FAQ C++/CLI et VC++.Net

Date de publication : 22 février 2013

Cette faq a été réalisée pour répondre aux questions les plus fréquemment posées concernant le C++/CLI et l'intéraction de Visual C++ et des MFC avec le framework .Net 2.0.

Je tiens à souligner que cette faq ne garantit en aucun cas que les informations qu'elle contient sont correctes ; Les auteurs font le maximum, mais l'erreur est humaine. Si vous trouvez une erreur, ou que vous souhaitez devenir redacteur, lisez Comment participer à cette faq ?.

Sur ce, je vous souhaite une bonne lecture.

Ont contribué à cette FAQ :

nico-pyright(c) - neo.51 - Clement Cunin - abelman - freegreg - Thomas Lebrun - Keihilin - Louis Guillaume Morand - Webman - swoög - dev01 - neguib - Ditch - doccpu - efficks - sam_XIII - HULK - merlin - StormimOn - Jérôme Lambert - Aspic - doudouallemand - farscape - NicolasJolet - trueman - smyley - jpjp507 - olsimare - Skalp

1. Informations générales (4) 4
2. Généralités sur le framework .Net (8) 6
3. Le langage C++/CLI (61) 9
3.1. Généralités (1) 10
3.2. Syntaxe (20) 11
3.2.1. Cast (3) 22
3.3. Tableaux, Collections et énumérations (5) 23
3.4. Classes, Interfaces, héritage et types de données (22) 26
3.4.1. Les types de données (12) 31
3.4.1.1. Variables et fonctions statiques (3) 35
3.4.2. Les interfaces (4) 37
3.5. La surcharge d'opérateur (5) 39
3.6. Les types et fonctions génériques (5) 43
3.7. Compatibilité d'assignation et comparaison de d'identité de types (3) 47
4. Mixer du C++/CLI avec du code Win32 ou MFC (24) 50
4.1. Conversions (6) 63
4.2. Intéropérabilité (2) 66
5. Intéraction du C++/CLI avec le framework .Net (140) 69
5.1. Système (24) 70
5.2. WinForms (39) 80
5.2.1. TextBox (6) 96
5.2.2. TreeView (3) 102
5.2.3. ListView (2) 104
5.2.4. Label (1) 106
5.2.5. Button (2) 108
5.3. GDI (4) 109
5.4. Fichiers, Répertoires, Disques (29) 111
5.4.1. Compression (2) 122
5.4.2. XML (5) 124
5.5. Thread, Processus (10) 131
5.6. Réseau (4) 138
5.7. ADO.NET (9) 140
5.8. Instanciation dynamique (2) 145
5.9. Office (9) 146
5.10. Divers (10) 151
6. IDE (11) 158
6.1. Modes de compilation (5) 160

Sommaire > Informations générales

Comment bien utiliser cette faq ?

Auteur : nico-pyright(c)

Le but : Cette faq a été conçue pour être la plus simple possible d'utilisation. Elle tente d'apporter des réponses simples et complètes aux questions auquelles les développeurs Visual C++ utilisant le framework .Net à l'aide du C++/CLI ont souvent été confrontés.
Lors de l'ajout ou de la modification d'une question/réponse, un indicateur est placé à côté du titre de la question.
L'organisation : Les questions sont organisées par thèmes, rendant la recherche plus facile.
Les réponses : Les réponses peuvent être complétées de liens vers d'autres réponses, vers la documentation en ligne de Microsoft ou vers un autre site en rapport.
Nouveautés et mises à jour : Lors de l'ajout ou de la modification d'une question/réponse, un indicateur est placé à côté du titre de la question. Cet indicateur reste visible pour une durée de 15 jours afin de vous permettre de voir rapidement les modifications apportées.
J'espère que cette faq pourra répondre à vos questions. N'hésitez pas à nous faire part de tous commentaires/remarques/critiques.

lien : Comment participer à cette faq ?

Comment participer à cette faq ?

Auteurs : Clement Cunin - neo.51

Cette faq est ouverte à toute collaboration. Pour éviter la multiplication des versions, il serait préférable que toute collaboration soit transmise aux administrateurs de la faq.
Plusieurs compétences sont actuellement recherchées pour améliorer cette faq :
Rédacteur : Bien évidemment, toute nouvelle question/réponse est la bienvenue.
Correcteur : Malgré nos efforts des fautes d'orthographe ou de grammaire peuvent subsister. Merci de contacter les administrateurs si vous en débusquez une... Idem pour les liens erronés.

lien : Quels sont les droits de reproduction de cette faq ?

Quels sont les droits de reproduction de cette faq ?

Auteur : neo.51

Merci de contacter les auteurs pour toute copie, intégrale ou partielle de ce document, voir Comment participer à cette faq ?.

lien : Comment participer à cette faq ?

Nous tenons à remercier

Auteur : nico-pyright(c)

Je tiens à remercier tout particulièrement LFE pour son travail sur la coloration syntaxique du code, et pour toutes les petites mises à jours du système de FAQ qui ont été réalisées.
Je remercie aussi :
Clément Cunin pour son travail sur ce système de génération de FAQ.
Nono40 pour son outil d'édition XML sans lequel la FAQ serait sortie bien plus tard ;-).
Anomaly pour la correction de l'orthographe ;-).

L'ensemble de l'équipe des rédacteurs de www.developpez.com pour leurs remarques constructives.

Sommaire > Généralités sur le framework .Net

Qu'est-ce que le framework .Net 2 ?

Auteur : nico-pyright(c)

Pour résumer en termes simples, .Net représente la volonté de Microsoft de délivrer le logiciel comme un service.
Pour les développeurs, il s'agit d'une nouvelle approche de conception, de développement et de déploiement du logiciel.
Pour les architectures, il s'agit d'augmentation de capacité de déploiement des applications Internet : moins sur les clients et plus sur les serveurs, pour une meilleure utilisation d'internet, orientant le futur d'Internet vers un système d'exploitation plutôt qu'en un réseau d'ordinateurs.

Le framework .Net comprend tout ce qu'il faut pour développer, déployer, exécuter des Web Services, des applications Web et des applications Windows. On peut voir le framework comme une application 3-tiers :

- **Technologies pour développer des applications**
- **Bibliothèque de classes du framework .Net**
- **Common language runtime (CLR)**

Le CLR est le moteur d'exécution du noyau .Net pour exécuter des applications. Il apporte des services tels que la sécurité d'accès de code, la gestion de la vie d'un objet, la gestion des ressources, la sûreté des types, etc ...

En terme de déploiement, Visual Studio 2005 propose des outils de déploiement extrêmement simples et rapides. De plus, il faut veiller à ce que le framework 2.0 soit installé sur la machine.

Qu'est-ce que le garbage collector ?

Auteur : nico-pyright(c)

Le garbage collector est un mécanisme qui permet à un ordinateur de détecter et de supprimer les objets managés du heap qui ne sont plus référencés par une application.
Le garbage collector du framework .net ajoute la fonctionnalité intéressante de compacter la mémoire après libération des objets managés inutilisés.
Le garbage collector est une révolution de programmation, car il annonce la fin des fuites mémoires, qui font rager tout développeur. Un développeur n'a plus à se soucier d'appeler la destruction de ses objets avec l'opérateur delete.

lien : Comment fonctionne le Garbage Collector ?

Qu'est-ce qu'It Just Work ?

Auteur : nico-pyright(c)

Sous Visual Studio 2003, la technologie d'intéropabilité s'appelle IJW : "It Just Works". Avec Visual Studio 2005, elle est désormais appelée "Interop Technologies". Vous entendrez parler indifféremment des deux, qui représentent le même concept.
Pour toute méthode native dans une application, le compilateur crée un point d'entrée managé et un point d'entrée non managé. L'un deux est l'actuelle méthode d'implémentation alors que l'autre permet de créer un pont et permet l'utilisation des opérations de Marshaling. C'est le point d'entré managé qui est toujours utilisé, sauf quand le compilateur ne peut pas retranscrire le code en MSIL ou lorsque l'on utilise le "#pragma unmanaged".

IJW permet d'accéder directement, sans ajouter de code, à des méthodes non managées à partir de code managé. On peut de même utiliser des wrapper pour avoir un accès direct aux objets COM, ou aux dll natives.
Toujours sans code ajouté, sans DLLImport, le méchanisme IJW est bien souvent plus rapide. Il s'agit juste d'inclure le fichier d'entête et de linker avec la librairie d'import.

Qu'est-ce que le CLR (Common Language Runtime)?

Auteur : nico-pyright(c)

On l'appelle aussi le runtime .NET. Le CLR est le moteur d'exécution du noyau .Net pour exécuter des applications. Il apporte des services tels que la sécurité d'accès de code, la gestion de la vie d'un objet, la gestion des ressources, la sûreté des types, etc ...

Qu'est-ce que le CLS (Common Language Specification)?

Auteur : nico-pyright(c)

Pour interagir entièrement avec des objets managés quel que soit le langage dans lequel ils ont été implémentés, les objets managés ne doivent exposer aux appelants que les fonctionnalités qui sont communes à tous les langages avec lesquels ils doivent fonctionner.
Pour cette raison, un ensemble de fonctionnalités de langage appelé spécification CLS (Common Language Specification), qui comprend les fonctionnalités de langage de base nécessaires à de nombreuses applications, a été défini.
La spécification CLS permet d'optimiser et d'assurer l'interopérabilité des langages en définissant un ensemble de fonctionnalités sur lequel les développeurs peuvent compter dans de nombreux langages. La spécification CLS établit également des exigences de conformité CLS ; elles vous permettent de déterminer si votre code managé est conforme à la spécification CLS et dans quelle mesure un outil donné prend en charge le développement du code managé qui utilise des fonctionnalités CLS.
Si votre composant n'utilise que des fonctionnalités CLS dans des interfaces API qu'il expose à un autre code (y compris des classes dérivées), le composant est assuré d'être accessible à partir d'un langage de programmation prenant en charge la spécification CLS.

Il est important de comprendre que le respect du CLS augmente la sécurité de votre programme.

Qu'est-ce que le CTS (Common Type System)?

Auteur : nico-pyright(c)

Afin que des classes définies dans plusieurs langages puissent communiquer entres elles, elles ont besoin d'un ensemble de types de données communs. C'est l'objet du CTS, il définit les types de données que le Runtime .NET comprend et que les applications .NET peuvent utiliser.

Qu'est-ce que le CLI (Common Language Infrastructure)?

Auteur : nico-pyright(c)

Dans le contexte de C++/CLI, cela correspond à du C++ pouvant interagir avec le CLR. Qu'est-ce que le C++/CLI ?

lien : Qu'est-ce que le C++/CLI ?

Qu'est-ce que le MSIL ?

Auteur : nico-pyright(c)

C'est un jeu d'instruction indépendant du CPU généré par les compilateurs .NET, à partir de langages comme le J#, C# ou Visual Basic. Le langage MSIL est compilé avant ou pendant l'exécution du programme par le VES (Virtual Execution System), qui fait partie intégrante du CLR.

Sommaire > Le langage C++/CLI

Sommaire > Le langage C++/CLI > Généralités

Qu'est-ce que le C++/CLI ?

Auteur : nico-pyright(c)

Dans C++/CLI, il y a deux choses. C++ et CLI.
C++ est bien sur le langage de programmation libre, inventé par Bjarne Stroustrup, soumis à la standardisation ISO. Plus de détails dans la faq C++.
CLI veut dire Common Language Infrastructure. C'est une spécification qui défini un environnement d'exécution pouvant supporter des langages multiples et être utilisés sur plusieurs plateformes sans avoir a être réécrit spécifiquement pour elles. Cette spécification est définie suivant la norme ECMA 335. Le CLR (comman language runtime, ou plus simplement le runtime .Net) est l'implémentation de Microsoft du CLI.
C++/CLI représente donc la mise en relation synallagmatique entre le C++ et le CLI qui relie le modèle objet statique du C++ au modèle objet dynamique du CLI. C'est cette mise en relation qui va permettre d'utiliser le langage C++ pour faire de la programmation .Net et d'accéder ainsi au framework .Net, grâce au compilateur de Visual Studio 2005.
C++/CLI est finalement une extension du C++ pour supporter le CLI.

lien : Qu'est-ce que le CLR (Common Language Runtime)?
lien : http://cpp.developpez.com/faq/cpp/?page=generalites

Sommaire > Le langage C++/CLI > Syntaxe

Qu'est-ce qu'un handle d'objet (^) ?

Auteur : nico-pyright(c)

On utilise l'opérateur carret (ou hat) [^] pour définir un handle vers un objet managé qui est en fait une référence vers cet objet managé. Attention ce n'est pas un pointeur. Un handle est une référence sur un objet managé sur le tas (heap) managé, alors que les pointeurs pointent vers une zone mémoire.
Pour allouer un objet managé et disposer d'un handle sur cet objet, on utilise gcnew.

lien : Comment allouer un objet managé avec gcnew ?

Qu'est-ce qu'une tracking reference (%) ?

Auteur : nico-pyright(c)

Une tracking reference est similaire à une référence C++ dans la mesure où elle représente un alias pour un objet / type définit sur le heap CLR.
Bien que les tracking references soient créés sur la pile, elles peuvent référencer un type sur la pile ou un handle dans le heap CLR.
On utilise l'opérateur % pour définir une telle référence. On s'en sert souvent pour pouvoir modifier un paramètre dans une fonction.

```
int entier = 1;
int% reference = entier;
reference = 2;
```

Après ces lignes, entier vaut 2.

Comment utiliser le référencement et le déférencement ?

Auteur : nico-pyright(c)

Il s'avère utile de comprendre les handles et les références, comme en C++.
Voyons cet exemple :

```
int main(array<System::String ^> ^args)
{
 int ^x = 5;
 x = *x + 2;
 int y = 6;
 int %z = y;
 inc(*x);
 inc(y);
 inc(z);
 Console::WriteLine("x : {0} / y : {1}", x,y);
 Console::WriteLine("Somme : {0} ", add(x,y));
}
void inc(int %a)
{
 a++;
}

int add(int ^a, int ^b)
{
```

```
 return *a + *b;
}
```

- On crée un handle d'entier x. La valeur de l'entier x vaut 5.
- On ajoute 2 à x, il vaut donc 7. Notez le déréférencement de x dans la partie droite qui permet d'accéder à sa valeur. Le déréférencement de x à gauche est facultatif, le compilateur le faisant pour nous.
- On déclare un entier y dont la valeur est 6.
- On créé une référence CLI z qui est un alias de y. Notez qu'on ne peut pas créer une référence CLI sur un handle, dans le cas de x, il faudrait évidement le déréférencer (int %z = *x).
- La fonction inc attend une référence afin de pouvoir modifier la valeur passée en paramètre. On déréférence x pour le passer à la fonction. x vaut maintenant 8.
- On incrémente y maintenant. y vaut 7.
- Puis on incrémente z. z étant un alias de y, c'est évidement y qui est incrémenté. y vaut maintenant 8.
- On affiche x et y, à savoir 8 et 8.
- Puis la fonction add qui accepte deux handles en paramètre, affiche donc 16.

Remarque : il est inutile de vouloir connaitre la valeur d'un handle, le garbage collector pouvant la changer à son aise.

Comment modifier la valeur d'un objet pointé par un handle passé en paramètre d'une fonction ?

Auteur : nico-pyright(c)

Comme en C++, on utilise une référence sur le handle, cela donne une syntaxe particulière mais le principe est le même :

```
void change(String ^%chaine)
{
 chaine = "Nouvelle";
}

int main()
{
 String ^chaine = gcnew String("Ancienne");
 Console::WriteLine(chaine);
 change(chaine);
 Console::WriteLine(chaine);
}
```

Qu'est-ce qu'un pointeur interne ?

Auteur : nico-pyright(c)

Un pointeur interne peut être assimilé au pointeur classique du C. Il "pointe" vers un objet managé, donc soumis au garbage collector. Il représente une adresse qui peut évoluer en fonction des opérations du GC (compactage, allocation, etc ...). On récupère un pointeur interne avec le mot clé interior_ptr.

```
array<double>^ tableau = {1, 2, 3, 4, 5};
interior_ptr<double> p_tableau = &tableau[0];
```

Si le pointeur interne n'est pas initialisé, il contient par défaut nullptr.
Voici un exemple de ce qu'on peut faire avec un pointeur interne :

```
array<String^>^ chaine = { L"Ne",L"touchez",L"pas",L"trop",L"aux",L"pointeurs"};
for(interior_ptr<String^> p_chaine = &chaine[0]; p_chaine - &chaine[0] <
 chaine->Length ; ++p_chaine)
 Console::WriteLine(*p_chaine);
```

Cet exemple produit en sortie :

```
Ne
touchez
pas
trop
aux
pointeurs
```

Je conseille bien évidement de se passer des pointeurs internes, sauf quand cela est indispensable.

Qu'est-ce qu'un pointeur épingle (pin_ptr) ?

Auteur : nico-pyright(c)

On utilise le mot clé pin_ptr pour déclarer un *pinning pointer*, ce que j'ai vu traduire par un pointeur épingle. Il s'agit en fait d'un pointeur interne (Qu'est-ce qu'un pointeur interne ?) dont l'objet pointé ne sera pas déplacé en mémoire par le mécanisme du garbage collector. Concrètement, ce pointeur ne changera pas de valeur.
Ceci est indispensable lorsqu'on travaille directement sur le pointeur dans des fonctions non managées, par exemple lors de conversion de chaînes.
Comment convertir une String ^ en wchar_t * ?

lien : Qu'est-ce qu'un pointeur interne ?

Comment allouer un objet managé avec gcnew ?

Auteur : nico-pyright(c)

On utilise gcnew pour obtenir un handle sur un objet managé par le CLR.

```
String ^ str = gcnew String("Ma chaîne managée est alloué sur le heap managé");
```

Le garbage collector s'occupe de la destruction de l'objet.

Quel est la différence entre le destructeur et le finalizer ?

Auteur : nico-pyright(c)

Un finalizer est une fonction particulière de la classe, au même titre que le constructeur ou le destructeur, qui est appelée automatiquement par le garbage collector quand un objet est détruit.
Il diffère du destructeur selon les éléments suivants :

- **Le finalizer n'est pas appelé si le destructeur a été appelé explicitement lors de la destruction de l'objet.**
- **Le finalizer est appelé lorsque l'objet meurt naturellement en sortant du scope.**

On définit le finalizer avec l'opérateur !.

```
ref class MaClasse
{
public:
 MaClasse(int v) : valeur(v){} // constructeur
 ~MaClasse() // destructeur
 {
  Console::WriteLine("Destructeur de l'objet ({0})", valeur);
 }
protected:
 !MaClasse() // finalizer
 {
  Console::WriteLine("Finalizer de l'objet ({0})", valeur);
 }
private:
 int valeur;
};

int main(array<System::String ^> ^args)
{
 MaClasse^ obj1 = gcnew MaClasse(1);
 MaClasse^ obj2 = gcnew MaClasse(2);
 delete obj1;
 Console::WriteLine("Fin");
 return 0;
}
```

Cet exemple produit le résultat suivant :

```
Destructeur de l'objet (1)
Fin
Finalizer de l'objet (2)
```

Notez que le finalizer de l'objet 2 est appelé après la fin du programme, lorsqu'il sort du scope.
Il est important de comprendre qu'il y a à chaque fois seulement le finalizer ou seulement le destructeur qui est appelé, suivant la méthode de destruction. Mais jamais les deux.

On peut conclure de cet exemple que si l'on veut être sur que des éléments (non managés par exemple) utilisés par un objet sont bien détruit, tout en ne tenant pas compte de la façon dont un objet est détruit (explicitement ou par le GC), alors il faut implémenter à la fois un destructeur et un finalizer.

Remarque : Le destructeur est déclaré en public, le finalizer doit l'être en protected.

Qu'est-ce qu'un espace de nom (namespace) ?

Auteur : nico-pyright(c)

Les classes (objets) .Net sont regroupées sémantiquement en catégories, sous la forme de bibliothèques. Elles forment un espace de noms : namespace.
Exemple : le namespace System::Collections contient les différents objets de collections du framework.net. Ils sont accessibles directement en indiquant le chemin et en utilisant les :: ou bien en important le namespace directement avec using namespace.

```
System::Collections::ArrayList ^ tableau = gcnew System::Collections::ArrayList();
```

ou bien :

```
using namespace System::Collections;
ArrayList ^ tableau = gcnew ArrayList();
```

Comment créer une fonction avec un nombre d'arguments variable ?

Auteur : nico-pyright(c)

Le C++/CLI nous autorise une telle chose grâce aux ... et à un array CLI :

```
int somme(... array<int>^ args)
{
 int result = 0;
 for each(int arg in args)
  result += arg;
 return result;
}

int main(array<System::String ^> ^args)
{
 Console::WriteLine(somme(1,2,3,4,5,6));
 return 0;
}
```

Quelles sont les visibilités ajoutées par le C++/CLI ?

Auteur : nico-pyright(c)

En plus des classiques *private, public et protected*, le C++/CLI ajoute trois nouvelles visibilités :

- **internal : Les membres sont accessibles depuis la classe, à l'intérieur de l'assembly parent.**
- **public protected : Les membres sont accessibles dans des classes dérivées de notre classe à l'extérieur de l'assembly parent et dans n'importe quelle classe à l'intérieur de l'assembly parent.**
- **private protected : Les membres sont accessibles dans les classes dérivées de notre classe uniquement à l'intérieur de l'assembly parent.**

Qu'est ce qu'un delegate ?

