later).
C	The C key.
Cancel	The CANCEL key.
Capital	The CAPS LOCK key.
CapsLock	The CAPS LOCK key.
Clear	The CLEAR key.
Control	The CTRL modifier key.
ControlKey	The CTRL key.
D	The D key.
D0	The 0 key.
D1	The 1 key.
D2	The 2 key.
D3	The 3 key.
D4	The 4 key.
D5	The 5 key.
D6	The 6 key.
D7	The 7 key.
D8	The 8 key.
D9	The 9 key.
Decimal	The decimal key.
Delete	The DEL key.
Divide	The divide key.
Down	The DOWN ARROW key.
E	The E key.
End	The END key.
Enter	The ENTER key.

Escape	The ESC key.
Execute	The EXECUTE key.
F	The F key.
F1	The F1 key.
F10	The F10 key.
F11	The F11 key.
F12	The F12 key.
F13	The F13 key.
F14	The F14 key.
F15	The F15 key.
F16	The F16 key.
F17	The F17 key.
F18	The F18 key.
F19	The F19 key.
F2	The F2 key.
F20	The F20 key.
F21	The F21 key.
F22	The F22 key.
F23	The F23 key.
F24	The F24 key.
F3	The F3 key.
F4	The F4 key.
F5	The F5 key.
F6	The F6 key.
F7	The F7 key.
F8	The F8 key.
F9	The F9 key.
G	The G key.
H	The H key.

Help	The HELP key.
Home	The HOME key.
I	The I key.
Insert	The INS key.
J	The J key.
K	The K key.
L	The L key.
LaunchMail	The launch mail key (Windows 2000 or later).
LButton	The left mouse button.
LControlKey	The left CTRL key.
Left	The LEFT ARROW key.
LMenu	The left ALT key.
LShiftKey	The left SHIFT key.
LWin	The left Windows logo key (Microsoft Natural Keyboard).
M	The M key.
MButton	The middle mouse button (three-button mouse).
MediaNextTrack	The media next track key (Windows 2000 or later).
MediaPlayPause	The media play pause key (Windows 2000 or later).
MediaPreviousTrack	The media previous track key (Windows 2000 or later).
MediaStop	The media Stop key (Windows 2000 or later).
Menu	The ALT key.
Modifiers	The bitmask to extract modifiers from a key value.
Multiply	The multiply key.
N	The N key.
Next	The PAGE DOWN key.
NoName	A constant reserved for future use.
None	No key pressed.
NumLock	The NUM LOCK key.
NumPad0	The 0 key on the numeric keypad.

NumPad1	The 1 key on the numeric keypad.
NumPad2	The 2 key on the numeric keypad.
NumPad3	The 3 key on the numeric keypad.
NumPad4	The 4 key on the numeric keypad.
NumPad5	The 5 key on the numeric keypad.
NumPad6	The 6 key on the numeric keypad.
NumPad7	The 7 key on the numeric keypad.
NumPad8	The 8 key on the numeric keypad.
NumPad9	The 9 key on the numeric keypad.
O	The O key.
P	The P key.
Pa1	The PA1 key.
PageDown	The PAGE DOWN key.
PageUp	The PAGE UP key.
Pause	The PAUSE key.
Play	The PLAY key.
Print	The PRINT key.
PrintScreen	The PRINT SCREEN key.
Prior	The PAGE UP key.
ProcessKey	The PROCESS KEY key.
Q	The Q key.
R	The R key.
RButton	The right mouse button.
RControlKey	The right CTRL key.
Return	The RETURN key.
Right	The RIGHT ARROW key.
RMenu	The right ALT key.
RShiftKey	The right SHIFT key.
RWin	The right Windows logo key (Microsoft Natural Keyboard).

S	The S key.
Scroll	The SCROLL LOCK key.
Select	The SELECT key.
SelectMedia	The select media key (Windows 2000 or later).
Separator	The separator key.
Shift	The SHIFT modifier key.
ShiftKey	The SHIFT key.
Sleep	The computer sleep key.
Snapshot	The PRINT SCREEN key.
Space	The SPACEBAR key.
Subtract	The subtract key.
T	The T key.
Tab	The TAB key.
U	The U key.
Up	The UP ARROW key.
V	The V key.
VolumeDown	The volume down key (Windows 2000 or later).
VolumeMute	The volume mute key (Windows 2000 or later).
VolumeUp	The volume up key (Windows 2000 or later).
W	The W key.
X	The X key.
XButton1	The first x mouse button (five-button mouse).
XButton2	The second x mouse button (five-button mouse).
Y	The Y key.
Z	The Z key.
Zoom	The ZOOM key.