Auteur : farscape

Un delegate est une classe managée (ref class) qui permet d'appeler une méthode qui partage la même signature qu'une fonction globale ou d'une classe possédant une méthode avec cette même signature. Le framework supporte deux formes de delegates :
un delegate qui accepte d'appeler uniquement une méthode :

```
System::Delegate
```

et un delegate qui accepte d'appeler une chaine de méthodes :

```
System::MulticastDelegate
```

Les méthodes utilisées pour le delegate peuvent être :
- Globale (comme en C) - une méthode statique à une classe - une méthode d'une instance.
Exemple :
Déclaration du delegate :

```
delegate void FuncMessDelegate(String ^mess);
```

Maintenant les trois formes d'utilisation :

```
void GlobalMess(String ^mess)
{
 Console::Write("Fonction Globale: ");
 Console::WriteLine(mess);
}
ref class OneClass
{
public:
 static void staticMethodeMess(String ^mess)
 {
  Console::Write("Fonction statique: ");
  Console::WriteLine(mess);
 }
};
ref class AnotherClass
{
public:
 void MethodeMess(String ^mess)
 {
  Console::Write("Fonction d'une instance: ");
  Console::WriteLine(mess);
 }
};
```

L'appel de la fonction :

```
int main(array<System::String ^> ^args)
{
 // declaration du delegate
 FuncMessDelegate ^Global= gcnew FuncMessDelegate(&GlobalMess);
 // ajouter une fonction au delegate
 FuncMessDelegate ^statique= gcnew FuncMessDelegate(&OneClass::staticMethodeMess);

 AnotherClass ^pAnotherClass= gcnew AnotherClass;

 FuncMessDelegate ^instance= gcnew FuncMessDelegate(pAnotherClass,&AnotherClass::MethodeMess);

 // l'appel de la fonction…
 Global->Invoke("Hello");

 statique->Invoke("Hello");
 instance->Invoke("Hello");

 return 0;
}
```

Les delagates peuvent être combinés sous forme de chaine et un élément peut être supprimé (Multicast Chain).
On utilisera la méthode Combine() ou l'operateur + pour l'ajout.
Et la méthode Remove() ou l'opérateur - pour la suppression.
Exemple :

```
FuncMessDelegate ^ChaineMess= gcnew FuncMessDelegate(&GlobalMess);

ChaineMess += gcnew FuncMessDelegate(&OneClass::staticMethodeMess);

AnotherClass ^pAnotherClass= gcnew AnotherClass;
ChaineMess += gcnew FuncMessDelegate(pAnotherClass,&AnotherClass::MethodeMess);

ChaineMess->Invoke("Hello");

ChaineMess -= gcnew FuncMessDelegate(&OneClass::staticMethodeMess);
ChaineMess->Invoke("Hello2");
```

Note :
J'ai volontairement utilisé les operateurs à la place des méthodes car ceux-ci imposent un cast vers le delegate déclaré ce qui alourdi grandement l'écriture.

Qu'est ce qu'un Event ?

Auteur : farscape

Un Event est une implémentation spécifique du delegate ou plutôt du multicast delegate.
Un event permet à partir d'une classe d'appeler des méthodes situées dans d'autres classes sans rien connaitre de ces classes.
Il est ainsi possible pour une classe d'appeler une chaine de méthodes issues de différentes classes.
Exemple :

```
using namespace System;
```

```
delegate void AnalyseHandler(int %n); // la fonction de traitement recoit une reference sur un entier.

// classe declencheur de traitment
ref class AnalyseTrigger
{
public:
    event AnalyseHandler ^OnWork;
    void RunWork(int %n)
    {
        OnWork(n);
    }
};

// classe effectuant un traitement
ref class Analyse
{
    public:
    AnalyseTrigger ^ m_AnalyseTrigger;
    Analyse(AnalyseTrigger ^src)
    {
        if(src==nullptr)
            throw gcnew ArgumentNullException("erreur argument non specifié");
        m_AnalyseTrigger=src;
        m_AnalyseTrigger->OnWork+= gcnew AnalyseHandler(this,&Analyse::Treatment);
        m_AnalyseTrigger->OnWork+= gcnew AnalyseHandler(this,&Analyse::TreatmentTwo);
    }
    void RemoveTreatmentTwo()
    {
        m_AnalyseTrigger->OnWork-=gcnew AnalyseHandler(this,&Analyse::TreatmentTwo);
    }
    // traitement declenché
    void Treatment(int %n)
    {
        Console::Write("Class Analyse Treatment =+10 n:= ");
        Console::Write(n);
        n+=10;
        Console::Write(" -> n=");
        Console::WriteLine(n);
    }
    // traitement declenché
    void TreatmentTwo(int %n)
    {
        Console::Write("Class Analyse TreatmentTwo =*2 n:= ");
        Console::Write(n);
        n*=2;
        Console::Write(" -> n=");
        Console::WriteLine(n);
    }
};

// classe effectuant un traitement
ref class AnalyseTwo
{
    public:
    AnalyseTrigger ^m_AnalyseTrigger;

    AnalyseTwo(AnalyseTrigger ^src)
    {
        if(src==nullptr)
            throw gcnew ArgumentNullException("erreur argument non specifié");
        m_AnalyseTrigger=src;
        m_AnalyseTrigger->OnWork+= gcnew AnalyseHandler(this,&AnalyseTwo::Treatment);

    }
    // traitement declenché
    void Treatment(int %n)
    {
```

```
        Console::Write("Class AnalyseTwo Treatment =+2 n:= ");
        Console::Write(n);
        n+=2;
        Console::Write(" -> n=");
        Console::WriteLine(n);
    }
};
int main(array<System::String ^> ^args)
{
    //element declecheur de l'action fixée par le delegate AnalyseHandler
    AnalyseTrigger ^Trigger = gcnew AnalyseTrigger();

    // classe traitements
    Analyse ^analyse = gcnew Analyse(Trigger);

    // classe traitements
    AnalyseTwo ^analyseTwo = gcnew AnalyseTwo(Trigger);

    int n=2;

    Trigger->RunWork(n);
    analyse->RemoveTreatmentTwo();

    Console::WriteLine("------------------------");
    n=2;
    Trigger->RunWork(n);

    return 0;
}
```

Comment faire de la destruction déterministe en C++/CLI (auto_handle) ?

Auteur : nico-pyright(c)

Le C++/CLI dispose du template auto_handle pour forcer la destruction d'un objet quand il sort de portée (il fonctionne comme le template auto_ptr du C++).
Pour utiliser le template, on doit inclure :

```
#include <msclr\auto_handle.h>
```

Exemple de code :

```
#include <msclr\auto_handle.h>

using namespace System;

ref class CManagee
{
private:
  String ^s;
public:
  CManagee(String ^val) {s = val;};
  ~CManagee() { Console::WriteLine("Destructeur : " + s);  }
protected:
  !CManagee() { Console::WriteLine("Finalizer :" + s); }
};

int main(array<System::String ^> ^args)
{
  {
    msclr::auto_handle<CManagee> cAutoHandle = gcnew CManagee("auto_handle");
    CManagee ^cNormal = gcnew CManagee("normal");
    Console::WriteLine("Après la construction");
```

```
    }
    Console::WriteLine("fin");
    return 0;
}
```

On aura en sortie :

```
Après la construction
Destructeur : auto_handle
fin
Finalizer : normal
```

J'ai rajouté des { } pour forcer un bloc et montrer alors la destruction par l'appel au destructeur quand le auto_handle sort de portée. Tandis que le handle normal, lui, est détruit par le garbage collector en fin de programme.
Le template surcharge les opérateurs ->, ==, etc ... ainsi il peut être utilisé comme un handle classique.
Notez que si on souhaite pouvoir continuer à utiliser le auto_handle une fois qu'il est sorti de portée, il faut au préalable l'avoir libéré avec release :

```
{
  msclr::auto_handle<CManagee> cAutoHandle = gcnew CManagee("auto_handle");
  cAutoHandle.release();
  Console::WriteLine("Après la construction");
}
Console::WriteLine("fin");
```

Cette fois-ci en sortie on aura :

```
Après la construction
fin
Finalizer : auto_handle
```

Comment utiliser un mot clé comme nom de variable ?

Auteur : nico-pyright(c)

Il est possible d'utiliser des mots clés réservés du C++/CLI comme nom de variable. On utilise le mot clé __identifier :

```
String ^ __identifier(gcnew) = gcnew String("Abc");
Console::WriteLine(__identifier(gcnew));
__identifier(gcnew) = "DEF";
Console::WriteLine(__identifier(gcnew));
```

Je trouve personnellement que ça alourdit le code pour un intérêt limité. Cela pourrait s'avérer utile dans le cadre d'une migration. Auparavant, on utilisait une variable nommée gcnew. Une fois passé à Visual C++

2005, pour continuer à utiliser cette variable on doit l'entourer de __identifier. Mon conseil est de changer de nom de variable.

Peut-on utiliser l'opérateur ?? du C# ?

Auteur : nico-pyright(c)

Non, cet opérateur n'est pas disponible en C++/CLI.
Pour rappel, en C# on peut utiliser cet opérateur comme suit :

```
Console.WriteLine(i ?? 0);
```

Cela permet d'utiliser l'entier 0 si jamais l'entier passé est nul.
Pour faire l'équivalent en C++/CLI, on fera

```
Console::WriteLine(i.HasValue ? i : 0);
```

Notez que l'on teste la propriété HasValue du type nullable i pour vérifier s'il contient une valeur.

Peut-on utiliser des classes partielles (partial) ?

Auteur : nico-pyright(c)

Non, il n'y a pas d'équivalent en C++/CLI.

Sommaire > Le langage C++/CLI > Syntaxe > Cast

Qu'est-ce que le cast ?

Auteur : nico-pyright(c)

Le cast entre types est le fait de convertir un type en un autre. Il fonctionne aussi bien pour les types natifs (int, double, etc ...) que pour les classes et les structures.
Le C++/CLI propose trois opérateurs de cast différents : static_cast, dynamic_cast et safe_cast.
Chacun a pour but d'essayer de convertir un type en un autre. "Essayer", car pour pouvoir convertir décemment une classe en une autre, elles ont besoin d'être du même type ou bien d'en hériter. (on parle alors de downcasting).

L'opérateur static_cast est le plus rapide d'entre eux, mais c'est également le plus dangereux, car il part du principe que le développeur sait ce qu'il fait, donc il ne teste pas la validité du cast.
L'opérateur dynamic_cast est plus lent que le static_cast car il vérifie que la conversion de type est valide. Si elle ne l'est pas, il renverra nullptr.
L'opérateur safe_cast enfin permet la conversion vérifiable entre les types de .Net. C'est cet opérateur qu'on va donc utiliser le plus souvent.

lien : Les casts dans la faq C++
lien : Comment implémenter l'opérateur d'assignation 'as' du C# ?
lien : Pourquoi utiliser safe_cast ?

Pourquoi utiliser safe_cast ?

Auteur : nico-pyright(c)

safe_cast est similaire à static_cast dans la mesure où il permet les conversions implicites et explicites de types. Il ressemble à dynamic_cast dans la mesure où il vérifie la validité du cast.

L'intérêt de caster en utilisant safe_cast plutôt que static_cast est qu'il produit du MSIL vérifiable. Cela permet aussi d'intercepter un message d'exception lorsque le cast échoue (System::InvalidCastException).

Que fait le cast C ?

Auteur : nico-pyright(c)

Le cast (old style) C a pour syntaxe :

```
MonType ^type1 = (MonType^)(type2);
```

Lorsqu'on l'utilise, c'est d'abord le safe_cast qui sera utlisé comme cast. Ce qui fait qu'on pourra se passer d'écrire safe_cast en utilisant le cast style C.
Ainsi, ces deux lignes sont équivalentes :

```
TextBox ^T = safe_cast<TextBox ^>(sender);
TextBox ^T = (TextBox ^)(sender);
```

Sommaire > Le langage C++/CLI > Tableaux, Collections et énumérations

Comment déclarer et utiliser les tableaux à une dimension ?

Auteur : nico-pyright(c)

Pour utiliser un tableau en C++/CLI, vous pouvez utiliser le mot clé array du namespace cli.
Ainsi, vous pouvez déclarer un tableau et l'initialiser en une fois ou élément après élément.

```
cli::array<String^>^ tabChaine = { "Element 1", "Element 2", "Element 3"};
```

```
cli::array<String ^, 1> ^ tabChaine = gcnew cli::array<String ^>(3);
tabChaine[0] = "Element 1";
tabChaine[1] = "Element 2";
tabChaine[2] = "Element 3";
```

Pour accéder aux éléments du tableau, on peut utiliser l'accès direct avec [] ou dans une boucle for each.

```
for (int i=0;i<tabChaine->Length;i++)
 Console::WriteLine(tabChaine[i]);
```

```
for each (String ^ s in tabChaine)
 Console::WriteLine(s);
```

Comment parcourir un tableau ou une collection avec for each ?

Auteur : nico-pyright(c)

Avec le framework .Net est apparu pour le C++ le mot clé for each. Il est très utile pour parcourir un tableau ou une collection.
Chaque itération initialise un objet qui contient la valeur courante de l'itération.

```
using namespace System;
using namespace System::Collections::Generic;

List <String ^> ^ listChaine = gcnew List<String ^>();
listChaine>Add("element 1");
listChaine>Add("element 2");

for each (String ^ s in listChaine)
 Console::WriteLine(s);
```

Comment trier un tableau ?

Auteur : nico-pyright(c)

Le namespace Array permet de trier des tableaux CLI grâce à la méthode Sort().

```
array<int>^ nombres = { 99, 43, 28, 11, 16, 52, 30};
Array::Sort(nombres);
for each(int nombre in nombres)
 Console::Write(nombre + " ");
```

Ceci affichera :

```
11 16 28 30 43 52 99
```

Comment rechercher une valeur dans un tableau ?

Auteur : nico-pyright(c)

Le namespace Array permet de rechercher un objet dans des tableaux CLI grâce à la méthode IndexOf(). On peut aussi utiliser la méthode BinarySearch.

```
array<String^>^ jours = { "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche"};
int pos = Array::IndexOf(jours, "Mercredi");
if (pos > 0)
 Console::WriteLine("Mercredi a été trouvé à la position {0}", pos );
else
 Console::WriteLine("Mercredi n'a pas été trouvé");
```

Remarque : Si BinarySeach retourne une valeur négative, alors l'élément n'est pas trouvé. Cependant, avec cette valeur nous disposons d'une indication sur l'emplacement de l'élément qui est juste supérieur à la recherche.
Cela peut être utile pour insérer une valeur dans un tableau trié. Il suffit alors de faire un complément à zéro de la position :

```
pos = ~pos;
```

Comment faire pour que sa classe soit énumérable avec for each ?

Auteur : nico-pyright(c)

Il peut être intéressant de faire en sorte que ses classes persos puissent être parcourues grâce au mot clé for each.

Pour ceci, il faut que la classe soit en mesure de fournir un objet IEnumerator et implémente l'interface IEnumerable afin de surcharger les méthodes MoveNext et Reset tout en permettant l'accès à la propriété Current.

Une solution est d'avoir une classe implémentant les interfaces IEnumerable et IEnumerator. Voici l'exemple d'une classe MyString, qui en l'occurence permet le parcours d'une chaîne avec for each (ce qui sert bien sur uniquement d'exemple, dans la mesure où la classe String le permet déjà).

```
using namespace System;
using namespace System::Collections;

public ref class MyString : public IEnumerable, public IEnumerator
{
private:
 String ^string_;
 int i;
public:
 MyString()
 {
  string_ = "";
  i = -1;
 }
 MyString(const char * c)
 {
  string_ = gcnew String(c);
  i = -1;
 }
 virtual bool MoveNext(void)
 {
  if ( i < string_->Length - 1 )
  {
   i++;
   return true;
  }
  return false;
 }
 virtual void Reset() {}
 property Object^ Current
 {
  virtual Object^ get()
  {
   return string_[i];
  }
 };
 virtual IEnumerator^ GetEnumerator()
 {
  return this;
 }
};

int main()
{
 MyString ^maChaine = gcnew MyString("abcd");
 for each ( Object^ c in maChaine )
  Console::WriteLine(c);
}
```

On remarquera la méthode GetEnumerator qui renvoi this.

Une autre solution pourrait être d'utiliser une structure nested implémentant IEnumerator.

Sommaire > Le langage C++/CLI > Classes, Interfaces, héritage et types de données

Comment définir une classe virtuelle pure (abstract) ?

Auteur : nico-pyright(c)

On définit en C++/CLI une classe virtuelle pure en utilisant le mot clé sensible au contexte abstract (voir Qu'est-ce qu'une fonction virtuelle pure ? dans la faq c++).
Ainsi, l'utilisation d'abstract indique qu'une classe et ses membres abstraits peuvent être uniquement définis dans une classe dérivée et ne pourront pas être instanciés (seulement servir de classe de base).
On déclare une classe virtuelle pure ainsi :

```
ref class c abstract
{
public:
 virtual void f() abstract;
};
```

lien : Qu'est-ce qu'une fonction virtuelle pure ? dans la faq c++

Comment surcharger une méthode abstraite ou virtuelle (override) ?

Auteur : nico-pyright(c)

On utilise le mot clé sensible au contexte override pour préciser explicitement que l'on surcharge une méthode (abstraite ou virtuelle).

```
ref class c abstract
{
public:
 virtual void f() abstract;
};

ref class d : c
{
public:
 virtual void f() override
 {
  // implémentation
 }
};
```

Il s'agit ici d'une surcharge implicite.

Comment définir une classe ou une méthode sealed ?

Auteur : nico-pyright(c)

On utilise le mot clé sensible au contexte sealed pour indiquer qu'une classe ou une méthode ne peut pas être dérivée.
Ainsi, dans l'exemple suivant, la méthode f de la casse c ne peut être surchargée, et provoque ainsi une erreur de compilation :

```
ref class c
```

```
{
public:
   virtual void f() sealed
   {
      System::Console::WriteLine("f ne pourra pas être dérivée");
   }
};

ref class d : public c
{
public:
   virtual void f() override  // erreur du compilo
   {
      // ....
   }
};
```

```
error C3764: 'd::f': cannot override base class method 'c::f'
```

On définit une classe sealed de la même facon :

```
ref class c sealed
{
public:
   void f()
   {
      // ...
   }
};

ref class d : public c  // erreur
{

};
```

Ici on obtiendra l'erreur de compilation suivante :

```
error C3246: 'd' : cannot inherit from 'c' as it has been declared as 'sealed'
```

Comment rompre le polymorphisme d'une fonction (new) ?

Auteur : nico-pyright(c)

On utilise le mot clé sensible au contexte new pour indiquer qu'une fonction ne surcharge pas une méthode d'une classe mère.
Ainsi l'exemple suivant :

```
ref class Animal
{
public :
 virtual void QuiSuisJe()
 {
  Console::WriteLine("Je suis un animal");
 }
};
```

```
ref class Dog : Animal
{
public :
 virtual void QuiSuisJe() override
 {
  Console::WriteLine("Je suis un chien");
 }
};

ref class Cat : Animal
{
public :
 virtual void QuiSuisJe() new
 {
  Console::WriteLine("Je suis un chat");
 }
};

int main()
{
 Animal ^an1 = gcnew Animal();
 Animal ^an2 = gcnew Dog();
 Animal ^an3 = gcnew Cat();

 an1->QuiSuisJe();
 an2->QuiSuisJe();
 an3->QuiSuisJe();
}
```

produira le resultat suivant :

```
Je suis un animal
Je suis un chien
Je suis un animal
```

On constate bien que l'objet Animal (Cat) ne peut pas utiliser la méthode QuiSuisJe() de la classe Cat, celle-ci n'étant pas la surcharge de la méthode QuiSuisJe de la classe Animal, à contrario de la classe Dog.

Comment implémenter un constructeur de copie ?

Auteur : nico-pyright(c)

Un constructeur de copie instancie un objet en créant une copie d'un autre objet. Alors qu'en C++, un constructeur de copie par défaut est généré automatiquement, en C++/CLI ce n'est pas le cas.
Considérons l'exemple suivant :

```
ref class Personne
{
private:
 String ^nom;
 String ^prenom;
public:
 Personne(String ^n, String ^p) : nom(n), prenom(p){}
 Personne(const Personne^ p) : nom(p->nom), prenom(p->prenom){}
};

// exemple d'appel
Personne ^p1 = gcnew Personne("nico", "pyright");
```

```
Personne ^p2 = gcnew Personne(p1);
```

Nous avons ainsi défini un constructeur de copie.

NB : Une limitation de cet exemple est qu'on ne peut utiliser le constructeur de copie qu'avec des handles d'objet, et non avec un objet lui même.
L'exemple suivant produit une erreur de compilation :

```
Personne p3("nico", "pyright");
Personne ^p4 = gcnew Personne(p3); // erreur de compilation C3073
```

Pour contourner ce problème, on peut utiliser l'opérateur de référence %.

```
Personne p3("nico", "pyright");
Personne ^p4 = gcnew Personne(%p3);
```

NB : Une autre solution pourrait être de définir un autre constructeur de copie utilisant une référence, mais cela entraine une duplication de code.
On pourrait bien sûr déporter l'initialisation dans une méthode privée, mais cela nous priverait d'utiliser les listes d'initialisations.

Comment implémenter opérateur d'affectation ?

Auteur : nico-pyright(c)

Un opérateur d'affectation permet d'affecter un objet à un autre. Il est généré automatiquement en C++ natif, en C++/CLI il faut le spécifier manuellement.
Dans l'exemple précédent, l'affectation suivante :

```
Personne p1("nico", "pyright");
Personne p2("", "");
p2 = p1;
```

Aurait généré l'erreur de compilation C2582 ('operator =' function is unavailable).
Pour y remédier, on définit l'opérateur d'affectation, comme en C++ classique :

```
Personne% operator=(const Personne% p)
{
 if(%p == this)
  return *this;
 nom = p.nom;
 prenom = p.prenom;
 return *this;
}
```

Il ne faut bien sûr pas oublier de tester si on est pas en train d'affecter le même objet.

NB : L'opérateur d'affectation est visible uniquement par des clients C++/CLI, et non depuis des langages comme C# ou VB.Net.

On prendra garde de même à ne pas confondre avec l'affectation de handles vers les objets managés.

Sommaire > Le langage C++/CLI > Classes, Interfaces, héritage et types de données > Les types de données

Qu'est-ce qu'une classe managée ?

Auteur : nico-pyright(c)

Une classe managée veut simplement dire que l'objet est pris en charge par le CLR. Son allocation (avec gcnew) et sa désallocation (utilisant soit delete explicitement, soit utilisant le garbage collector) sont donc entièrement gérées (managed en anglais) par le CLR. Microsoft a essayé de rayer le mot managé de son vocabulaire pour éviter l'abus de langage (pour parler par exemple de C++/CLI plutôt que de C++ managé).

Cet objet est alloué sur le tas (heap) managé par l'intermédiaire de gcnew et est référencé grâce à un handle.

```
String ^ str = gcnew String("Ma chaîne managée est alloué sur le heap managé");
```

La désallocation est automatique grâce au garbage collector.

Evidement, les objets existants du framework .net (comme le String ci-dessus) sont managés.

lien : Comment allouer un objet managé avec gcnew ?
lien : Qu'est-ce que le garbage collector ?

Qu'est-ce qu'une classe non managée ?

Auteur : nico-pyright(c)

Pour déclarer une classe non managée par le CLR, on n'utilise aucun autre mot clé que class.

```
class CMaClasseNonManagee
{
 CMaClasseNonManagee(void);
 ~CMaClasseNonManagee(void);
 bool maFonctionNonManagee();
 CString maChaineNonManagee;
};
```

Cette ancienne déclaration fonctionne toujours de la même façon, et est gérée par le CRT, allouée sur le tas (non managé) avec new.

Une classe non managée par le CLR est dite native.

Qu'est-ce qu'un type de référence ?

Auteur : nico-pyright(c)

Un type ref est un type CLI qui est alloué sur le heap managé. Il est pris complètement en charge par le garbage collector qui gère sa désallocation quand il n'est plus utile.
Le mot clé ref permet de créer donc des objets managés ou des structures managées (on utilise alors ref class ou bien ref struct).

```
ref class CMaClasse
{
 CMaClasse(void);
 ~CMyClass(void);
 bool maFonction();
 String ^ monAttributChaine;
};
```

Qu'est-ce qu'un type de valeur ?

Auteur : nico-pyright(c)

Un type de valeur (value type) est un type CLI qui est alloué sur la pile et qui est détruit lorsqu'il sort de sa portée. Il se comporte essentiellement comme un type POD (Plain Old Data : un type POD est un type C++ qui possède un équivalent en C. Il correspond aux types composés à partir des types primitifs).
On ne peut pas hériter d'un type de valeur. Ils n'ont pas non plus de constructeur par défaut, de destructeur, d'opérateur de copie ou d'affectation.
Lorsque l'on conçoit un objet qui fonctionne comme un type natif (un int par exemple), alors cette classe est une candidate parfaite pour un type de valeur.
Pour créer un type de valeur, on utilise alors value class ou bien value struct.

```
value class MonTypeDeValeur {
 String^ uneChaine;
 String^ uneAutreChaine;
};
```

Remarque :

- **Les types de valeurs ont été créés pour permettre une copie rapide et efficace de données, sans utiliser d'indirections de mémoire inutile pour y accéder (pas besoin d'une place mémoire dans le garbage collector et d'une place mémoire pour le handle le référençant). Ainsi, le type de mémoire n'est pas stocké sur le heap managé et permet d'éviter l'encombrement du garbage collector. Ils bénéficient de même des optimisations de registre.**
- **Sans les types de valeurs, .Net aurait été beaucoup trop lent face à ses concurrents Java et C++, même pour des petits projets.**
  Cependant, il faut noter que lors de la phase d'optimisation du compilateur, celui-ci peut décider lui même si le type de valeur aurait été un meilleur choix qu'un type de référence et choisir ainsi le meilleur emplacement pour ce type, dans la plupart des cas.
- **On utilise l'expression "types managés" ou "objets managés" par abus de langage. Il est plus correct de parler de types CLI.**

Il est en général recommandé pour le CLS d'utiliser des structures pour les types de valeurs et des classes pour les types de références.

Quelles sont les équivalences des types natifs dans le framework .Net ?

Auteur : nico-pyright(c)

Vous trouverez dans ce tableau une équivalence des types natifs avec les types du framework .Net.

Types natifs C++	Types du framework .Net
bool	System::Boolean
signed char	System::SByte
unsigned char	System::Byte
wchar_t	System::Char
double et long double	System::Double
float	System::Single
int, signed int, long, et signed long	System::Int32
unsigned int et unsigned long	System::UInt32
__int64 et signed __int64	System::Int64
unsigned __int64	System::UInt64
short et signed short	System::Int16
unsigned short	System::UInt16
void	System::Void

Comment créer une énumération C++/CLI ?

Auteur : nico-pyright(c)

Les énumérations C++/CLI diffèrent sensiblement des énumérations du C++ ISO/ANSI. Elles correspondent par défaut à des entiers.
Dans la mesure où ces énumérations sont des types CLI, elles doivent être définies en global. Il est impossible d'avoir une définition locale.

```
enum class Jours{Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche};

Jours jeudi = Jours::Jeudi;
```

Il est possible de définir un type différent pour les énumérations, dans la liste suivante :
short, int, long, long long, signed char, char, unsigned short, unsigned int , unsigned long, unsigned long long, unsigned char et bool.
Ainsi, il est possible de définir une énumération de type bool :

```
enum class JoursDeLaSemaine : bool {Lundi = true, Mardi = true, Mercredi = true,
    Jeudi = true, Vendredi = true, Samedi = false, Dimanche = false};
```

Comment définir un literal ?

Auteur : nico-pyright(c)

On se sert d'un literal en C++/CLI comme on se servirait d'une variable static const en C++.
Cette variable doit être initialisée à sa déclaration et est une manière élégante de nommer des constantes.

```
ref struct constMath
{
 literal double pi = 3.14159265;
 literal double nombreOr = 1.61803399;
};

int main()
{
 static const int rayon = 5;
 double circonference = 2*rayon*constMath::pi;
 return 0;
}
```

Comment définir une variable initonly ?

Auteur : nico-pyright(c)

Une variable initonly fonctionne comme un literal à la différence qu'elle n'est pas obligée d'être initialisée à sa construction.
Cela veut dire qu'on peut par exemple initialiser cette variable dans un constructeur. Par contre, une fois initialisée, il sera impossible de la changer.

```
ref class constMath
{
public:
 initonly double pi;
 constMath(int x)
 {
  pi = x;
 }
};
```

Comment savoir si un handle est nul ?

Auteur : nico-pyright(c)

En C/C++, on utilise 0 ou null pour savoir si un pointeur est null.
En C++/CLI, lorsqu'on affecte 0 à un handle, il s'effectue une conversion (un boxing) en Int32 qui sera castée en Object ^.
Ainsi le C++/CLI nous offre un nouveau mot clé : nullptr.

```
MonObjet ^m = nullptr;
if (m == nullptr)
 ....
```

nullptr correspond aux nothing et null du VB.Net et du C#.

Sommaire > Le langage C++/CLI > Classes, Interfaces, héritage et types de données > Les types de données > Variables et fonctions statiques

Comment initialiser des variables static dans une classe ?

Auteur : nico-pyright(c)

Il existe plusieurs façons d'initialiser des variables statiques dans une classe. On peut tout d'abord le faire à l'initialisation :

```
public ref class MyStatic
{
public:
   static int ^ val1 =2;
};
```

C++/CLI permet d'utiliser également un constructeur statique, alors que le C++ ne l'autorise pas. C'est dans ce constructeur qu'on va s'occuper d'initialiser les variables statiques.
Ce constructeur devra être déclaré en private, et ne pourra pas posséder de paramètres. On le précédera par le mot clé static.

```
public ref class MyStatic
{
public:
   static int ^ val1;
private:
   static MyStatic()
   {
    val1 = 3;
   }
};
```

NB : Le constructeur statique est appelé par défaut par le CLR.

Comment déclarer une variable de manière globale ?

Auteurs : dev01 - nico-pyright(c)

Afin d'avoir une variable accessible de n'importe quel endroit de son application il faut la déclarer static. Cette pratique est en général à déconseiller aux débutants, mais peut s'avérer indispensable dans certaines utilisations.
En général, on créera une classe qui regroupe toutes les variables accessibles de n'importe quel endroit. Il peut s'agir d'une classe de configuration par exemple.

```
public ref class MyConfiguration
{
public:
   static property String ^StringConnection;
};
```

Exemple d'appel :

```
MyConfiguration::StringConnection = "ABC";
```

```
Console::WriteLine(MyConfiguration::StringConnection);
```

Comment déclarer une méthode globale ?

Auteur : nico-pyright(c)

Une méthode globale "façon C" ne respecte pas la norme CLS. On peut construire un équivalent en faisant une classe statique, non dérivable (grâce aux mots clés abstract et sealed):

```
public ref class ClassMethodeGlobale abstract sealed
{
public:
   static void faitQuelqueChose() {};
};
```

Sommaire > Le langage C++/CLI > Classes, Interfaces, héritage et types de données > Les interfaces

Qu'est-ce qu'une interface ?

Auteur : nico-pyright(c)

Une interface est une classe qui spécifie un certain nombre de fonctions qui devront être obligatoirement implémentées par les classes qui implémenteront cette interface.
Il s'agit ici de fournir une structure standardisée pour réaliser des fonctionnalités spécifiques.
Une interface n'implémente aucune des ses fonctions membres, c'est aux classes qui implémentent cette interface de le faire.

Par exemple, dans le framework .Net, pour comparer des objets, nous pouvons utiliser la fonction CompareTo(). Chaque objet qui peut être comparé implémente l'interface *IComparable* qui oblige l'écriture d'une fonction CompareTo().
Ainsi, chaque objet disposera du même mécanisme de comparaison à travers cette méthode.

Comment définir une interface ?

Auteur : nico-pyright(c)

Pour définir une interface, on utilise les mots clés : interface class ou interface struct.
Tous les membres d'une interface sont publics et ne peuvent pas être autrement. Ces membres peuvent être des fonctions, des opérateurs de fonctions, des propriétés, des champs statics ou bien des événements *(event)*.
Il est également possible de spécifier un constructeur static ou de contenir des classes *Nested* (i.e. des classes déclarées à l'intérieur d'autres classes). Cependant, nonobstant ces possibilités, les interfaces sont en général très simples.

```
interface class IDessinable
{
 void DessineToi();
};
```

Comment implémenter une interface ?

Auteur : nico-pyright(c)

Une classe peut implémenter une ou plusieurs interface, la notation étant la même que pour dériver une classe.
Ainsi, pour implémenter l'interface IComparable, il faut surcharger la méthode CompareTo :

```
ref class x : IComparable
{
 int a;
 int b;
public:
 virtual int CompareTo(Object^ obj)
 {
  x ^x1 = dynamic_cast<x ^>(obj);
  if (x1)
  {
```

```
    if (a+b < x1->a + x1->b)
     return -1;
    if (a+b > x1->a + x1->b)
     return 1;
    return 0;
   }
   else
    throw gcnew Exception("L'objet n'est pas de type x");
  }
 };
```

Comment savoir si un type implémente une interface donnée ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Grâce à la réflection, il est possible d'obtenir énormément d'information à partir du type d'une classe. Pour vérifier si une classe implémente une interface, il vous suffira d'utiliser la méthode GetInterface de la classe Type :

```
interface class IMonInterface
{ };

ref class Test : IMonInterface
{ };

int main()
{
 Type ^objType = Test::typeid;
 Type ^objTypeInterface = objType->GetInterface("IMonInterface", false);
 if (objTypeInterface != nullptr)
  Console::WriteLine("La classe 'Test' implémente l'interface 'IMonInterface' !");
 else
  Console::WriteLine("La classe 'Test' n'implémente pas l'interface 'IMonInterface' !");
 return 0;
}
```

lien : System.Type

Sommaire > Le langage C++/CLI > La surcharge d'opérateur

Qu'est ce que la surcharge d'opérateur en C++/CLI ?

Auteur : nico-pyright(c)

La surcharge d'opérateur du C++/CLI part du même principe que pour le C++. Elle permet de tirer parti de l'intuition des utilisateurs de la classe. L'utilisateur va en effet pouvoir écrire son code en s'exprimant dans le langage du domaine plutôt que dans celui de la machine.
Cependant, il y a quelques différences. La différence majeure vient de la capacité du framework .Net d'être utilisé à travers plusieurs langages. Ainsi, cet aspect impose que les opérateurs de surcharge en C++/CLI soient déclarés en static.
Remarque : On ne peut pas surcharger l'opérateur gcnew, alors qu'on peut surcharger new en C++ natif. C'est également le cas pour les opérateurs delete, new, [] et ().

lien : http://cpp.developpez.com/faq/cpp/?page=surcharge/

Pourquoi utiliser la surcharge d'opérateur en C++/CLI plutôt qu'en C++ natif ?

Auteur : nico-pyright(c)

Etant donnée que l'on peut utiliser la surcharge d'opérateur en C++ natif, pourquoi utiliser la surcharge du C++/CLI ?
Cette question revient à se poser la question de pourquoi utiliser une classe référence plutôt qu'une classe C++.
Il y a bien sur plusieurs raisons pour un tel choix, citons par exemple la gestion de la classe par le garbage collector ou la possibilité d'utiliser cette classe depuis un autre langage .Net (C#, VB.Net, etc ...).
Si l'on choisit d'implémenter une classe référence, on est obligé d'utiliser la surcharge d'opérateur du C++/CLI. Et comme une classe ref ne supporte pas les fonctions amies *(friend)*, il devient indispensable d'utiliser des fonctions statiques pour les opérateurs binaires.

Quelques exemples de surcharge d'opérateurs

Auteur : nico-pyright(c)

J'ai écrit une petite classe pour l'occasion qui me permet d'illustrer la surcharge de différents opérateurs. (Cette classe n'est bien sur pas complète et ne gère pas non plus les erreurs).
Il s'agit d'une classe Temps, composée d'un entier heure, d'un entier minute et d'un entier seconde. Vous l'aurez compris, nous allons additionner des Temps.
Pour cet exemple, nous surchargeons l'opérateur unaire + qui additionne deux Temps, puis les opérateurs binaires + qui additionnent un Temps et un entier représentant un nombre de secondes (et inversement). Il s'agira ensuite d'implémenter l'opérateur ++ qui ajoutera simplement une seconde à notre Temps.

```
ref class Temps
{
private:
 int heure;
 int min;
 int sec;
public:
 Temps(int h, int m, int s)
 {
  heure = h;
  min = m;
  sec = s;
 }
```

```
 virtual String^ ToString() override
 {
  return "Il est " + heure + ":" + min + ":" + sec;
 }

 Temps^ operator+(Temps^ t)
 {
  int s = (sec + t->sec)`;
  int m = (min + t->min + (sec + t->sec)/60)`;
  int h = (heure + t->heure + (min + t->min + (sec + t->sec)/60)/60)$;
  return gcnew Temps(h, m, s);
 }
 static Temps^ operator+(Temps^ t, int sec);
 static Temps^ operator+(int sec, Temps^ t);
 static Temps^ operator++(Temps^ t);
};

Temps^ Temps::operator+(Temps ^ t, int sec)
{
 int s = (sec + t->sec)`;
 int m = (t->min + (sec + t->sec)/60)`;
 int h = (t->heure + (t->min + (sec + t->sec)/60)/60)$;
 return gcnew Temps(h, m, s);
}
Temps^ Temps::operator+(int sec, Temps ^ t)
{
 return t+sec;
}
Temps^ Temps::operator++(Temps^ t)
{
 return t+1;
}

int main(array<System::String ^> ^args)
{
 Temps ^t1 = gcnew Temps(3,20, 40);
 Temps ^t2 = gcnew Temps(4,50, 20);
 Console::WriteLine(t1+t2);
 Console::WriteLine(t1+10);
 Console::WriteLine(90+t1);
 Console::WriteLine(++t1);
 Console::WriteLine(t1);
}
```

Remarques

- **Nous sommes obligés d'implémenter une surcharge avec des méthodes statiques pour les opérateurs + avec un entier et l'opérateur ++. (en effet, une classe ref ne supporte pas les classes amies *(friend)*).**
  Notez aussi que l'on définit une seule fois l'opérateur ++ et que nous pouvons écrire t1++ ou ++t1.
- **On implémente la surcharge de la même façon pour une classe référence ou pour une classe de valeur. La seule différence vient du fait que dans un cas, on utilise des handles ^.**

Comment implémenter une propriété (get/set) ?

Auteur : nico-pyright(c)

Pour implémenter une propriété (un accesseur), on utilise le mot clé property. On peut soit l'avoir en lecture seule uniquement (implémentation de get uniquement), en écriture seule (implémentation de set uniquement) ou en lecture/écriture (implémentation de get et de set).

```
ref class Personne
{
private:
 String ^_nom;
 String ^_prenom;
 int ^_age;
public:
 Personne(String ^nom, String ^prenom, int age) : _nom(nom), _prenom(prenom), _age(age) {}
 property String ^Nom
 {
  String ^ get() { return _nom; }
 }
 property String ^Prenom
 {
  String ^ get() { return _prenom; }
  void set(String ^value) { _prenom = value; }
 }
 property int Age
 {
  void set(int value) { _age = value; }
 }
 virtual String ^ ToString() override
 {
  return String::Format("Je m'appelle {0} {1} et j'ai {2} ans", _prenom, _nom, _age);
 }
};

int main()
{
 Personne ^ moi = gcnew Personne("Pyright", "", 15);
 moi->Prenom = "Nico";
 moi->Age = 28;
 // int a = moi->Age;  => error C2039: 'get' : is not a member of 'Personne::Age'
 Console::WriteLine(moi->Nom);
 Console::WriteLine(moi);
}
```

Une manière très simple de simplifier la propriété en lecture et écrire Prenom ci-dessus (qui affecte et retourne un membre privé) est de la remplacer par cette écriture :

```
property String ^Prenom;
```

Comment implémenter un opérateur d'indexation ?

Auteur : nico-pyright(c)

Il s'agit de créer une classe qui contient une propriété d'indexation par défaut :

```
ref class Jours
{
private:
 array<String^>^ lesJours;
public:
 Jours(...array<String^>^ j) : lesJours(j) {}

 property int Length
 {
  int get()
  {
   return lesJours->Length;
  }
 }
```

```
 property String^ default[int]
 {
  String^ get(int index)
  {
   if(index >= lesJours->Length)
    throw gcnew Exception("Index en dehors des limites");
   return lesJours[index];
  }
  void set(int index, String^ elt)
  {
   if(index >= lesJours->Length)
    throw gcnew Exception("Index en dehors des limites");
   lesJours[index] = elt;
  }
 }
};

int main(array<System::String ^> ^args)
{
 Jours ^ lesjours =
 gcnew Jours("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimance");

 lesjours[3] = "Jeudi c'est permis";
 for(int i = 0 ; i < lesjours->Length; i++)
  Console::WriteLine("Le jour {0} est {1}", i+1, lesjours[i]);
}
```

Sommaire > Le langage C++/CLI > Les types et fonctions génériques

Qu'est-ce qu'un générique ?

Auteur : nico-pyright(c)

Les génériques du C++/CLI (mot clé generic) ressemblent beaucoup aux templates du C++.
Ils sont quand même subtilement différents dans la mesure où les génériques CLI sont instanciés par le VES (Virtual Execution System) au moment de l'exécution plutôt qu'à la compilation pour les templates.
Une définition générique doit être :

- **une classe référence *(ref class)*.**
- **une classe de valeur *(value class)*.**
- **une interface *(interface class)*.**
- **un délégate.**
- **Ou enfin une fonction.**

lien : http://cpp.developpez.com/faq/cpp/?page=templates

Pourquoi utiliser un generic ?

Auteur : nico-pyright(c)

Le C++ supporte les templates, pourquoi choisir les génériques ?

Pour pouvoir choisir, il est important de connaître certaines des différences entre ceux-ci.
Les templates sont instanciés à la compilation alors que les generics sont instanciés au moment de l'exécution. Ce qui implique que l'on peut spécialiser un generic dans un autre assembly, ce que l'on ne pourra pas faire avec un template. En effet ces derniers au moment de l'exécution ne sont plus des types paramétrables. Ainsi, deux generics spécialisés dans deux assemblys différentes avec le même type d'arguments correspondront au même type. A contrario de deux templates qui seront considérés comme deux types différents.
Cela s'explique parce que les generics de type référence génèrent une seule portion de code valable pour tous les types d'arguments. Les tempates (ainsi que les generic de type de valeur) généreront une portion de code pour chaque spécialisation.
Le compilateur est aussi capable d'optimiser les generics en fonction des types en paramètres.
En revanche les templates pourront supporter des templates de templates en paramètres, alors que les generics ne le pourront pas.

```
template<template<class T> class X> class MyClass
```

Comment créer une fonction générique ?

Auteur : nico-pyright(c)

Pour créer une fonction générique, on utilise le mot clé generic sur la première ligne pour indiquer que nous faisons une définition générique.
Le mot clé typename entre crochets <> indique que le type du paramètre est T dans la fonction générique. Ce type sera remplacé par le type utilisé au moment de l'utilisation de la fonction.
On peut utiliser plusieurs paramètres en séparant les types dans les crochets par des virgules.
Imaginons que nous voulions faire une fonction qui énumère des éléments d'un tableau CLI :

```
generic<typename T>
void afficheTableau(array<T> ^ tab)
{
 for each(T i in tab)
  if (i)
   Console::Write(i->ToString() + " ");
 Console::WriteLine();
}
```

On pourra appeler cette fonction par exemple de la sorte :

```
array<String^> ^s = {"1", "2", "4", "6", "8"};
afficheTableau(s);
array<int> ^i = {1, 2, 4, 6, 8};
afficheTableau(i);
```

Cela peut aussi fonctionner pour une classe perso, dans la mesure où celle-ci implémente dans notre cas la méthode ToString() :

```
ref class Personne
{
private:
 String ^ nom;
 String ^ prenom;
public:
 Personne(String ^n, String ^p)
 {
  nom = n;
  prenom = p;
 }
 virtual String ^ ToString() override
 {
  return nom + " " + prenom;
 }
};

array<Personne^> ^p = gcnew array<Personne ^>(3);
p[0] = gcnew Personne("nico", "pyright(c)");
p[1] = gcnew Personne("Herb", "Sutter");
afficheTableau(p);
```

Remarque : Au niveau de la syntaxe il est aussi bien correct d'écrire :

```
array<int> ^i = {1, 2, 4, 6, 8};
afficheTableau<int>(i);
```

que

```
array<int> ^i = {1, 2, 4, 6, 8};
afficheTableau(i);
```

Dans le second cas, le type du paramètre est déduit à partir de i.

Comment définir une classe générique ?

Auteur : nico-pyright(c)

Voici un exemple de classe générique, qui reproduit le fonctionnement (basique) d'une pile :

```
generic<typename T>
public ref class Pile
{
 array<T>^ elements;
 int curElement;
public:
 Pile(int taille)
 {
  elements = gcnew array<T>(taille);
  curElement = 0;
 }
 void Push(T elt)
 {
  if (curElement > elements->Length -1)
   throw gcnew Exception("Il n'y a plus de place dans la pile");
  else
  {
   elements[curElement] = elt;
   curElement++;
  }
 }
 T Pop()
 {
  if (curElement>0)
  {
   curElement--;
   return elements[curElement];
  }
  throw gcnew Exception("Il n'y a plus d'éléments dans la pile");
 }
};
```

Pour l'utiliser, il suffit de l'instancier ainsi :

```
 Pile<int>^ p = gcnew Pile<int>(10);
 try
 {
  p->Push(1);
  p->Push(2);
  p->Push(3);
  Console::WriteLine(p->Pop());
  Console::WriteLine(p->Pop());
  p->Push(4);
  Console::WriteLine(p->Pop());
  Console::WriteLine(p->Pop());
 }
 catch (String ^e)
 {
  Console::WriteLine(e);
 }
```

Pour utiliser la pile avec un autre type que int, il suffirait de l'instancier ainsi :

```
Pile<MonObjet^>^ p = gcnew Pile<MonObjet^>(10);
p->Push(gcnew MonObjet);
MonObjet^ m = p->Pop();
```

Comment créer une fonction générique avec une contrainte ?

Auteur : nico-pyright(c)

On procède de la façon que pour Comment créer une fonction générique ?, en utilisant le mot clé where. La contrainte se fait sur l'utilisation d'une interface.
Dans notre exemple, nous n'accepterons uniquement que des types en premier paramètre qui implèmentent l'interface IEnumerable (notamment le for each) et qui implémentent IComparable pour le deuxième paramètre (notamment la méthode Equals) :

```
generic<typename T1, typename T2> where T1:IEnumerable where T2:IComparable
int donnePosition(T1 x, T2 y)
{
 int i=0;
 for each(T2 t in x)
 {
  if(y->Equals(t))
   return i;
  i++;
 }
 return -1;
}
```

Cette fonction nous retourne la position d'un élément dans un tableau. On pourra l'appeler ainsi :

```
array<int> ^a = {4, 5, 11, 2, 3};
Console::WriteLine(donnePosition(a,11));
array<String ^> ^s = {"un", "deux", "trois", "quatre", "cinq"};
Console::WriteLine(donnePosition(s,"cinq"));

ArrayList ^g = gcnew ArrayList();
g->Add("un");
g->Add("deux");
Console::WriteLine(donnePosition(g,"un"));
```

Vous pouvez constater que cette fonction marche aussi bien pour des array CLI que des ArrayList .Net qui implémentent tous les deux IEnumerable.
Remarque : On obtient une erreur de compilation si les types ne respectent pas les contraintes :

```
'monObjet ^' : invalid type argument for generic parameter 'T2' of generic 'donnePosition', does not
 meet constraint 'System::IComparable ^'
```

Sommaire > Le langage C++/CLI > Compatibilité d'assignation et comparaison de d'identité de types

Comment implémenter l'opérateur d'assignation 'as' du C# ?

Auteur : nico-pyright(c)

L'opérateur as du C# n'existe pas en C++/CLI. Il est utilisé pour effectuer des conversions entre des types compatibles.
Il retourne nullptr lorsque la conversion est impossible.
En C++, nous allons donc utiliser l'opérateur dynamic_cast pour effectuer une telle tâche.
Ainsi, il suffira d'utiliser le code suivant pour simuler le as du C# (ici T est un type quelconque).

```
T t = dynamic_cast<T>(e); // cette ligne uniquement correspond au 'as'
if (t != nullptr)
{
    // ...
}
```

On peut rendre ceci générique en utilisant les templates du C++. Il est impossible ici d'utiliser les génériques du C++/CLI, car on ne peut pas combiner l'utilisation de dynamic_cast avec ceux-ci.
Voici ce que serait l'implémentation du 'as' C# :

```
template < class T, class U >
T as(U u)
{
   return dynamic_cast< T >(u);
}
```

Observons maintenant l'exemple suivant pour illuster cette fonction :

```
ref class class1 {};
ref class class2 {};

int main()
{
 array<Object ^> ^tabObj = gcnew array<Object ^>(6);
 tabObj[0] = gcnew class1();
 tabObj[1] = gcnew class2();
 tabObj[2] = "chaine";
 tabObj[3] = 123;
 tabObj[4] = 123.4;
 tabObj[5] = nullptr;

 for (int i = 0; i < tabObj->Length ; i++)
 {
  String ^ s = as<String ^>(tabObj[i]);
  Console::Write("{0} : ", i);
  if (s)
   Console::WriteLine("'{0}'", s);
  else
   Console::WriteLine("n'est pas un String ^");
 }

 return 0;
}
```

Cet exemple produit :

```
0 : n'est pas un String ^
1 : n'est pas un String ^
2 : 'chaine'
3 : n'est pas un String ^
4 : n'est pas un String ^
5 : n'est pas un String ^
```

Comment implémenter l'opérateur 'is' du C# ?

Auteur : nico-pyright(c)

L'opérateur is du C# n'existe pas en C++/CLI. Il permet de vérifier si un objet est compatible avec un type. Il effectue cette vérification en runtime.
L'opérateur C++ dynamic_cast est le candidat parfait pour cette vérification.
Il suffit, sur le même principe que Comment implémenter l'opérateur d'assignation 'as' du C# ?, d'utiliser ce code (T étant un type quelconque) :

```
T t = dynamic_cast<T>(e);
if (t != nullptr)
{
    // ...
}
```

Le 'is' consiste en la combinaison des deux instructions ci-dessus.
On peut en faire une version générique, en utilisant les templates du C++. (En effet, on ne peut pas utiliser les génériques du C++/CLI à cause de dynamic_cast).

```
template < class T, class U >
Boolean is(U u)
{
   return dynamic_cast< T >(u) != nullptr;
}
```

Observons maintenant l'exemple suivant pour illuster cette fonction :

```
void Test(Object ^o)
{
 if (is<String ^>(o))
 {
  Console::WriteLine("o est une chaine");
  String ^a = safe_cast<String ^>(o);
  DoSomethingWith(a);
 }
 else if (is<Boolean ^>(o))
 {
  Console::WriteLine("o est un Boolean");
  Boolean ^ b = safe_cast<Boolean ^>(o);
  DoSomethingWith(b);
 }
 else
  Console::WriteLine("o n'est ni une chaine , ni un Boolean");
```

```
}

int main()
{
 String ^s = gcnew String("chaine");
 Boolean ^a = gcnew Boolean(true);
 Test(s);
 Test(a);
 Test(1234);

 return 0;
}
```

On obtient en sortie :

```
o est une chaine
o est un Boolean
o n'est ni une chaine , ni un Boolean
```

Comment effectuer une comparaison d'identité ?

Auteur : nico-pyright(c)

Nous avons vu dans Comment implémenter l'opérateur d'assignation 'as' du C# ? et dans Comment implémenter l'opérateur 'is' du C# ? comment effectuer une comparaison d'assignation.
Il peut être utile dans certains cas d'effectuer une comparaison d'identité.
Ces deux concepts sont différents dans la mesure où dans la comparaison d'assignation, on va vérifier si les types sont compatibles entre eux alors que dans la comparaison d'identité, on va vérifier si les types sont de même type.
On utilise pour ca la capacité du framework .Net de reflexion (d'introspection), en comparant le type d'un objet (méthode GetType) et le type d'un Type (méthode statique typeid).
Créons donc une fonction générique en C++/CLI pour cette comparaison :

```
generic <typename T, typename U>
bool isIdentity(U u)
{
 return T::typeid == u->GetType();
}
```

Nous allons illustrer cette fonction à travers l'exemple suivant, qui consiste à remettre à zéro tous les contrôles de type "textbox" dans une Winform, en parcourant tous les contrôles et en testant l'identité de ceux-ci :

```
for each (Control ^c in this->Controls)
 if (isIdentity<TextBox ^>(c))
  c->Text = "";
```

Comment ajouter une form .Net (winform) à mon application MFC ?

Auteur : nico-pyright(c)

Il s'agit tout d'abord de transformer son application MFC en application managée, pour ceci, il faut ajouter le support du CLR.
Bouton droit sur le projet -> Common properties -> general -> Common language runtime support ; Mettre à /clr (common language runtime support).

Ensuite ajoutez simplement une nouvelle winform, rajoutez des composants dessus, etc ...
Instanciez votre nouvelle form :

```
mfcPlusWinforms::mfcWinForm dlg;
dlg.ShowDialog(); // pour qu'elle se comporte comme une dialog
```

N'oubliez pas bien sur d'inclure le fichier .h correspondant au code de la winform.
Téléchargez le programme d'exemple

Comment utiliser un contrôle standard .Net dans une application MFC ?

Auteur : nico-pyright(c)

Il s'agit tout d'abord de transformer son application MFC en application managée, pour ceci, il faut ajouter le support du CLR.
Bouton droit sur le projet -> Common properties -> general -> Common language runtime support ; Mettre à /clr (common language runtime support).

Ensuite, pour ajouter un contrôle .Net, il vous faut d'abord ajouter un static MFC classique où vous voudrez positionner votre contrôle .Net
Ajouter ensuite l'include dans stdafx.h :

```
#include <afxWinForms.h>
```

On utilise ensuite l'objet CWinFormsControl pour instancier une donnée membre de notre contrôle.
Par exemple, si je veux utiliser le contrôle LinkLabel :

```
Microsoft::VisualC::MFC::CWinFormsControl< System::Windows::Forms::LinkLabel >m_linkLabelDotNet;
```

Ensuite, dans le DoDataExchange :

```
DDX_ManagedControl( pDX, IDC_DOTNET, m_linkLabelDotNet );
```

Rajouter ensuite un handler d'événement :

```
void linkLabel_LinkClicked( System::Object^
 sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e );
```

et utiliser la macro _DELEGATE_MAP pour cabler les méthodes :

```
BEGIN_DELEGATE_MAP( CMfcControlDotNetDlg )

  EVENT_DELEGATE_ENTRY( linkLabel_LinkClicked, System::Object ^, System::Windows::Forms::LinkLabelLinkClickedEve
 )
END_DELEGATE_MAP()
```

On crée le délégate à l'initialisation du composant :

```
m_linkLabelDotNet->LinkClicked += MAKE_DELEGATE(System::Windows::Forms::LinkLabelLinkClickedEventHandler ,
 linkLabel_LinkClicked);
```

Et vous pouvez ainsi désormais utiliser la fonction :

```
void CMfcControlDotNetDlg::linkLabel_LinkClicked(System::Object ^sender, System::Windows::Forms::LinkLabelLinkCl
{
    System::Windows::Forms::MessageBox::Show(L"Je capte l'événement click");
}
```

pour capter l'événement du click.
Téléchargez le programme d'exemple

Comment ajouter un contrôle utilisateur .Net (UserControl) à mon application MFC ?

Auteur : nico-pyright(c)

On procède de la même façon que pour Comment utiliser un contrôle standard .Net dans une application MFC ?, à la différence près que l'on passe le type de l'usercontrol au CWinFormsControl.

```
Microsoft::VisualC::MFC::CWinFormsControl< MonUserControlNamespace::MonUserControl >m_monControle;
```

Remarque : N'oubliez pas de rajouter une référence à la DLL UserControl.

lien : Comment utiliser un contrôle standard .Net dans une application MFC ?

Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Auteur : nico-pyright(c)

Il s'agit tout d'abord de transformer son application MFC en application managée, pour ceci, il faut ajouter le support du CLR.
Bouton droit sur le projet -> Common properties -> general -> Common language runtime support ; Mettre à /clr (common language runtime support).

Ajouter ensuite une Winform à votre projet. La form, pour pouvoir fonctionner en tant que vue, doit hériter de UserControl. Changer alors la définition de la classe, pour qu'elle ressemble à la ligne suivante :

```
public ref class WinformView : public System::Windows::Forms::UserControl
```

N'oubliez pas de rajouter l'include dans stdafx.h :

```
#include <afxWinForms.h>
```

Modifier le fichier .h de la vue pour faire hériter la classe, non plus de CView, mais de Microsoft::VisualC::MFC::CWinFormsView

```
class CMfcWinformAsViewView : public Microsoft::VisualC::MFC::CWinFormsView
```

Il faut ensuite modifier le .cpp pour rester cohérent avec ce nouvel héritage. Changer la définition de l'IMPLEMENT_DYNCREATE ainsi que du BEGIN_MESSAGE_MAP :

```
IMPLEMENT_DYNCREATE(CMfcWinformAsViewView, Microsoft::VisualC::MFC::CWinFormsView)
BEGIN_MESSAGE_MAP(CMfcWinformAsViewView, Microsoft::VisualC::MFC::CWinFormsView)
```

Il ne reste plus qu'à appeler le constructeur de CWinformsView dans le constructeur de notre vue, en lui passant en paramètre le type de notre form :

```
CMfcWinformAsViewView::CMfcWinformAsViewView():Microsoft::VisualC::MFC::CWinFormsView(MfcWinformAsView::WinformV
{
}
```

Compiler et exécuter : la winform est utilisée comme vue, les événements de la winform fonctionnant indépendamment.

Comment communiquer entre le document MFC et la vue Winform dans mon application MFC ?

Auteur : nico-pyright(c)

En préambule, vous devez avoir défini une Winform en tant que vue MFC : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Il s'agit ici simplement de faire passer le pointeur de document à la Winform.
Rajoutez un membre public à votre classe Winform du type de votre Document (n'oubliez pas d'inclure le .h correspondant) :

```
public:
 CMfcWinformAsViewDoc *pDoc;
```

Il faut maintenant initialiser ce pointeur à partir du document courant MFC, on peut le faire par exemple dans le OnInitialUpdate de la vue. *Pour ajouter cette méthode, positionnez vous sur le .h de la vue, cliquez sur le bouton "overrides" de la fenêtre de propriétés et générer la méthode OnInitialUpdate.* On utilisera alors la méthode GetControl pour récupérer un handle sur la Winform, puis on initialise le membre public de la Winform avec le document courant (méthode GetDocument) :

```
void CMfcWinformAsViewView::OnInitialUpdate()
{
 CWinFormsView::OnInitialUpdate();

 MfcWinformAsView::WinformView ^ viewWinform =
 safe_cast<MfcWinformAsView::WinformView ^>(GetControl());
 viewWinform->pDoc = GetDocument();
}
```

**Vous pouvez désormais manipuler les données du document depuis votre winform.
Pour l'exemple, j'ai rajouté une méthode dans ma classe document qui renvoit l'heure courante :**

```
public:
 CTime GetCurrentTime() { return CTime::GetCurrentTime();}
```

Je peux alors récupérer l'heure issue de mon document MFC dans ma Winform :

```
textBox2->Text =  gcnew String(pDoc->GetCurrentTime().Format("%A, %B %d, %Y / %H:%M:%S"));
```

Téléchargez le programme d'exemple

lien : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Comment intercepter les événements d'une vue Winform dans mon application MFC ?

Auteur : nico-pyright(c)

En préambule, vous devez avoir défini une Winform en tant que vue MFC : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Il peut être intéressant de capter les événements classiques d'une vue dans notre winform CWinFormsView. Pour ceci, il faut implémenter l'interface Microsoft::VisualC::MFC::IView au niveau de la classe de la Winform. Ceci donne :

```
public ref class
 WinformView : public System::Windows::Forms::UserControl, Microsoft::VisualC::MFC::IView
```

Pour implémenter cette interface, on doit surcharger les trois fonctions virtuelles publiques OnUpdate, OnActivateView et OnInitialUpdate.

```
virtual void OnUpdate()
{
}
virtual void OnActivateView(bool activate)
{
}
virtual void OnInitialUpdate()
{
 textBox3->Text = textBox3->Text + "Passage dans OnInitialUpdate\r\n";
}
```

Notez ici que j'ai simplement surchargé une seule de ces méthodes, mais les 3 fonctions doivent être obligatoirement définies.
Voilà, c'est fini.
Remarque : Notez dans mon exemple, que c'est la méthode OnInitialUpdate de la vue MFC qui est appelée avant la méthode OnInitialUpdate de la vue Winform. En effet, cette dernière est appelée avec CWinFormsView::OnInitialUpdate().

```
void CMfcWinformAsViewView::OnInitialUpdate()
{
 CWinFormsView::OnInitialUpdate(); // la méthode OnInitialUpdate de la winform est invoquée ici
}
```

lien : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Comment intercepter les événements MFC dans ma vue Winform ?

Auteur : nico-pyright(c)

En préambule, vous devez avoir défini une Winform en tant que vue MFC : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Il peut être intéressant de capter les événements MFC dans notre winform CWinFormsView, par exemple un clic sur un élément du menu principal.
Pour ceci, il faut implémenter l'interface Microsoft::VisualC::MFC::ICommandTarget au niveau de la classe de la Winform, qui va nous permettre de mapper les événements à l'intérieur de notre Winform. Ceci donne :

```
public ref class
 WinformView : public System::Windows::Forms::UserControl, Microsoft::VisualC::MFC::ICommandTarget
```

Pour implémenter cette interface, il y a une seule méthodes virtuelle publique à surcharger : il s'agit de la méthode Initialize.

```
public:
 Microsoft::VisualC::MFC::ICommandSource ^ cmdSrc;
 virtual void Initialize(Microsoft::VisualC::MFC::ICommandSource ^cmdSource)
 {
  cmdSrc = cmdSource;
  cmdSrc->AddCommandHandler(ID_APP_EXIT,
 gcnew Microsoft::VisualC::MFC::CommandHandler(this, &WinformView::OnExit));
 }
 void OnExit(unsigned int id)
 {
```

```
    MessageBox::Show("Click sur menu Exit");
  }
```

On définit la méthode Initialize qui prend un ICommandSource en paramètre. Ce ICommandSource est stocké en tant que membre de la classe.
Il faut maintenant effectuer le mapping à proprement dit. On utilise la méthode AddCommandHandler. On lui passe en paramètre l'ID de la ressource (du menu dans l'exemple) à intercepter.
On lui passe aussi un pointeur sur la méthode qui va être appelée lors du click sur l'élément du menu.
Et ici, on affiche un MessageBox lors du clic sur le menu. Voilà, c'est fini.
Remarque : Notez dans mon exemple, qu'on ne peut plus quitter par le menu, il faudra quitter par la croix ; ceci montre que l'interception du message est remplacée, on ne pourra pas traiter ce message dans la classe d'application.

Téléchargez le programme d'exemple

lien : Comment utiliser une Winform (UserControl) en tant que vue dans mon application MFC ?

Comment gérer la cohabitation de deux fonctions ayant le même nom dans l'API Win32 et le framework.net 1.x ?

Auteur : nico-pyright(c)

Ceci est valable pour un projet C++ utilisant les extensions managées du framework .Net 1.x :
Lorsqu'on migre petit à petit son projet en C++.Net ou simplement lorsqu'on veut faire cohabiter l'Api Win32 (ou les MFC) et le framework.Net, on peut rencontrer des problèmes de compilation du style :

```
error C2039: 'GetObjectA' : is not a member of 'System::Resources::ResourceManager'

error C2653: 'MessageBoxA' : is not a class or namespace name
```

Le problème vient du faire que lorsqu'on souhaite utiliser la fonction MessageBox par exemple, le compilateur ne sait pas s'il doit utiliser la fonction messageBox définie dans Windows.h ou bien dans le namespace System::Windows::Forms.
Il faut donc à ce moment dire au compilateur quelle fonction on veut utiliser.
On enlève alors la définition du messageBox comme ci-dessous :

```
#pragma push_macro("MessageBox")
#undef MessageBox

...
MessageBox::Show(S"hello");
...

#pragma pop_macro("MessageBox")
```

Ainsi le compilateur saura qu'il faut utiliser le MessageBox de .Net.
Remarque : Avec IJW et le compilateur Visual C++ 2005, une simple utilisation de l'opérateur de résolution de portée suffit :

```
::MessageBox(NULL, "Hello", "", MB_ICONSTOP);
```

Comment éviter la redéfinition des objets COM ?

Auteur : nico-pyright(c)

Lorsqu'on inclut windows.h, il arrive fréquemment qu'on ait une erreur de ce style :

```
'IDataObject' : ambiguous symbol error
```

Ce problème vient de cet include, qui définit lui aussi un IDataObject, qui est une interface COM.
Il faut définir WIN32_LEAN_AND_MEAN, qui a pour but d'exclure tous les entêtes Win32 qui ne sont pas utiles.
On rajoute donc avant notre premier include :

```
#define WIN32_LEAN_AND_MEAN
```

Comment compiler du code natif ?

Auteur : nico-pyright(c)

Lorsque l'on développe une application "managée", le compilateur génère du MSIL et du langage machine lorsqu'il ne peut pas générer de MSIL (j'entends par langage machine du code x86/x64/IA64).
Il peut être intéressant de forcer le compilateur à générer du langage machine pour certaines portions de code. Cela est très utilisé lorsque vous souhaitez porter votre application Win32 petit à petit en .Net ou lorsque vous souhaitez profiter de la couche la plus basse possible, pour des portions de code critiques par exemple.
Dans ces portions de code, il sera bien sur impossible d'accéder au CLR, car le compilateur ne génère pas de MSIL.

On utilise alors les pragmas :

```
#pragma unmanaged
```

et

```
#pragma managed
```

Il suffit alors d'entourer du code par ces deux pragmas pour qu'il soit généré en langage machine. Cela implique donc bien sur d'utiliser le mode de compilation mixte (/clr).

```
#pragma unmanaged

#include <windows.h>
#pragma comment(lib, "User32.lib")
```

```
class CNonManagee
{
public:
 CNonManagee(void);
 ~CNonManagee(void);
 void Show()
 {
  MessageBoxW(NULL, L"Message depuis le langage machine", L"", 0);
 }
};

#pragma managed
```

lien : Voir l'article complet

Que faire avec l'erreur d'éditions de liens LNK1313 ?

Auteur : nico-pyright(c)

Lorsqu'on essaie de mixer du code natif et du code pure (/clr:pure) on obtient une erreur de link :

```
ijw/native module detected; cannot link with pure modules
```

Ceci n'est pas autorisé. Il faut alors passer le mode de compilation à mixte (/clr) pour résoudre cette erreur.

lien : Compilation avec support mixte du CLR (/clr)

Que faire avec l'erreur d'édition de liens LNK2031 ?

Auteur : nico-pyright(c)

Cette erreur apparait dans certains cas précis, lorsqu'on essaie d'utiliser des fonctions dont la convention d'appel n'a pas été précisée explicitement.
On obtient l'erreur de link LNK2031 :

```
error LNK2031: unable to generate p/invoke for "extern "C" int __clrcall ...
```

Une première solution est de faire l'édition de lien en mode mixte, donc de ne pas utiliser le mode /clr:pure, mais le mode de compilation /clr.
Une deuxième solution est de définir précisément les prototypes pour faire concorder les conventions d'appels (ce qui n'est pas toujours évident lors de l'utilisation d'une bibliothèque tierce). En effet, les conventions implicites ne sont pas les mêmes pour le monde natif et le monde managé.
Enfin, une troisième solution est d'utiliser DLLImport au lieu d'une édition de liens mixte.

Exemple, une bibliothèque définie ainsi :

code natif

```
__declspec(dllexport) int addition(int a,int b)
{
 return a+b;
}
```

a une convention d'appel implicite. Par défaut, à la création d'un projet, il s'agit de __cdecl.
Ainsi, lorsqu'on prototype la fonction dans une application managée, si on fait :

code managé

```
#pragma comment (lib, "testDll.lib")
__declspec(dllimport) int addition(int a,int b);
```

on utilisera la convention d'appel implicite __clrcall. D'où l'erreur.
En mode de compilation /clr, la convention d'appel implicite n'est pas la même qu'en compilation pure.
En cas d'oubli de spécification précise de la convention d'appel, le compilateur nous met le warning C4272.

```
warning C4272 : [...] is marked __declspec(dllimport); must specify native calling convention when
 importing a function.
```

Ainsi, on pourra définir ses fonctions ainsi :

code natif

```
__declspec(dllexport) int __stdcall addition(int a,int b)
{
 return a+b;
}
```

code managé

```
#pragma comment (lib, "testDll.lib")
__declspec(dllimport) int __stdcall addition(int a,int b);
```

Avec DLLImport, on utilisera les mécanismes de P/Invoke :

```
using namespace System::Runtime::InteropServices;
[DllImport("testDll")]
extern "C" int addition(int a,int b);
```

Comment utiliser des variables natives dans une classe managée ?

Auteur : nico-pyright(c)

Lorsqu'on veut utiliser une variable native dans une classe managée, comme ci dessous :

```
#include <iostream>
ref class MaClasse
{
public:
 MaClasse() {};
private:
 std::string s;
};
```

On obtient l'erreur de compilation C4368 :

```
error C4368: cannot define 's' as a member of managed 'MaClasse': mixed types are not supported
```

Il faut passer obligatoirement par un pointeur et procéder à son allocation.

```
#include <iostream>
ref class MaClasse
{
public:
 MaClasse() {};
private:
 std::string *s;
};
```

Comment utiliser des objets managés dans une classe native ?

Auteur : nico-pyright(c)

Lorsqu'on veut utiliser un objet managé dans une classe native, comme ci dessous :

```
ref class CManagee
{
 CManagee(){};
};

class CNative
{
public:
 CNative();
 CManagee ^c;
};
```

On obtient l'erreur de compilation C3265 :

```
error C3265: cannot declare a managed 'c' in an unmanaged 'CNative'
        may not declare a global or static variable, or a member of a native type that refers to
 objects in the gc heap
```

Une classe native ne sait pas gérer, allouer, etc ... les objets managés en tant que tel. Il faut utiliser les mécanismes du CLR qui sont factorisés dans le template gcroot. (utilisation nottament de GCHandle::Alloc).

```
#include <vcclr.h>
ref class CManagee
{
 CManagee() {};
};

class CNative
{
public:
 CNative();
 gcroot<CManagee ^> c;
};
```

On n'oubliera pas bien sur d'inclure le fichier vcclr.h.

Quelle est la différence entre gcroot et auto_gcroot ?

Auteur : nico-pyright(c)

gcroot est un template qui permet d'utiliser des types CLI dans une classe native comme Comment utiliser des objets managés dans une classe native ?. Pour l'utiliser, on inclut :

```
#include <vcclr.h>
```

auto_gcroot est aussi un template qui permet d'utiliser des types CLI dans une classe native, à la différence près qu'en utilisant ce template, on est assuré de la libération des ressources quand il est supprimé ou quand il sort de son scope (comme auto_ptr en C++).
Il se situe dans le namespace msclr. Pour l'utiliser, on inclut :

```
#include <msclr\auto_gcroot.h>
```

Pour s'en convaincre et pour voir un exemple d'utilisation, soit le code suivant :

```
ref class CManagee
{
private:
 String ^s;
public:
 CManagee(String ^val) {s = val;};
 ~CManagee()
 {
  Console::WriteLine("Destructeur : " + s);
 }
protected:
 !CManagee()
 {
  Console::WriteLine("Finalizer " + s);
 }
};

class CNative
{
public:
 CNative()
 {
  c = gcnew CManagee("gcroot");
  cAuto = gcnew CManagee("auto_gcroot");
 }
private:
 gcroot<CManagee^> c;
 msclr::auto_gcroot<CManagee^> cAuto;
};

int main(array<System::String ^> ^args)
{
 CNative *c = new CNative();
 delete c;
 Console::WriteLine("fin du programme");
 return 0;
}
```

Produit en sortie :

```
Destructeur : auto_gcroot
fin du programme
Finalizer gcroot
```

On a donc bien, lors de la destruction explicite de l'objet natif, la libération du handle cAuto grâce au template auto_gcroot.
On a ensuite la fin du programme.
Puis la libération du handle c par le garbage collector lorsque l'objet n'est plus utilisé, par l'appel du finalizer.

Comment savoir si un handle géré par le template gcroot est nul ?

Auteur : nico-pyright(c)

Il peut être utile de tester si un handle managé par gcroot ou auto_gcroot est nul.
Il n'est pas possible de faire ça :

```
class CNative
{
public:
 CNative()
 {
  if (chaine == nullptr)
   chaine = gcnew String("gcroot");
 }
private:
 gcroot<String ^> chaine;
};
```

On obtient pour gcroot en sortie :

```
error C2088: '==' : illegal for struct
```

Pour comparer à nullptr, il faut caster :

```
class CNative
{
public:
 CNative()
 {
  if (safe_cast<String ^>(chaine) == nullptr)
   chaine = gcnew String("gcroot");
 }
private:
 msclr::auto_gcroot<String ^> chaine;
};
```

Cependant, on peut utiliser une écriture plus simple pour tester si la variable référence un objet managé ou un nullptr :

```
class CNative
{
public:
 CNative()
 {
  if (!chaine)
   chaine = gcnew String("gcroot");
 }
```

```
private:
 msclr::auto_gcroot<String ^> chaine;
};
```

Comment convertir une String ^ en char * ?

Auteurs : nico-pyright(c) - StormimOn

Il est parfois utile de pouvoir convertir une String .Net en char *, notamment lorsqu'on veut utiliser des API Win32.
Après cette conversion, on pourra par exemple utiliser chaineChar avec SetWindowText.

```
String ^chaineManagee = "Chaine Managée";
char*
 chaineChar = static_cast<char *>(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(chaineManagee)
// utilisation de la chaine
System::Runtime::InteropServices::Marshal::FreeHGlobal(safe_cast<IntPtr>(chaineChar));
```

Remarque, la chaîne devra être utilisée avant sa libération avec FreeHGlobal.

On peut procéder aussi d'une autre façon :

```
#include <vcclr.h>

String ^chaineManagee = gcnew String("une chaine Managée");
char * chaineChar;
pin_ptr<const wchar_t> wch = PtrToStringChars(chaineManagee);
int taille = (chaineManagee->Length+1) * 2;
chaineChar = new char[taille];
int t = WideCharToMultiByte(CP_ACP, 0, wch, taille, NULL, 0, NULL, NULL);
WideCharToMultiByte(CP_ACP, 0, wch, taille, chaineChar, t, NULL, NULL);
```

Cette solution a l'inconvénient d'appeler une méthode de l'API Win32 pour la conversion.
Ne pas oublier de libérer chaineChar après utilisation (avec delete).

Comment convertir une String ^ en wchar_t * ?

Auteurs : abelman - nico-pyright(c)

Il est parfois utile de pouvoir convertir une String .Net en wchar_t *, notamment lorsqu'on veut utiliser des API Win32 en Unicode.
Après cette conversion, on pourra par exemple utiliser wchar avec SetWindowText.

```
String ^chaineManagee = "Chaine Managée";
IntPtr p =  System::Runtime::InteropServices::Marshal::StringToHGlobalUni(chaineManagee);
LPCWSTR wchar = static_cast<LPCWSTR>(static_cast<void *>(p));
System::Runtime::InteropServices::Marshal::FreeHGlobal(p);
```

On peut procéder aussi d'une autre façon :

```
// Helper pour extensions managées
#include <vcclr.h>

// pin_ptr indique que le Garbage collector ne pourra pas déplacer la zone mémoire correspondante
```

```
pin_ptr<const wchar_t> strValueDLLPath =  PtrToStringChars(sValuesDLLPath);
```

Comment convertir un char * en String ^ ?

Auteur : nico-pyright(c)

Pour convertir une chaine de type char * en String ^, rien de plus simple :

```
char chaine[255] = "Une chaine";
String ^ str = gcnew String(chaine);
```

Comment convertir un string de la STL en String ^ ?

Auteur : nico-pyright(c)

On utilise la surcharge du constructeur pour un type const char * :

```
String ^ s = gcnew String(chaineSTL.c_str());
```

Comment convertir une String ^ en string de la STL ?

Auteur : nico-pyright(c)

Le principe est de convertir d'abord en char *, puis de convertir en string :

```
String ^chaineManagee = gcnew String("une chaine Managée");
char * chaineChar;
pin_ptr<const wchar_t> wch = PtrToStringChars(chaineManagee);
int taille = (chaineManagee->Length+1) * 2;
chaineChar = new char[taille];
int t = WideCharToMultiByte(CP_ACP, 0, wch, taille, NULL, 0, NULL, NULL);
WideCharToMultiByte(CP_ACP, 0, wch, taille, chaineChar, t, NULL, NULL);

std::string chaineSTL = chaineChar;
```

Comment convertir un objet d'un type de base en un objet d'un autre type de base ?

Auteurs : abelman - nico-pyright(c)

La classe System::Convert permet de convertir des objets de types de base.
Il existe d'autres méthodes pour le faire, mais l'avantage de la classe System::Convert est qu'elle est indépendante du langage utilisé.
System::Convert dispose d'une multitude de méthodes pour effectuer toutes sortes de conversions possibles entre les types de bases.
Voici quelques exemples :

```
// Conversion d'un entier vers une chaînes de caractères
int i = 10;
String ^s = Convert::ToString(i);

// Conversion d'une chaine vers un entier
i = Convert::ToInt32(s);
// Notez que si la conversion ne peut se faire, une exception est levée.
// Ce serait le cas si s = "NonNumérique"
```

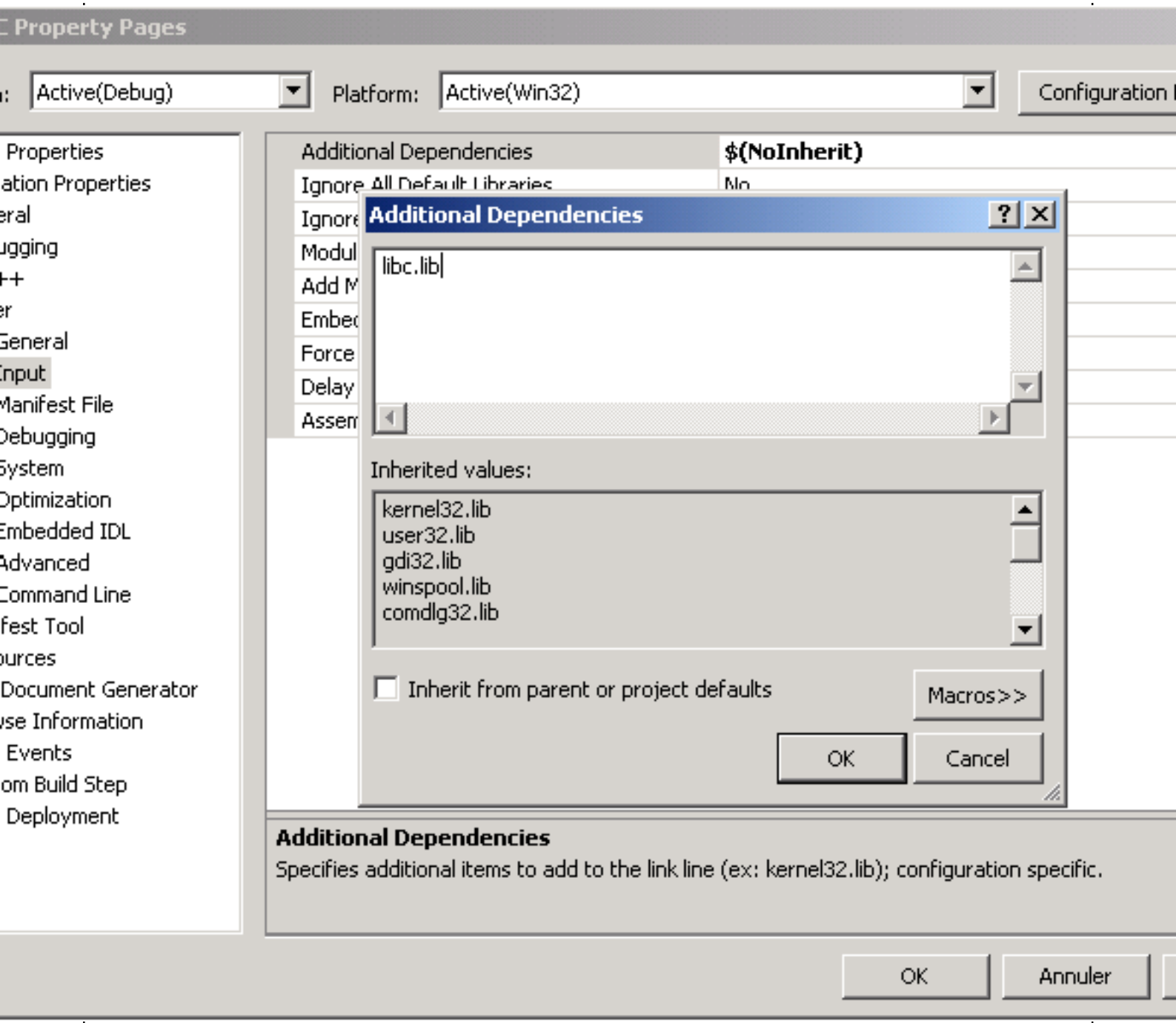
Il est aussi possible d'utiliser la méthode Parse de chaque type de base.

```
String ^s = "35000";
int i = int::Parse(s);
```

Comment utiliser une bibliothèque native dans mon application ?

Auteur : nico-pyright(c)

Il s'agit de pouvoir utiliser une bibliothèque style C dans notre application. Pour ce faire, on va utiliser les mécanismes d'IJW, ce qui fait qu'on ne va pas avoir grand chose à faire.
On dispose en général d'un .h et d'un .lib livrés avec la dll.
Il faudra donc lier la lib. Pour ceci, deux solutions possibles :
- bouton droit sur le projet => properties => configuration properties => linker => input => additional depedencies, et ici, on rajoute le nom du .lib.



- soit par pragma :

```
#pragma comment (lib, "laLib.lib")
```

Il ne restera plus qu'à inclure le .h correspondant, et on pourra utiliser les fonctions telles quelles.

NB : s'il n'y a pas de .h livré, on pourra toujours déclarer explicitement les prototypes dans notre application :

```
extern "C" __declspec(dllexport) int __stdcall addition(int a,int b);
```

lien : Que faire avec l'erreur d'édition de liens LNK2031 ?
lien : Introduction à l'intéropérabilité

Comment utiliser une dll dynamiquement ?

Auteur : nico-pyright(c)

Il est possible d'utiliser les mécanismes de P/Invoke pour charger des dll dynamiquement, grâce à l'attribut DllImport.

Il faudra référencer l'assembly InteropServices :

```
using System::Runtime::InteropServices;
```

Et ensuite utiliser DLLImport.

```
using namespace System::Runtime::InteropServices;
[DllImport("testDll.dll")]
extern "C" int addition(int a,int b);
```

lien : Introduction à l'intéropérabilité

Sommaire > Intéraction du C++/CLI avec le framework .Net

Sommaire > Intéraction du C++/CLI avec le framework .Net > Système

Comment obtenir la version de mon application ?

Auteurs : abelman - nico-pyright(c)

Pour obtenir la version de mon application, on utilise la classe System::Diagnostics::FileVersionInfo.

```
FileVersionInfo ^ ver = FileVersionInfo::GetVersionInfo(Environment::GetCommandLineArgs()[0]);
String ^ myVer = ver->FileMajorPart + "." + ver->FileMinorPart + "." + ver->FileBuildPart + "." +
 ver->FilePrivatePart;
```

Pour obtenir la version d'un fichier (EXE, DLL, OCX, etc..) quelconque, il suffit de passer le chemin du fichier à la fonction GetVersionInfo.

Comment obtenir la version du .NET Framework en cours ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété Version de la classe System::Environment.

```
Version ^ ver = Environment::Version;
Console::WriteLine("Version Framework = {0}", ver);
```

Comment obtenir la version de l'OS en cours ?

Auteurs : freegreg - nico-pyright(c)

On utilise la propriété OSVersion de la classe System::Environment.

```
OperatingSystem ^ os = Environment::OSVersion;
Console::WriteLine("Version OS = {0}", os);
```

Comment obtenir le nom de l'utilisateur dont la session est en cours ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété UserName de la classe Environment.

```
Console::WriteLine("Nom utilisateur : {0}", Environment::UserName);
```

Comment obtenir le nom de la machine ?

Auteur : Jérôme Lambert

On utilise la propriété MachineName de la classe System::Environment :

```
// Nom de la machine
String ^MachineName = Environment::MachineName;
Console::WriteLine("Nom de la machine = {0}", MachineName);
```

lien : Classe Environment

Comment obtenir la quantité de mémoire physique allouée à mon application ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété WorkingSet de la classe Environment.

```
Console::WriteLine("Mémoire allouée au process en cours : {0}", Environment::WorkingSet);
```

Comment obtenir la ligne de commande de l'application ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété CommandLine de la classe Environment pour avoir la ligne entière dans une chaîne de caractères.

```
Console::WriteLine("La ligne de commande est : {0}", Environment::CommandLine);
```

On utilise GetCommandLineArgs pour obtenir un tableau de chaînes de caractères contenant chaque argument.

```
// Arguments de ligne de commande
Console::WriteLine("Arguments de ligne de commande");
cli::array<String ^> ^args = Environment::GetCommandLineArgs();
for (int i = 0; i<args->Length; i++)
 Console::WriteLine("Args({0}) = {1}", i, args[i]);
```

Remarque : Ce même tableau est passé en paramètre de la fonction main :

```
int main(array<System::String ^> ^args)
```

Comment obtenir les variables d'environnement ?

Auteurs : abelman - nico-pyright(c)

On utilise la fonction GetEnvironmentVariables de la classe Environment.

```
IDictionary ^ env = Environment::GetEnvironmentVariables();
// Pour afficher une variable dont on connait le nom
Console::WriteLine("USERNAME = {0}", env["USERNAME"]);
// Pour lister toutes les variables
for each (String ^s in env->Keys)
 Console::WriteLine("{0} = {1}", s, env[s]);
```

Comment générer des nombres aléatoires ?

Auteurs : abelman - nico-pyright(c)

La classe System::Random nous permet de générer des nombres aléatoires.
Il s'agit en fait de nombres pseudo-aléatoires, car la séquence générée dépend de l'initialisation.

```
// Pour générer toujours la même séquence,
// on passe la même valeur au constructeur.
// Random rnd = new Random(100);

// Initialisation par défaut basée sur le temps.
// La séquence est différente à chaque fois.
Random ^rnd = gcnew Random();

// Génération de 15 nombres aléatoires compris entre 0 et 255
cli::array<unsigned char> ^ rndNumbers = gcnew cli::array<unsigned char>(15);
rnd->NextBytes(rndNumbers);
this->listBox1->Items->Clear();
for (int i=0; i<15; i++)
    // On peut aussi faire un modulo pour n'obtenir que
    // des nombres entre 0 et 100 par exemples
    if (rndNumbers[i] > 100)
        listBox1->Items->Add(rndNumbers[i] % 100);
    else
        listBox1->Items->Add(rndNumbers[i]);

// Pour générer des nombres aléatoire de type Integer
int i = rnd->Next();
int j = rnd->Next(500, 1000); // j sera compris entre 500 et 999 inclu

// Pour générer des nombre aléatoire de type Double
// d sera compris entre 0,0 et 1,0.
// Il suffit de combiner cet appel à celui de Next()
// pour avoir des doubles supérieurs à 1,0
double d = rnd->NextDouble();
```

Comment lire et écrire des données sur la console ?

Auteurs : abelman - nico-pyright(c) - Thomas Lebrun

Il vous faut utiliser la classe System::Console et les méthodes qui la composent.

```
Console::WriteLine ("Saisissez un texte: ");
String ^ _Texte = Console::ReadLine ();
Console::WriteLine ("Vous avez écrit : " + _Texte);
```

Console::Writeline écrit par défaut sur

- **La console si il y en a une active**
- **La fenêtre de debug (onglet Sortie - Déboguer) si l'application ne possède pas de console. Une application WinForm par exemple.**

Comment fonctionne le Garbage Collector ?

Auteur : Keihilin

Ce qui suit est extrait d'un échange sur le forum dotnet.
C'est très shématique mais cela résume bien comment le .NET Framework s'y prend pour gérer la mémoire
Le .NET Framework : Salut OS, j'ai des trucs à lancer, j'peux te prendre de la ram ?
L'OS : Hé Salut ! Je t'en pris, sers-toi !
Le .NET Framework : Sympa mec. J't'en prend 50Mo maintenant, j'ai besoin que de 15 Mo,
mais comme ça je te dérange pas si j'ai besoin de plus.
...
Le .NET Framework : Hé l'OS, t'es short niveau mémoire ?
L'OS : Non non, tout va bien.
Le .NET Framework : Bon, alors je garde mes 50 Mo encore un peu.
L'OS : Oki.
...
SQL Server : Bonjour M. l'OS, j'ai un gros besoin de mémoire...au moins 200 Mo.
L'OS : Ben sers-toi donc.
SQL Server : Ouais mais y a plus que 180Mo !
L'OS : Ah OK, attend 2 millisecondes stp...
L'OS : Hé Framework, tu peux me rendre un peu de RAM ?
Le .NET Framework : No problemo, j'te fais ça tout de suite...
Le .NET Framework : Garbage Collector, soit un amour et va rendre de la mémoire à l'OS.
Garbage Collector : J'y cours patron.
C'est clair non ?

Comment forcer la libération de la mémoire par le Garbage Collector ?

Auteur : Thomas Lebrun

Pour forcer le Garbage Collector à libérer la mémoire inutilisée par le .NET Framework, on peut appeller la méthode Collect de la classe GC.

```
System::GC::Collect();
```

Par contre, pour des raisons qui justifieraient à elles seules un article, il n'est pas conseillé d'appeller GC::Collect() directement.
Par conséquent, ne le faites pas à moins d'être un expert du garbage collector.

Comment puis-je appeler une fonction présente dans une DLL win32 ?

Auteur : nico-pyright(c)

Grâce à It Just Work, il est très facile d'appeler une fonction d'une API Win32. Il suffit d'inclure les entêtes nécessaires (comme windows.h) et de lier la librairie par pragma si ce n'est pas déjà le cas.
Le mode de compilation ne pourra pas être /clr:safe.

```
#include <windows.h>
#pragma comment(lib, "Kernel32.lib")
....
GetModuleFileName(0, nom, MAX_PATH);
```

Comment mesurer précisément le temps d'exécution d'une partie de votre code?

Auteurs : nico-pyright(c) - Webman

Tout d'abord, ajoutez en entête de votre classe :

```
using namespace System::Diagnostics;
```

Ensuite, ajoutez ceci dans la partie de code qui vous intéresse :

```
//Instanciation d'un objet StopWatch
Stopwatch ^monStopWatch = gcnew Stopwatch();

// Déclenchement du "chronomètre"
monStopWatch->Start();

// ////////////////////////////////
// Placez ici le code que vous voulez évaluer, cela peut être par exemple du code 'brut' ou
// alors l'appel d'un méthode...
// ////////////////////////////////

// Arrêt du "chronomètre"
monStopWatch->Stop();

// Le temps écoulé peut être récupéré très facilement avec un membre de StopWatch,
// de la façon suivante. Le résultat est exprimé en millisecondes
long tempsExecution;
tempsExecution = safe_cast<long>(monStopWatch->ElapsedMilliseconds);
```

Comment remplacer un mot (insensible à la casse) par un autre grâce aux expressions régulières ?

Auteurs : nico-pyright(c) - Webman

Tout d'abord, ajoutez en entête de votre classe :

```
using namespace System::Text::RegularExpressions;
```

Un exemple vaut mieux qu'un long discours :

```
String ^monTexte = "Une astuce de dvp.com ! Une astuce de Dvp.Com !";

// Paramétrage de notre expression régulière :
// Ici on spécifie que l'on ne veut pas tenir compte dela casse du
// texte dans nos remplacements.
Regex ^maRegEx = gcnew Regex("\\bdvp.com\\b", RegexOptions::IgnoreCase);

// Remplacement des occurences de "dvp.com" par "Developpez.com"
monTexte = maRegEx->Replace(monTexte, "Developpez.com");
```

Comment utiliser une ressource dans mon exécutable ?

Auteur : nico-pyright(c)

Il faut dans un premier temps dire au linker d'intégrer un ou plusieurs fichiers de ressources :

- **Click droit sur le projet --> Properties --> Linker --> Input --> Embed Managed Resource File**
- **Indiquer le path complet du fichier à intégrer *(exemple : e:\abc.bmp)*.**

Il faut ensuite utiliser le namespace Reflection::Assembly pour récupérer l'assembly et ensuite, invoquer la méthode qui récupère le flux du fichier en ressources. *Exemple pour afficher une image dans un pictureBox*.

```
pictureBox1->Image = Image::FromStream(Reflection::Assembly::GetExecutingAssembly()->GetManifestResourceStream(
```

Comment écrire dans le journal des évènements des services Windows ?

Auteurs : neguib - neo.51 - nico-pyright(c)

Tout simplement en utilisant les fonctionnalités offertes par l'espace de noms System::Diagnostics. Notamment la classe EventLog et sa méthode WriteEntry qui dispose d'une dizaine de surcharges.

```
if (! EventLog::SourceExists("MonApplication"))
 EventLog::CreateEventSource("MonApplication", "MonJournal");
```

```
EventLog^ MonJournal = gcnew EventLog();
MonJournal->Source = "MonApplication";

try
{
 // instructions à exécuter  (ex : division par zéro;
 int i = 0;
 Console::WriteLine(2/i);
}
catch(Exception^ ex)
{
 // écrire l'erreur
 MonJournal->WriteEntry(ex->Message,EventLogEntryType::Error);
}
```

lien : System.Diagnostics.EventLog
lien : EventLog.WriteEntry
lien : EventLogEntryType

Comment connaître le nombre d'écrans connectés à l'ordinateur ?

Auteur : Jérôme Lambert

On utilise la propriété MonitorCount de la classe SystemInformation :

```
// Nombre d'écrans connectés
int MonitorCount = SystemInformation::MonitorCount;
Console::WriteLine("Nombre d'écrans connectés à l'ordinateur = {0}", MonitorCount);
```

On n'oubliera pas de référencer l'assembly :

```
#using "System.Windows.Forms.dll"
using namespace System::Windows::Forms;
```

lien : Classe SystemInformation

Comment connaître le nombre de processeurs que possède la machine ?

Auteur : Jérôme Lambert

On utilise la propriété ProcessorCount de la classe System::Environment :

```
// Nombre de processeurs de la machine
int countProc = Environment::ProcessorCount;
Console::WriteLine("Nombre processeurs = {0}", countProc);
```

lien : Classe Environment

Comment tester si l'utilisateur de la session fait partie du groupe d'un domaine ?

Auteur : Jérôme Lambert

On utilisera la méthode IsInRole de la classe WindowsPrincipal afin de déterminer si l'utilisateur courant fait bien partie ou pas d'un groupe x du domaine auquel il est rattaché :

```
// Vérification si un utilisateur appartient à un groupe de domaine donné
WindowsPrincipal ^wp = gcnew WindowsPrincipal(WindowsIdentity::GetCurrent());
if (wp->IsInRole("Developpez\\Moderateur") == true)
 Console::WriteLine("L'utilisateur fait bien partie du groupe Developpez\\Moderateur");
else
 Console::WriteLine("L'utilisateur ne fait pas partie du groupe Developpez\\Moderateur");
```

lien : Classe WindowsPrincipal

Comment déterminer le mode de démarrage de la machine ?

Auteur : Jérôme Lambert

On utilise la propriété BootMode de la classe SystemInformation. On obtient une des valeurs de l'énumération BootMode : FailSafe, FaileSafeWithNetwork ou Normal.

```
#using "System.Windows.Forms.dll"
// ......
switch (SystemInformation::BootMode)
{
 case BootMode::Normal:
  Console::WriteLine("Démarrage normal !");
  break;
 case BootMode::FailSafe:
  Console::WriteLine("Attention, on est en mode sans echec !");
  break;
 case BootMode::FailSafeWithNetwork:
  Console::WriteLine("Attention, on est en mode sans échec mais on a accès au réseau !");
  break;
     }
```

lien : System.Windows.Forms.SystemInformation

Comment créer une exception personnalisée ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Toute exception dérive de la classe Exception. Cependant, il est conseillé de faire dériver vos exceptions personnalisées de la classe ApplicationException. Un tiers développeur qui utilisera votre assembly pourra donc facilement reconnaître une exception du framework à une exception métier/applicative (dérivée de ApplicationExeption).

```
ref class MonExceptionApplication : ApplicationException
{
public:
    MonExceptionApplication() : ApplicationException("Exception personnalisée")
    { }

 MonExceptionApplication(String ^p_message) : ApplicationException("Exception personnalisée : " +
 p_message)
    { }
};

int main(array<System::String ^> ^args)
{
 try
 {
  throw gcnew MonExceptionApplication("rien à faire");
 }
```

```
 catch (MonExceptionApplication^ ex)
 {
  Console::WriteLine(ex->Message);
 }
    return 0;
}
```

Comment obtenir la description d'une extension, comme dans l'explorateur de Windows ?

Auteurs : nico-pyright(c) - smyley

Tout est dans le titre ...
Il faut rajouter dans les usings

```
using namespace Microsoft::Win32;
```

et voici le code ...

```
String ^AGetExtentionDescription(String ^ext)
{
 String ^description = GetExtentionDescription(ext);
 if (description == nullptr || description == "")
  return String::Format("Fichier {0}", ext);
 else
  return description;
}

String ^GetExtentionDescription(String ^ext)
{
 if (ext == nullptr || ext == "" || ext == ".")
  return nullptr;
 String ^description = "";
 try
 {
  RegistryKey ^keyDesc = Registry::ClassesRoot->OpenSubKey(ext, false);
  if (keyDesc == nullptr)
   return nullptr;
  else
  {
   try
   {
    if (keyDesc->GetValueKind("") != RegistryValueKind::String)
     return nullptr;
    else
    {
     description = (String^)keyDesc->GetValue("");
     if (description == nullptr || description == "")
      return nullptr;
     RegistryKey ^optionnalRedirection = Registry::ClassesRoot->OpenSubKey(description, false);
     if (optionnalRedirection == nullptr)
      return description;
     else
     {
      try
      {
       if (optionnalRedirection->GetValueKind("") != RegistryValueKind::String)
        return description;
       else
        return (String ^)optionnalRedirection->GetValue("");
      }
      finally
      {
       if (optionnalRedirection != nullptr)
```

```
          optionnalRedirection->Close();
        }
       }
      }
     }
     finally
     {
      if (keyDesc != nullptr)
       keyDesc->Close();
     }
    }
   }
   catch(Exception ^)
   {
    return nullptr;
   }
  }

  System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e)
  {
   MessageBox::Show(AGetExtentionDescription(".exe"));
   MessageBox::Show(AGetExtentionDescription(".jpg"));
   MessageBox::Show(AGetExtentionDescription(".dat"));
   MessageBox::Show(AGetExtentionDescription(".iso"));
   MessageBox::Show(AGetExtentionDescription(".bmp"));
  }
```

Comment ajouter le numéro de version à mon exécutable/assembly ?

Auteur : nico-pyright(c)

Pour ajouter la version à son projet C++/CLI, on utilise une ressource de type version.
Explorateur de ressources -> bouton droit -> Add resource -> Version

Comment créer et afficher une nouvelle Winform ?

Auteur : nico-pyright(c)

Il faut dans un premier temps créer une nouvelle Winform :
Click droit sur le projet --> Add -- New Item --> UI --> Winforms Form.
Là, nommez-la comme bon vous semble, par exemple Form2. Visual Studio génère donc une nouvelle classe, et la winform est modifiable également à travers l'IDE.

Pour afficher cette nouvelle form, par exemple depuis un click sur un bouton de la première form, il faut inclure le Form2.h dans le fichier .h de notre première Form (Form1.h par défaut).

```
#include "Form2.h"
```

Ensuite dans l'événement du click sur le bouton (ou autre), il faut instancier la classe et appeler la méthode Show() sur l'objet créé.

```
Form2 ^maForm2 = gcnew Form2();
maForm2->Show();
```

Notez que la méthode Show() affiche la fenêtre simplement. On peut également utiliser ShowModal() pour qu'elle soit modale comme une boîte de dialogue.

Comment changer le curseur de mon application ?

Auteurs : abelman - nico-pyright(c) - Thomas Lebrun

Lors de la réalisation d'applications, il peut être utile de montrer à l'utilisateur que le traitement demandé est en cours (et qu'il faut patienter).
Un bon moyen de réaliser ceci est de changer la forme du curseur, pendant la durée du traitement.
Pour ce faire, vous devez utiliser la Classe Cursors.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{
 // On passe le curseur en sablier
 Cursor = Cursors::WaitCursor;
 System::Threading::Thread::Sleep(4000);
 // On repasse le curseur en normal
 Cursor = Cursors::Arrow;
}
```

N'hésitez pas à vous reporter à la classe Cursors pour voir les types de curseurs disponibles et utilisables par votre application.

Comment annuler la fermeture de la session Windows ?

Auteurs : nico-pyright(c) - swoög

Il faut tout d'abord gérer l'évènement FormClosing (depuis l'IDE ou par code).

```
this->FormClosing +=
 gcnew System::Windows::Forms::FormClosingEventHandler(this, &Form1::Form1_FormClosing);
```

Ensuite, insérer ce petit bout de code dans notre classe, ce qui permet de détecter une tentative de fermeture de session en surchargeant la méthode de traitement des messages windows et en interceptant le message de fermeture de session.
Il ne reste plus qu'à l'annuler dans le FormClosing.

```
private : bool _systemShutdown; // a initialiser à false dans le constructeur

protected:virtual void WndProc(Message %m) override
{
 // Mise dans systemShutdown la présence du message fermeture Windows
 if (m.Msg == 0x11) // ou utiliser WM_QUERYENDSESSION de windows.h
  _systemShutdown = true;
 Form::WndProc(m);
}

private: System::Void Form1_FormClosing(System::Object^
 sender, System::Windows::Forms::FormClosingEventArgs^  e)
{
 //Si le message fermeture Windows a été envoyé, on l'annule !
 if (_systemShutdown)
 {
  e->Cancel = true;
  _systemShutdown = false;
  MessageBox::Show("Fermeture de session windows annulée");
 }
}
```

Pour tester ce code, éxecutez le en dehors de l'IDE.

Comment permettre à l'utilisateur de choisir un répertoire ?

Auteurs : abelman - nico-pyright(c)

On utilise le composant FolderBrowserDialog pour cela.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{
 folderBrowserDialog1->Description = "Choisissez votre répertoire";
 if (folderBrowserDialog1->ShowDialog(this) == System::Windows::Forms::DialogResult::OK)
 {
  MessageBox::Show(this,
```

```
    "Vous aves choisi " + folderBrowserDialog1->SelectedPath,
    "Repertoire",
    MessageBoxButtons::OK,
    MessageBoxIcon::Information);
  }
 }
```

Comment permettre à l'utilisateur de choisir un fichier pour ouvrir un document ?

Auteurs : abelman - nico-pyright(c)

Le composant System::Windows::Form::OpenFileDialog permet à l'utilisateur de choisir interactivement un fichier afin d'y lire des données.
Créez une form et placez-y un bouton nommé button1, un composant RichTextBox nommé richTextBox1 et un composant OpenFileDialog nommé openFileDialog1.
Un clic sur le bouton1 permet de lire le fichier choisi et d'afficher son contenu dans le RichTextBox.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{
 // Titre
 openFileDialog1->Title = "Chargement";
 // Extension par défaut
 openFileDialog1->DefaultExt = "txt";
 // Filtre sur les fichiers
 openFileDialog1->Filter = "fichiers textes (*.txt)|*.txt|Tous les fichiers (*.*)|*.*";
 openFileDialog1->FilterIndex = 1;
 // Ouverture boite de dialogue OpenFile
 if (openFileDialog1->ShowDialog(this) == Windows::Forms::DialogResult::OK)
 {
  // On vide le TextBox
  richTextBox1->Text = String::Empty;
  // Ouverture du fichier sélectionné
  // son nom est dans openFileDialog1->FileName
  IO::StreamReader ^sr =
 gcnew IO::StreamReader(openFileDialog1->OpenFile(), System::Text::Encoding::Default);
  try
  {
   richTextBox1->Text = sr->ReadToEnd();
  }
  finally
  {
   if (sr!=nullptr)
    sr->Close();
  }
 }
}
```

Comment permettre à l'utilisateur de choisir un fichier pour enregistrer un document ?

Auteurs : abelman - nico-pyright(c)

Le composant System::Windows::Form::SaveFileDialog permet à l'utilisateur de choisir interactivement un fichier afin d'y écrire des données.
Créez une form et placez-y un bouton nommé button1, un composant RichTextBox nommé richTextBox1 et un composant SaveFileDialog nommé saveFileDialog1.
Un clic sur le bouton1 permet de sauvegarder le contenu du RichTextBox vers le fichier choisi.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{
 // Demande de confirmation
 if (MessageBox::Show(this,
   "Sauvegarder le document?",
   "Sauvegarde",
   MessageBoxButtons::YesNo,
   MessageBoxIcon::Question) == Windows::Forms::DialogResult::No)
  return;
 // Sauvegarde du document
 saveFileDialog1->Title = "Sauvegarde";
 saveFileDialog1->DefaultExt = "txt";
 saveFileDialog1->Filter = "fichiers textes (*.txt)|*.txt|Tous les fichiers (*.*)|*.*";
 saveFileDialog1->FilterIndex = 1;
 // Ouverture boite de dialogue Enregistrer
 if (saveFileDialog1->ShowDialog(this) == Windows::Forms::DialogResult::OK)
 {
  // StreamWriter pour écrire dans le fichier sélectionné
  IO::StreamWriter ^sw =
 gcnew IO::StreamWriter(saveFileDialog1->OpenFile(), System::Text::Encoding::Default);
  try
  {
   for (int i=0; i<richTextBox1->Lines->Length; i++)
    sw->WriteLine(richTextBox1->Lines[i]);
  }
  finally
  {
   // Fermeture writer
   if (sw!=nullptr)
    sw->Close();
  }
 }
}
```

Comment accèder à une méthode publique d'une form à partir d'une autre form créé par la première ?

Auteurs : dev01 - nico-pyright(c)

Afin de pouvoir communiquer à partir d'une form nouvellement ouverte vers la form créatrice vous devez passer la form créatrice à la nouvelle form.

```
//Dans Form1
public void TraitementForm1()
{

}

public void OuvertureForm2()
{
form2 ^maForm2 = gcnew form2();
maForm2->Owner = this;
maForm2->Show();
}

//Dans MaForm2

private void ButtonOk_Click(object sender, System.EventArgs e)
{
Form1 ^maForm1 = safe_cast<Form1 ^>(this->Owner);
maForm1->TraitementForm1();
```

```
}
...
```

Comment récupérer une valeur d'un contrôle depuis une autre Form (inclusions croisées et déclaration anticipée) ?

Auteur : nico-pyright(c)

Lorsqu'une winform 1 doit instancier une winform 2 qui doit connaitre la winform 1, on est obligé de faire un include de form1.h dans form2.h et inversement de form2.h dans form1.h.
Ceci amène à des références croisées qui provoquent des erreurs de compilation.

Imaginons que nous voulions faire une application qui ouvre une fenêtre qui possède un textBox et un bouton. Lors du click sur le bouton, on affiche une deuxième form qui possède elle aussi un bouton et un textBox (ça aurait pu être un label). On souhaite lors du clic sur le bouton de la deuxième form afficher la valeur saisie dans le textBox de la première form...
Voici comment on peut faire :
Dans la form1, je dépose le textBox1 et le bouton.
J'ajoute une nouvelle form, Form2 dans laquelle je mets le textbox et le bouton.
le but est de pouvoir lire la valeur saisie dans la Form1 depuis la Form2 sachant que c'est la form1 qui va créer la form2.
Donc, en toute logique, dans la form1, on va avoir à un endroit la création et l'affichage de la form2. Comme il faut que la form2 connaisse la form1, on va lui passer l'objet form1 (c'est à dire this), disons dans le constructeur :

```
Form2 ^maForm2 = gcnew Form2(this);
maForm2->Show();
```

Dans la form2, on aura donc fatalement la récupération de la form1 dans le constructeur :

```
Form2(Form1 ^ laForm1)
{
  Form1 ^currForm1 =
 laForm1; // sachant qu'en vrai, on aura currForm1 comme membre privé de la classe
}
```

et au moment de récupérer la valeur, on aura un

```
textBox1->Text = currForm1->GetValeur();
```

GetValeur sera bien sur une méthode publique de form1 :

```
public:
  String ^GetValeur()
  {
    return textBox1->Text;
  }
```

Tant est si bien que dans Form1.h, on va devoir faire un

```
#include "Form2.h"
```

pour pouvoir instancier la form2
et dans la Form2.h, on va devoir faire un

```
#include "Form1.h"
```

pour pouvoir utiliser la form1.
Si vous essayez de compiler directement comme ca, il va y avoir une tripotée d'erreurs, la plus significative étant qu'il ne connait pas Form2, avec une erreur du genre dans Form1.h :

```
error C2143: erreur de syntaxe : absence de ';' avant '^'
```

sur la ligne Form2 ^ currForm2;
Comme on s'en doute, chaque fichier incluant l'autre, il y a des inclusions croisées qui empêchent la bonne compilation.
Pour que ceci compile, il faut plusieurs choses :
Déjà, déporter l'implémentation des méthodes des classes dans un .cpp (plus précisément, toutes celles qui utilisent des méthodes ou des attributs de Form2 pour le fichier form1.h et toutes celles qui utilisent des méthodes ou des attributs de Form1 pour form2.h).
Donc, il faut créer un fichier Form1.cpp (form2.cpp étant généré automatiquement lors de l'ajout d'une nouvelle form).
Dans mon .h, j'ai donc laissé la définition des méthodes :

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e);
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
```

et dans le .cpp, j'ai mis l'implémentation des méthodes :

```
include "StdAfx.h"
#include "Form1.h"

namespace testWinforms {

System::Void Form1::Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
  currForm2 = gcnew Form2(this);
}

System::Void Form1::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
  currForm2->Show();
}

}
```

NB : ne pas oublier le namespace, ni l'opérateur de résolution de portée. Idem dans form2.h

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e);
```

et dans le Form2.cpp

```
System::Void Form2::button1_Click(System::Object^ sender, System::EventArgs^ e)
{
  textBox1->Text = currForm1->GetValeur();
}
```

Voilà pour le premier point.
Il reste encore à faire une chose pour que ça compile, c'est faire une déclaration anticipée de Form1 dans le fichier Form2.h.
par exemple, en premier dans le namespace

```
ref class Form1;
```

il restera plus qu'à faire de même dans le fichier form1.h : déclarer form2 de cette facon :

```
ref class Form2;
```

Compilez maintenant, tout est ok
Téléchargez le programme d'exemple

Comment suspendre la capture d'un évènement ?

Auteurs : neguib - nico-pyright(c)

Suite à une action utilisateur sur le clavier, nous devons parfois effectuer un long traitement. Et nous souhaiterions que les actions clavier ne soient pas enregistrées pendant ce traitement. Cette suspension peut être implémentée de la façon suivante :

```
private: System::Void button1_KeyUp(System::Object^  sender, System::Windows::Forms::KeyEventArgs^
 e)
{
 //Exemple après l'appui de la touche "Enter"
 if (e->KeyCode == Keys::Enter)
 {
  //Arrêter la capture d'événements clavier sur le contrôle
  this->button1->KeyUp -= gcnew KeyEventHandler(this, &Form1::button1_KeyUp);
  //Traitements longs

  //Reprendre la capture d'événements clavier sur le contrôle
  this->button1->KeyUp += gcnew KeyEventHandler(this, &Form1::button1_KeyUp);
    }
}
```

Comment rendre un contrôle transparent ?

Auteurs : Ditch - nico-pyright(c)

Dans le constructeur, il suffit d'ajouter :

```
this->SetStyle(ControlStyles::SupportsTransparentBackColor, true);
```

```
this->BackColor = Color::Transparent;
```

après le InitializeComponents().

Comment détecter la fermeture intempestive de l'application ?

Auteurs : neguib - nico-pyright(c)

Parfois, nous avons besoin de distinguer si la fenêtre de l'application a été fermée par un click sur un contrôle prévu à cet effet, ou si le click a été effectué sur la petite croix en haut à droite de la fenêtre.
Pour ce faire, nous avons juste besoin de savoir que l'évènement concerné est lié à l'Application et non au contrôle Form.
Voici un exemple de code test, qui vous démontrera que seule la fermeture intempestive de l'application déclenchera le MessageBox.

```
Application::ApplicationExit += gcnew EventHandler(this, &Form1::Application_Exit);
.....
private: void Application_Exit(System::Object^  sender, System::EventArgs^  e)
{
 MessageBox::Show("Adieu vilaine brute");
}

private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{
 Application::Exit();
}
```

Comment créer des contrôles dynamiquement et gérer leurs événements ?

Auteurs : HULK - neguib

Ce code permet de créer 10 contrôles CheckBox, de les ajouter dans une Listbox et de gérer leur évènements *CheckedChanged*. Il nécessite la présence d'un contrôe ListBox nommé "listBox1" et d'un bouton nommé "BTN".

```
private: System::Void BTN_Click(System::Object^  sender, System::EventArgs^  e)
{
    CheckBox^x;
    for( int i = 1; i <= 10; i++)
    {
        x = gcnew CheckBox();
        x->Name = "Macase" + Convert::ToString(i);
        // on place les CheckBox les unes en dessous des autres
        x->Left = 10;
        x->Top = i * 20;
        x->Width = 150;
        x->Text = "Je suis la case " + Convert::ToString(i);
        // abonner CheckedChanged à la méthode commune MesCasesCheckedChanged
        x->CheckedChanged += gcnew EventHandler(this, &Form1::MesCasesCheckedChanged);
        // ajouter les checkbox à la ListBox
        this->listBox1->Controls->Add(x);
    }
}

private : void MesCasesCheckedChanged(Object^ sender, EventArgs^ e)
```

```
{
    CheckBox^ cbx = safe_cast<CheckBox^>(sender);
    MessageBox::Show("L'état de " + cbx->Name + " est " + (cbx->Checked ? "coché" : "non coché"));
}
```

Comment modifier dynamiquement l'icône d'un NotifyIcon ?

Auteurs : Aspic - neguib

L'intérêt est ici de pouvoir modifier le NotifyIcon de notre application, afin par exemple d'y faire apparaître un texte en fonction de différentes circonstances.
L'exemple implémenté se contentera de faire ici défiler dans l'objet notifyIcon1 les secondes d'un objet DateTime. La procédure est assez simple puisqu'il s'agit de travailler via un objet Bitmap pour la modification et de le convertir en Icon à assigner à la propriété notifyIcon1->Icon.
Ce qu'il ne faut absolument pas oublier c'est de détruire au fur et à mesure les icônes en mémoire.
Le code suivant ne montre que l'essentiel, il vous faut bien sûr en plus déclarer et instancier les objets Font, Brush, Color et StringFormat souhaités.

```
private:
 /// <summary> Méthode privée timer1_Tick
 /// gestionnaire de l'évènement Tick de timer1
 /// </summary>
 System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e)
 {
  // lancer la mise à jour de notifyIcon1
  System::DateTime^ dt = System::DateTime::Now;
  this->UpdateNotiFyIcon((safe_cast<Int32^>(dt->Second))->ToString());
 }
 ///<summary> Méthode privée UpdateNotiFyIcon
 /// chargée de mofifier notifyIcon1 dynamiquement
 ///</summary>
 ///<param name="texte">String : représente le texte à afficher
 ///</param>
 void UpdateNotiFyIcon(String^ texte)
 {
  // redessiner iconBitmap
  this->UpdateBitmap(texte);
  // récupérer un icone à patir de iconBitmap
  System::Drawing::Icon^ newIcon = System::Drawing::Icon::FromHandle(this->iconBitmap->GetHicon());
  // assigner le nouvel icône de NotifyIcon1
  this->notifyIcon1->Icon = newIcon;
  // détruire en mémoire newIcon
  delete(newIcon);
 }
 ///<summary>
 /// Méthode privée chargée de redessiner iconBitmap
 /// en fonction d'un texte à afficher
 ///</summary>
 ///<param name="texte">String : représentant le texte à afficher
 ///</param>
 void UpdateBitmap(String^ texte)
 {
  Graphics^ g = Graphics::FromImage(this->iconBitmap);
  // assigner la couleur de fond
  g->Clear(this->iconBackColor);
  // dessiner le texte
  g->DrawString(texte, this->iconFont,this->iconForeBrush,14,14,this->iconStringFormat);
  // libérer l'objet Graphics
  delete(g);
```

```
}
```

Comment déplacer un contrôle avec la souris ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Pour déplacer un contrôle, par exemple un PictureBox, nous avons besoin de 2 informations :

- **Le bouton gauche vient-il d'être enfoncé ?**
- **La souris se déplace-t-elle en étant au dessus du PictureBox ?**

Le Framework met justement à notre disposition 2 évènements nous permettant de connaître ces 2 informations :

- **MouseDown : notifie si le bouton de la souris a été enfoncé mais aussi à quel endroit;**
- **MouseMove : notifie si la souris se déplace.**

Il suffit donc de traiter le premier évènement afin de connaître la position d'origine du clic et dans le second évènement, nous repositionnerons le PictureBox par rapport au déplacement de la souris.

```
/// <summary> Position de la souris lorsque le bouton a été enfoncé </summary>
Point ^positionClick;
/// <summary>
/// Notifie si le bouton de la souris a été enfoncé sur le PictuBox
/// </summary>
private: System::Void pictureBox1_MouseDown(System::Object^
sender, System::Windows::Forms::MouseEventArgs^  e)
{
 // Vérification si bouton gauche de la souris est bien enfoncé
 if (e->Button == System::Windows::Forms::MouseButtons::Left)
  positionClick = e->Location;
}
/// <summary>
/// Notifie si le curseur se déplace au dessus du PictureBox
/// </summary>
private: System::Void pictureBox1_MouseMove(System::Object^
sender, System::Windows::Forms::MouseEventArgs^  e)
{
 // Vérification si bouton gauche de la souris est bien enfoncé
 if (e->Button == System::Windows::Forms::MouseButtons::Left)
 {
  // Calcul de la nouvelle position du PictureBox
  pictureBox1->Location = Point(pictureBox1->Location.X + e->X - positionClick->X,
    pictureBox1->Location.Y + e->Y - positionClick->Y);
 }
}
```

A noter que ce code peut être utilisé pour les contrôles qui proposent les évènements MouseDown et MouseMove.

Comment charger une image dans un pictureBox tout en libérant les ressources ?

Auteurs : doudouallemand - neguib - nico-pyright(c)

Lors du chargement d'une image dans un PictureBox avec la fonction Image::FromFile, le fichier sous-jascent se trouve en utilisation et donc non disponible jusqu'à la fermeture de l'application. Il est ainsi par exemple impossible de le supprimer.
Pour remédier à ce problème, il faut en conséquence pouvoir libérer la ressource du flux sur ce fichier.
Plusieurs solutions permettent d'atteindre cet objectif. Nous vous proposons ici celui de gérer directement le flux et notamment par l'implémentation de la méthode Image::FromStream.
L'exemple suivant charge une image via un FileStream, l'affecte à un PictureBox (appelé pictureBox1) par Image::FromStream, puis libère les ressources du flux (appelé photoStream) sur le fichier pour pouvoir le supprimer :

```
// Créer le FileStream sur le fichier monimage.jpg
FileStream ^photoStream = gcnew FileStream("C:\\monimage.jpg", FileMode::Open);
// affecter l'image à pictureBox1
pictureBox1->Image = Image::FromStream(photoStream);
// libérer les ressources
photoStream->Close();
// supprimer le fichier monimage.jpg
File::Delete("C:\\monimage.jpg");
```

lien : System.IO.FileStream
lien : Image.FromStream

Que faire avec l'erreur de compilation C3352 ?

Auteur : nico-pyright(c)

Cette erreur survient lorsqu'on essaie de déclarer un délégate, en s'inspirant sur la syntaxe C#.
Si on part de cette syntaxe, on aura tendance à arriver à :

```
delegate void MonDelegateHandler(String ^chaine);
void methodeX ()
{
 MonDelegateHandler ^monDelegate = gcnew MonDelegateHandler(&Form1::MethodeDuDelegate);
}
void MethodeDuDelegate(String ^ chaine)
{
}
```

Ceci est incorrect et produit l'erreur C3352.
La construction correcte est la suivante :

```
MonDelegateHandler ^monDelegate = gcnew MonDelegateHandler(this, &Form1::MethodeDuDelegate);
```

Comment mettre à jour un contrôle d'une winform depuis un thread ?

Auteur : nico-pyright(c)

Il peut arriver qu'on doive mettre à jour un contrôle, comme une barre de progression ou un textbox, lors d'un traitement de longue durée.
Ce traitement doit bien sûr être mis dans un thread pour éviter de bloquer l'application.
Cependant, on ne peut pas directement mettre à jour les contrôles depuis un thread, il faut passer par une méthode particulière. On utilisera un delegate et la méthode invoke.

Voici un exemple pour une barre de progression. Je déclare un délegate et une méthode qui s'occupe de faire avancer ma barre de progression :

```
delegate void ProgressBarDelegateHandler();
ProgressBarDelegateHandler ^ProgressBarDelegate;
void MyUpdateProgressBar()
{
 progressBar1->PerformStep();
}
```

Dans le form_load ou dans le constructeur, on instancie le delegate :

```
ProgressBarDelegate = gcnew ProgressBarDelegateHandler(this, &Form1::MyUpdateProgressBar);
```

Et par exemple sur le click d'un bouton, on initialise la barre de progression et on démarre le thread qui ici, ne fait pas grand chose à part attendre une seconde à chaque itération :

```
System::Void button2_Click(System::Object^  sender, System::EventArgs^  e)
{
 this->progressBar1->Minimum = 0;
 this->progressBar1->Maximum = 10;
 this->progressBar1->Value = 0;
 this->progressBar1->Step = 1;

 Threading::Thread ^t =
gcnew Threading::Thread(gcnew Threading::ThreadStart(this, &Form1::ExecuteLongTraitement));
 t->Start();
}

void ExecuteLongTraitement()
{
 for (int i = 0; i < 10 ; i++)
 {
  System::Threading::Thread::Sleep(1000);
  this->Invoke(ProgressBarDelegate);
 }
}
```

Comment forcer une fenêtre à apparaître à l'avant plan ?

Auteur : Jérôme Lambert

Il suffit tout simplement de mettre à true la propriété TopMost de la fenêtre :

```
this->TopMost = true;
```

Comment intercepter n'importe quelle exception non catchée dans une application Windows ?

Auteurs : Jérôme Lambert - nico-pyright(c)

L'objet application offre la possibilité de s'abonner à l'évènement ThreadException permettant d'être notifié lorsqu'un thread génère une exception non interceptée. Etant donné que la fenêtre est gérée par un thread, vous pourrez ainsi éviter tout "plantage" de votre application ou d'effectuer un traitement adapté avant la fermeture de votre application.

```
static void Application_ThreadException(Object ^sender, System::Threading::ThreadExceptionEventArgs ^e)
{
 System::Windows::Forms::MessageBox::Show("L'exeption générée est : " + e->Exception->Message);
}

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
 // Enabling Windows XP visual effects before any controls are created
 Application::EnableVisualStyles();
 Application::SetCompatibleTextRenderingDefault(false);

 Application::ThreadException +=
 gcnew System::Threading::ThreadExceptionEventHandler(Application_ThreadException);

 // Create the main window and run it
 Application::Run(gcnew Form1());
 return 0;
}
```

Comment connaître la longueur en pixel d'une chaîne de caractères ?

Auteurs : Jérôme Lambert - nico-pyright(c)

La classe Graphics possède une méthode MeasureString permettant de connaître la taille exacte d'une chaîne de caractères en fonction de la Font utilisée :

```
System::Drawing::Graphics ^g = textBox1->CreateGraphics();
System::Drawing::SizeF ^objSizeF = g->MeasureString(textBox1->Text, textBox1->Font);
MessageBox::Show(String::Format("Votre texte '{0}' a une longueur de {1} pixels.", textBox1->Text,
 objSizeF->Width));
```

lien : System.Graphics.MeasureString

Comment lister toutes les forms d'un projet ?

Auteurs : Aspic - nico-pyright(c)

On pourra utiliser la reflexion pour savoir si un type est de type Form.

```
List<Form ^> ^ ListerForms()
{
 List<Form ^> ^ resultat = gcnew List<Form ^> ();
 Reflection::Assembly ^a = System::Reflection::Assembly::GetAssembly(GetType());
 for each (Type ^t in a->GetTypes())
 {
  if (Form::typeid->IsAssignableFrom(t))
```

```
    {
     Form ^f = (Form ^)Activator::CreateInstance(t);
     resultat->Add(f);
    }
   }
   return resultat;
  }
```

Comment utiliser un raccourci clavier sur une form pour effectuer une action ?

Auteur : nico-pyright(c)

il faut mettre en préambule l'attribut KeyPreview de la form à true, pour dire que la form doit traiter tous les événements clavier en premier et ensuite surcharger le KeyDown de la form.
Exemple pour faire un raccourci CTRL+M :

```
System::Void Form1_KeyDown(System::Object^  sender, System::Windows::Forms::KeyEventArgs^  e)
{
    if (e->Control && e->KeyCode == Keys::M)
    {
        MessageBox::Show("Control + M");
    }
}
```

Comment fermer un formulaire en fondu ?

Auteurs : jpjp507 - nico-pyright(c)

Il est facile de fermer un formulaire en fondu. Il suffit de jouer sur l'opacité de la form :

```
for (double dblOpacity = 1; dblOpacity > 0; dblOpacity += -0.05)
{
 Opacity = dblOpacity;
 //laisse le formulaire se recolorer
 Refresh();
 //crée un délai
 System::Threading::Thread::Sleep(50);
}
Close();
```

Comment effectuer un binding bidirectionnel ?

Auteur : nico-pyright(c)

Le binding bidirectionnel consiste à "associer" une propriété de classe à un controle de Formulaire.
Par exemple, j'ai un textbox sur ma form qui est associé à une chaine (String) dans ma classe. Toute modification de ce textbox entraine automatiquement une modification de ma chaine. Inversement, toute modification de cette chaine en code, implique une répercution visuelle sur la valeur du textbox.
Le principe est d'utiliser l'interface INotifyPropertyChanged. L'appel de l'événement PropertyChanged permet d'informer que la valeur a changé et automatiquement répercuter ce changement.
Tout d'abord, la form doit implémenter INotifyPropertyChanged

```
public ref class Form1 : public System::Windows::Forms::Form, INotifyPropertyChanged
```

Ceci implique de définir l'événement suivant

```
virtual event PropertyChangedEventHandler ^PropertyChanged;
```

Il nous faut ensuite une propriété de type String. C'est cette propriété qui sera bindée à notre TextBox :

```
private:
  String ^_chaine;
public:
  property String ^Chaine
  {
    String ^ get() { return _chaine; }
    void set(String ^value)
    {
      if (value != _chaine)
      {
        _chaine = value;
        NotifyPropertyChanged("Chaine");
      }
    }
  }
```

On note l'appel à la méthode NotifyPropertyChanged pour indiquer une changement de valeur.

```
private:
  void NotifyPropertyChanged(String ^info)
  {
    PropertyChanged(this, gcnew PropertyChangedEventArgs(info));
  }
```

La méthode NotifyPropertyChanged appelle l'événement PropertyChanged pour signaler le changement afin qu'il soit répercuté à tous les endroits où le binding est effectué.
Ne pas oublier de définir le binding, au plus tot, dans le form_load par exemple :

```
textBox1->DataBindings->Add("Text", this, "Chaine");
```

Comment écrire des informations dans une console dans mon projet Windows Forms ?

Auteur : nico-pyright(c)

Pour faire ceci, il faut déjà disposer d'un projet Console. Pour changer son projet en un projet console, il faut ouvrir les propriétés du projet, dans Linker -> System, il faut régler le subsystem à Console.
Ensuite il faut appeler :

```
AttachConsole( ATTACH_PARENT_PROCESS );
```

Pour le permettre, il faudra inclure windows.h :

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
```

De manière classique, on pourra afficher une form ainsi :

```
Application::EnableVisualStyles();
Application::SetCompatibleTextRenderingDefault( false );
Application::Run(gcnew testConsoleWinform::Form1());
```

Tout appel à Console::WriteLine sera désormais écrit dans la console.

Comment intercepter les touches du clavier dans mon TextBox ?

Auteurs : abelman - nico-pyright(c)

On se sert des événements de touches. Ils se produisent lorsque le TextBox a le focus dans l'ordre suivant :

- **KeyDown :**
  Une touche a été enfoncée
- **KeyPress :**
  Déclenché si la touche enfoncée représente un caractère. (Il n'est pas déclenché pour les touches F1, F2 par exemple)
- **KeyUp :**
  Une touche a été relâchée

Voici un exemple d'utilisation.
Créer un projet WinForm.
Ajouter deux TextBox à la form principale de votre projet.
Le code suivant utilise une form nommée Form1, et deux TextBox nommés textBox1 et textBox2.
Toutes les touches frappées sur le textBox1 se répercutent sur textBox2.

```
private: System::Void button1_KeyUp(System::Object^  sender, System::Windows::Forms::KeyEventArgs^
 e)
{
 //Exemple après l'appui de la touche "Enter"
 if (e->KeyCode == Keys::Enter)
 {
  //Arrêter la capture d'événements clavier sur le contrôle
  this->button1->KeyUp -= gcnew KeyEventHandler(this, &Form1::button1_KeyUp);
  //Traitements longs

  //Reprendre la capture d'événements clavier sur le contrôle
  this->button1->KeyUp += gcnew KeyEventHandler(this, &Form1::button1_KeyUp);
 }
}
private: System::Void textBox1_KeyPress(System::Object^
 sender, System::Windows::Forms::KeyPressEventArgs^  e)
{
 // On affiche tous les caractères imprimables
 if (!Char::IsControl(e->KeyChar))
  textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart) +
   e->KeyChar + textBox1->Text->Substring(textBox1->SelectionStart + textBox1->SelectionLength);
}
private: System::Void textBox1_KeyDown(System::Object^
 sender, System::Windows::Forms::KeyEventArgs^  e)
{
 // Gestion Touche Back
 if (e->KeyCode == Keys::Back && textBox1->Text->Length > 0)
 {
  if (textBox1->SelectionLength > 0)
  {
   // Suppression sélection
   textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart) +
     textBox1->Text->Substring(textBox1->SelectionStart + textBox1->SelectionLength);
  }
  else if (textBox1->SelectionStart > 0)
  {
   // Suppression caractère précédant le curseur
   if (textBox1->SelectionStart == textBox1->Text->Length)
    textBox2->Text = textBox1->Text->Substring(0, textBox1->Text->Length-1);
   else
```

```
      textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart-1) +
         textBox1->Text->Substring(textBox1->SelectionStart +
         textBox1->SelectionLength);
    }
   }
   // Touche Delete (suppr)
   else if (e->KeyCode == Keys::Delete &&  textBox1->Text->Length > 0)
   {
    // Le curseur est en fin de chaîne
    if (textBox1->SelectionStart == textBox1->Text->Length)
    {
     // Suppression dernier caractère par Shift+Del
     if (e->Shift)
      textBox2->Text = textBox1->Text->Substring(0, textBox1->Text->Length-1);
    }
    else
    {
     // On prend tous les caractères à gauche du curseur
     textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart);
     if (textBox1->SelectionLength != 0)
      // Suppression de la selection
      textBox2->AppendText(textBox1->Text->Substring(textBox1->SelectionStart +
        textBox1->SelectionLength));
     else
     {
      // Si la touche control est enfoncée, tous les caractères
      // à droite du curseur seront supprimés. Sinon on en supprime
      // un seul
      if (!e->Control)
       textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart) +
         textBox1->Text->Substring(textBox1->SelectionStart+1);
     }
    }
   }
   // Coller (Ctrl+V) ou (Shift+insert).
   else if ((e->Shift && e->KeyCode == Keys::Insert) || (e->Control && e->KeyCode == Keys::V))
   {
    // Données dans presse papier
    IDataObject ^cpdata = Clipboard::GetDataObject();
    // Test si cpdata contient du texte
    if (cpdata != nullptr && cpdata->GetDataPresent(String::Empty->GetType()))
    {
     String ^data = cpdata->GetData(String::Empty->GetType())->ToString();
     bool print = false;
     // Gestion caractères non imprimables (comme les tabulations par exemple)
     for (int i=0; i<data->Length-1; i++)
     {
      if (Char::IsControl(data, i) && print)
      {
       data = data->Substring(0, i);
       break;
      }
      else if (!Char::IsControl(data, i) && !print)
       print = true;
     }
     textBox2->Text = textBox1->Text->Substring(0, textBox1->SelectionStart) +
        data + textBox1->Text->Substring(textBox1->SelectionStart + textBox1->SelectionLength);
    }
   }
  }
```

Comment ne saisir que des caractères numériques dans mon TextBox ?

Auteurs : abelman - nico-pyright(c)

On se sert de l'événement KeyPress pour intercepter les caractères entrés dans le TextBox.
La propriété Handled de la classe KeyPressEventArgs indique à l'application ce qu'il faut faire du caractère intercepté.
Si elle vaut false, le traitement par défaut du caractère (l'affichage pour les caractères imprimables) est appliqué.
Si elle vaut true, c'est votre code qui décide ce qu'il faut faire du caractère. Si vous ne faites rien, il ne sera pas affiché. Sa valeur par défaut est false.
Exemple simple :

```
private: System::Void textBox1_KeyPress(System::Object^
 sender, System::Windows::Forms::KeyPressEventArgs^  e)
{
 if (!Char::IsDigit(e->KeyChar))
 // Tous les caractères non numériques ne sont pas traités sur le TextBox.
  e->Handled = true;
}
```

Remarque: Avec ce code, des touches générant un caractère non imprimable (comme la touche BACK) n'auront aucun effet sur le textbox.
Si vous voulez faire un véritable contrôle TexBox numérique, vous devez en tenir compte, gérer le copier-coller CTRL+C et SHIFT+INS avec l'événement KeyDown, et aussi avec le clic droit (menu contextuel coller) sur la souris.

Comment assurer la saisie de nombres corrects dans une textbox avec les exceptions ?

Auteurs : doccpu - nico-pyright(c)

Dans le Validating du ou des TextBox : Pour des nombres Réels :

```
private: System::Void textBox1_Validating(System::Object^
 sender, System::ComponentModel::CancelEventArgs^  e)
{
 if (sender->GetType() == TextBox::typeid)
 {
  TextBox ^T = safe_cast<TextBox ^>(sender);
  try
  {
   Double::Parse(T->Text);
   epErrorProvider->SetError(T, "");
  }
  catch (ArgumentNullException^)
  {
   epErrorProvider->SetError(T, "La case ne peut être vide !");
   T->SelectAll();
   e->Cancel = true;
  }
  catch (OverflowException^)
  {
   epErrorProvider->SetError(T, "Le nombre est trop grand !");
```

```
    T->SelectAll();
    e->Cancel = true;
   }
   catch (FormatException^)
   {
    epErrorProvider->SetError(T, "Le format n'est pas correct");
    T->SelectAll();
    e->Cancel = true;
   }
  }
}
```

Pour les nombres décimaux :

```
private void TextBox_Validating(object sender, CancelEventArgs e)
{
     if (sender->GetType() == TextBox::typeid)
     {
        TextBox ^T = safe_cast<TextBox ^>(sender);
        try
        {
           Decimal::Parse(T->Text);
           epErrorProvider->SetError(T, "");
        }
        catch (ArgumentNullException^)
        {
           epErrorProvider->SetError(T, "La case ne peut être vide !");
           T->SelectAll();
           e->Cancel = true;
        }
        catch (OverflowException^)
        {
           epErrorProvider->SetError(T, "Le nombre est trop grand !");
           T->SelectAll();
           e->Cancel = true;
        }
        catch (FormatException^)
        {
           epErrorProvider->SetError(T, "Le format n'est pas correct");
           T->SelectAll();
           e->Cancel = true;
        }
     }
}
```

N'oubliez pas de mettre le bon caractère comme séparateur decimal :

```
private: System::Void textBox1_KeyPress(System::Object^
 sender, System::Windows::Forms::KeyPressEventArgs^  e)
 {
 // stoque le séparateur décimal du système
 wchar_t
 Separateur = System::Globalization::CultureInfo::CurrentCulture->NumberFormat->NumberDecimalSeparator[0];
 // dans le cas de l'ecriture d'un séparateur
 if ((e->KeyChar == '.') || (e->KeyChar == ','))
 {
  // Force l'ecriture du bon séparateur
  e->KeyChar = Separateur;
 }
 }
```

Pour des nombres Entiers :

```
private void TextBox_Validating(object sender, CancelEventArgs e)
{
     if (sender->GetType() == TextBox::typeid)
     {
  TextBox ^T = safe_cast<TextBox ^>(sender);
  try
  {
   Integer::Parse(T->Text);
   epErrorProvider->SetError(T, "");
  }
  catch (ArgumentNullException^)
  {
   epErrorProvider->SetError(T, "La case ne peut être vide !");
   T->SelectAll();
   e->Cancel = true;
  }
  catch (OverflowException^)
  {
   epErrorProvider->SetError(T, "Le nombre est trop grand !");
   T->SelectAll();
   e->Cancel = true;
  }
  catch (FormatException^)
  {
   epErrorProvider->SetError(T, "Le format n'est pas correct");
   T->SelectAll();
   e->Cancel = true;
  }
 }
}
```

Comment assurer la saisie de nombres corrects dans une textbox avec les expressions régulières ?

Auteurs : efficks - nico-pyright(c)

Dans le Validating du ou des TextBox, en utilisant la classe System::Text::RegularExpressions :

```
private: System::Void textBox1_Validating(System::Object^
 sender, System::ComponentModel::CancelEventArgs^  e)
 {
 TextBox ^ txtChamp = safe_cast<TextBox^>(sender);
 Regex ^rexValideur = gcnew Regex("expression");
 if(rexValideur->IsMatch(txtChamp->Text))
 {
  epErrorProvider->SetError(txtChamp, "");
 }
 else
 {
  epErrorProvider->SetError(txtChamp, "Valeur du champ invalide.");
  e->Cancel = true;
 }
}
```

Il faut ensuite remplacer expression par une chaîne de caractère représentant une expression régulière. Cette méthode est plus recommandée que les try{}catch{} car ceux-ci utilisent beaucoup plus de ressources.

Comment mettre en place un système de suggestion ?

Auteur : Jérôme Lambert

La classe TextBox donne la possibilité de compléter automatiquement les saisies de l'utilisateur… Un peu comme ce que fait déjà votre navigateur.
Pour cela, vous devez utiliser la collection AutoCompleteCustomSource pour spécifier la liste des valeurs que le Textbox pourra proposer.

```
// Activer l'autocompletion
textBoxRecherche->AutoCompleteSource = AutoCompleteSource::CustomSource;
textBoxRecherche->AutoCompleteCustomSource->AddRange(gcnew
 array<String^>{"Chat", "Cheval", "Chien"});
```

En plus de cela, il est possible de préciser comment le Textbox doit afficher les propositions : Dans une liste (Suggest), directement dans le textbox (Append) ou un mix des 2 solutions précédentes (SuggestAppend).

```
textBoxRecherche->AutoCompleteMode = AutoCompleteMode::SuggestAppend;
```

Comment placer le curseur à la fin d'un textbox multiligne ?

Auteur : nico-pyright(c)

Quand on rajoute du texte à un textbox multiligne avec, par exemple :

```
for (int i = 0 ; i < 100 ; i++)
 textBox1->Text += "abc" + Convert::ToString(i) + Environment::NewLine;
```

le texte s'ajoute bien, mais pour peu que la longueur du texte dépasse la zone visible, on ne voit pas la fin du texte comme il est souvent d'usage.
Pour y remédier, on peut utiliser ces deux lignes de code pour renvoyer le curseur à la fin et provoquer le défilement jusqu'à celui-ci.

```
textBox1->SelectionStart = textBox1->TextLength;
textBox1->ScrollToCaret();
```

On peut aussi envisager de mettre le focus sur le textbox si besoin :

```
textBox1->Focus();
```

Sommaire > Intéraction du C++/CLI avec le framework .Net > WinForms > TreeView

Comment capturer un Click dans l'icone d'un TreeNode ?

Auteurs : neguib - nico-pyright(c)

Utilisons un TreeView appelé ici "TreeView1" possédant nécessairement dans ce cas de figure une ImageList pour sa collection de TreeNode.
La capture de l'événement Click se fait ici par la méthode TreeView->MouseDown afin de récupérer les coordonnées du pointeur de la souris.
La méthode ci-dessous GetNodeIconeRectangle permet de calculer le Rectangle de l'icone du TreeNode sélectionné. Il ne reste plus alors qu'à vérifier si le Point localisé du pointeur de la souris appartient à l'icone du TreeNode sélectionné.

```
private: System::Void treeView1_MouseDown(System::Object^
 sender, System::Windows::Forms::MouseEventArgs^  e)
{
 Point p(e->X, e->Y);
 TreeNode ^selectedTreeNode = this->treeView1->GetNodeAt(p);
 Rectangle ^imgRectangle;
 if (selectedTreeNode !=nullptr)
 {
  imgRectangle = GetNodeIconeRectangle(selectedTreeNode);
  if (imgRectangle->Contains(p))
  {
   //traitement
  }
 }
}
private : Rectangle ^GetNodeIconeRectangle(TreeNode ^node)
{
 System::Drawing::Size^ s = node->TreeView->ImageList->ImageSize;
 int h = node->TreeView->ItemHeight;
 Rectangle ^r = node->Bounds;
 int x = r->X - s->Width;
 int y = r->Y + ((h - s->Height)/2);
 Point p(x,y);
 return gcnew Rectangle(p, *s);
}
```

Comment charger l'arborescence de son disque dur dans un TreeView ?

Auteurs : HULK - neguib

Ce code permet de charger l'arborescence du disque dur C:\ dans un TreeView de la même façon que l'Explorateur Windows.
Ce code nécessite un contrôle TreeView nommé TV et un Button nommé BTN.

```
private: System::Void BTN_Click(System::Object^  sender, System::EventArgs^  e)
{
    TV->Nodes->Add("C:\\");
    this->Explorer(TV->Nodes[0]);
}

private: void Explorer(TreeNode^ node)
{
    try
    {
        node->Nodes->Clear();
```

```
        for each(String^ s in Directory::GetDirectories(node->FullPath))
        {
            node->Nodes->Add(Path::GetFileName(s));
        }
    }
    catch(Exception ^) {}
}

private: System::Void TV_AfterExpand(System::Object^
 sender, System::Windows::Forms::TreeViewEventArgs^  e)
{
    for each(TreeNode^ tn in e->Node->Nodes)
        this->Explorer(tn);
}
```

Comment déterminer le TreeNode survolé par la souris ?

Auteur : Jérôme Lambert

Il faut pour cela traiter l'évènement MouseMove du TreeView :

```
      /// <summary>
      /// Evènement MouseMove du treeView1
      /// </summary>
 private: System::Void treeView1_MouseMove(System::Object^
 sender, System::Windows::Forms::MouseEventArgs^  e)
 {
  // Récupération du TreeNode survolé
  TreeNode ^currentNode = treeView1->GetNodeAt(e->X, e->Y);
  // Vérification que la souris survole bien un TreeNode
  if (currentNode != nullptr)
   textBox1->Text = "Le node survolé est " + currentNode->Name;
 }
```

Comment colorier une cellule en mode Details ?

Auteurs : neguib - nico-pyright(c)

L'objet ListViewSubItem a une propriété BackColor mais pour que l'assignation soit effective il ne faut pas oublier de modifier la propriété UseItemStyleForSubItems du ou des ListViewItem concerné(s) en lui attribuant la valeur false.
Si vous souhaitez effectuer cette manoeuvre par programmation, voici un exemple :
Coloration en jaune de toute la première colonne d'un ListView1.

```
for each (ListViewItem ^lvi in this->listView1->Items)
{
 lvi->UseItemStyleForSubItems = false;
 lvi->SubItems[0]->BackColor = Color::Yellow;
}
this->listView1->View = View::Details;
```

Comment trier les colonnes d'une listview en cliquant sur leurs intitulés ?

Auteur : nico-pyright(c)

L'objet ListView dispose d'une propriété ListViewItemSorter de type IComparer. Cela permet d'implémenter sa propre gestion du tri.
Pour ce faire, il faut créer une classe dérivant de IComparer implémentant la méthode Compare.

```
ref class ListViewItemComparer: public IComparer
{
private:
 int col;
 SortOrder sortOrder;
public:
 ListViewItemComparer()
 {
  col = 0;
  sortOrder = SortOrder::Ascending;
 }
 ListViewItemComparer( int column )
 {
  col = column;
  sortOrder = SortOrder::Ascending;
 }
 ListViewItemComparer( int column, SortOrder s )
 {
  col = column;
  sortOrder = s;
 }
 virtual int Compare( Object^ x, Object^ y )
 {
  if (sortOrder == SortOrder::Ascending)
   return String::Compare(safe_cast<ListViewItem^>(x)->SubItems[col]->Text,
     safe_cast<ListViewItem^>(y)->SubItems[col]->Text);
  else
   return String::Compare(safe_cast<ListViewItem^>(y)->SubItems[col]->Text,
     safe_cast<ListViewItem^>(x)->SubItems[col]->Text);
 }
};
```

Ensuite, il suffit d'intercepter le click sur les colonnes *(événement ColumnClick)* et d'instancier cette classe.

```
private: System::Void listView1_ColumnClick(System::Object^
 sender, System::Windows::Forms::ColumnClickEventArgs^  e)
{
 if (this->listView1->Sorting == SortOrder::Ascending)
  this->listView1->Sorting = SortOrder::Descending;
 else
  this->listView1->Sorting = SortOrder::Ascending;
 this->listView1->ListViewItemSorter =
 gcnew ListViewItemComparer( e->Column, this->listView1->Sorting );
}
```

Remarque : J'effectue ici une comparaison de chaînes, il faudra implémenter différemment si on compare des nombres par exemple.

Comment écrire un label avec plusieurs couleurs ?

Auteur : nico-pyright(c)

Il n'est pas possible d'écrire le texte d'un label avec plusieurs couleurs par défaut, mais on peut y arriver en créant un nouveau contrôle qui dérive de Label et en surchargeant la méthode OnPaint.

Tout d'abord, créer une classe dérivée de Label et surcharger OnPaint :

```
ref class MonLabel : public System::Windows::Forms::Label
{
protected:
 virtual void OnPaint(PaintEventArgs ^e) override
 {

 array<Drawing::Brush ^> ^mesCouleurs = { Brushes::Red, Brushes::Orange, Brushes::Green, Brushes::Blue};
  int br = 0;
  String ^chaine = Text;

  List<String ^> ^listString = gcnew List<String ^>();
  StringBuilder ^sb = gcnew StringBuilder();
  for (int i = 0 ; i < chaine->Length ; i++)
  {
   if (chaine[i] == ' ' || chaine[i] == '\'')
   {
    if (chaine[i] == '\'')
     sb->Append(chaine[i]);
    listString->Add(sb->ToString());
    sb = gcnew StringBuilder();
   }
   else
    sb->Append(chaine[i]);
  }
  listString->Add(sb->ToString());

  float startX = 0;
  for each (String ^s in listString)
  {
   e->Graphics->DrawString(s, Font, mesCouleurs[br], startX, 0);
   startX += e->Graphics->MeasureString(s, Font).Width;
   br++;
   if (br >= mesCouleurs->Length)
    br = 0;
  }
  Width = (int)startX;
 }
};
```

Ici, j'ai choisi de changer de couleur à chaque espace trouvé ou à chaque ' trouvée. J'utilise DrawString pour écrire la partie de mot avec la couleur choisie, et je n'oublie pas de mesurer sa taille pour écrire la partie de mot suivante.
Il faut également adapter la taille du contrôle à la nouvelle taille du texte.

Ensuite, il ne reste plus qu'à utiliser notre contrôle, par exemple dans le constructeur :

```
MonLabel ^ monLabel = gcnew MonLabel();
AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
monLabel->Font =
 (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.0F, System::Drawing::FontStyle::Regular,
  System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
monLabel->Text = "Je m'appelle Nico-pyright(c)";
```

```
this->Controls->Add(monLabel);
```

Comment empêcher le sous-lignement du caractére de raccourci clavier d'un Button ?

Auteurs : nico-pyright(c) - olsimare

Il existe la propriété ShowKeyboardCues héritée de Control qui permet de déterminer si le sous-lignement du caractére de raccourci clavier d'un Button doit être affiché ou pas.
Cette propriété étant ReadOnly, il est nécessaire de créer un Button personnalisé qui la substitue.

```
public ref class NonKeyboardCuesButton : public Button
{
public:
 virtual property bool ShowKeyboardCues
 {
  bool get() override { return false; }
 }
};
```

Comment empêcher l'affichage du rectangle de focus d'un Button ?

Auteurs : nico-pyright(c) - olsimare

Il existe la propriété ShowFocusCues héritée de Control qui permet de déterminer si le rectangle de focus doit être affiché ou pas.
Cette propriété étant ReadOnly, il est nécessaire de créer un Button personnalisé qui la substitue.

```
public ref class NonFocusCuesButton : public Button
{
public:
 virtual property bool ShowFocusCues
 {
  bool get() override { return false; }
 }
};
```

Sommaire > Intéraction du C++/CLI avec le framework .Net > GDI

Qu'est ce que le GDI+ ?

Auteur : Jérôme Lambert

Cette définition est tirée de MSDN
Le Common Language Runtime recourt à une implémentation avancée de l'interface graphique (GDI, Graphics Design Interface) Windows, appelée GDI+. GDI+ vous permet de créer des graphiques, de dessiner du texte et de manipuler des images graphiques en tant qu'objets. Cette interface est conçue pour allier performances et simplicité d'utilisation. Vous pouvez l'utiliser en vue du rendu des images graphiques sur des Windows Forms et des contrôles. GDI+ a entièrement remplacé GDI et constitue aujourd'hui la seule option disponible pour le rendu des graphiques par programme dans les applications Windows Forms.

lien : Graphiques GDI+

Comment implémenter un PrintScreen rapidement ?

Auteurs : HULK - neguib

Ce code permet de réaliser simplement une capture d'écran de la fenêtre active et de l'écran en entier.
Le principe est de simuler l'appui sur la touche "Impr écran" puis de récupérer l'image obtenue dans le Presse-Papier. Pour l'exemple donné ci-dessous, les résultats sont affichés dans 2 PictureBox : PBWindow, PBScreen après le click sur un Button BPrint.

```
private: System::Void BPrint_Click(System::Object^  sender, System::EventArgs^  e)
{
 Bitmap^ screen;
 Bitmap^ window;

 // simuler l'appui de la touche PrintScreen
 System::Windows::Forms::SendKeys::SendWait("{PRTSC}");
 // récupérer l'image obtenue dans le Presse-Papier
 window = safe_cast<Bitmap^>(Clipboard::GetDataObject()->GetData("Bitmap"));
 // attribuer l'image à PBWindow
 PBWindow->Image = window;

 // simuler le PrintScreen enrichi
 System::Windows::Forms::SendKeys::SendWait("+{PRTSC}");
 // récupérer l'image obtenue dans le Presse-Papier
 screen = safe_cast<Bitmap^>(Clipboard::GetDataObject()->GetData("Bitmap"));
 //attribuer l'image à PBScreen
 PBScreen->Image = screen;
}
```

Comment convertir une image en tableau de bytes ?

Auteurs : nico-pyright(c) - sam_XIII

Voici une méthode permettant de convertir une image (ici au format Jpeg) en un tableau de bytes :

```
static array<unsigned char> ^ Image2ByteArray(Image ^img)
{
    try
    {
        MemoryStream ^mstImage  = gcnew MemoryStream();
        img->Save(mstImage, System::Drawing::Imaging::ImageFormat::Jpeg);
```

```
        array<unsigned char> ^ bytImage = mstImage->GetBuffer();
        return bytImage;
    }
    catch(Exception ^)
    {
        return nullptr;
    }
}
```

Comment convertir un tableau de bytes en image ?

Auteurs : nico-pyright(c) - sam_XIII

Voici une méthode permettant de convertir une image (ici au format Jpeg) en un tableau de bytes :

```
static Image ^ByteArray2Image(array<unsigned char> ^ BArray)
{
    try
    {
        MemoryStream ^mstImage = gcnew MemoryStream(BArray);
        Image ^img = Image::FromStream(mstImage);
        return img;
    }
    catch(Exception ^)
    {
        return nullptr;
    }
}
```

Sommaire > Intéraction du C++/CLI avec le framework .Net > Fichiers, Répertoires, Disques

Comment créer, copier, déplacer, supprimer un fichier ?

Auteurs : abelman - nico-pyright(c)

Pour créer, copier, déplacer ou supprimer un fichier, on utilise la classe System::IO::File

```
using namespace System::IO;
using namespace System;

public: static void FileTests()
{
 try
 {
  //Création d'un fichier vide.
  FileStream ^fs = File::Create("myfile.txt");

  fs->Close();
  Console::WriteLine("fichier myfile.txt créé");

  // Copie de fichier
  File::Copy("myfile.txt", "copyofmyfile.txt");
  Console::WriteLine("fichier myfile.txt copié vers copyofmyfile.txt");

  // Déplacement de fichier
  File::Move("copyofmyfile.txt", "c:\\copyofmyfile.txt");
  Console::WriteLine("fichier copyofmyfile.txt déplacé vers c:\\copyofmyfile.txt");

  // Suppression de fichier
  File::Delete("c:\\copyofmyfile.txt");
  Console::WriteLine("Fichier c:\\copyofmyfile.txt supprimé");
 }
 catch  (Exception ^ex)
 {
  Console::WriteLine(ex->ToString());
  Console::WriteLine(ex->Message);
 }
}
```

Comment obtenir les attributs d'un fichier ou d'un répertoire?

Auteurs : abelman - nico-pyright(c)

On utilise la méthode GetAttributes de la classe System::IO::File

```
static void GetFileAttributes(String ^sFilename)
{
 FileAttributes flagAttr = File::GetAttributes(sFilename);

 // Date de création
 Console::WriteLine("Créé le {0} à {1}",
  File::GetCreationTime(sFilename).ToShortDateString(),
  File::GetCreationTime(sFilename).ToShortTimeString());

 // Date de la dernière modification
 Console::WriteLine("Modifié le {0} à {1}",
  File::GetLastWriteTime(sFilename).ToShortDateString(),
  File::GetLastWriteTime(sFilename).ToShortTimeString());

 // Date du dernier accès
```

```
    Console::WriteLine("Dernier accès le {0} à {1}",
     File::GetLastAccessTime(sFilename).ToShortDateString(),
     File::GetLastAccessTime(sFilename).ToShortTimeString());

    Console::WriteLine("Attributs de {0}", sFilename);

    // Attribut Archive
    if ((flagAttr & FileAttributes::Archive) == FileAttributes::Archive)
     Console::WriteLine(FileAttributes::Archive);

    // Attribut Compressé
    if ((flagAttr & FileAttributes::Compressed) == FileAttributes::Compressed)
     Console::WriteLine(FileAttributes::Compressed);

    // Attribut Device
    if ((flagAttr & FileAttributes::Device) == FileAttributes::Device)
     Console::WriteLine(FileAttributes::Device);

    if ((flagAttr & FileAttributes::Directory) == FileAttributes::Directory)
     Console::WriteLine(FileAttributes::Directory);

    if ((flagAttr & FileAttributes::Encrypted) == FileAttributes::Encrypted )
     Console::WriteLine(FileAttributes::Encrypted);

    // Attribut caché
    if ((flagAttr & FileAttributes::Hidden) == FileAttributes::Hidden)
     Console::WriteLine(FileAttributes::Hidden);

    // Attribut Normal
    if ((flagAttr & FileAttributes::Normal) == FileAttributes::Normal)
     Console::WriteLine(FileAttributes::Normal);

    // Attibut non indexé
    if ((flagAttr & FileAttributes::NotContentIndexed) == FileAttributes::NotContentIndexed)
     Console::WriteLine(FileAttributes::NotContentIndexed);

    // Attribut Offline
    if ((flagAttr & FileAttributes::Offline) == FileAttributes::Offline)
     Console::WriteLine(FileAttributes::Offline);

    // Attribut ReadOnly
    if ((flagAttr & FileAttributes::ReadOnly) == FileAttributes::ReadOnly)
     Console::WriteLine(FileAttributes::ReadOnly);

    // Attribut ReparsePoint
    if ((flagAttr & FileAttributes::ReparsePoint) == FileAttributes::ReparsePoint)
     Console::WriteLine(FileAttributes::ReparsePoint);

    // Attribut SparseFile
    if ((flagAttr & FileAttributes::SparseFile) == FileAttributes::SparseFile)
     Console::WriteLine(FileAttributes::SparseFile);

    // Attribut System
    if ((flagAttr & FileAttributes::System) == FileAttributes::System)
     Console::WriteLine(FileAttributes::System);

    // Attribut Temporary
    if ((flagAttr & FileAttributes::Temporary) == FileAttributes::Temporary)
     Console::WriteLine(FileAttributes::Temporary);
   }
```

Comment obtenir la liste des fichiers et des sous-répertoires d'un répertoire ?

Auteurs : abelman - nico-pyright(c)

On utilise la fonction GetFileSystemEntries de la classe System::IO::Directory

```
static void Dir(String ^directory)
{
 array<String ^> ^ files;

 // pour avoir les noms des fichiers et sous-répertoires
 files = Directory::GetFileSystemEntries(directory);

 for each (String ^file in files)
  Console::WriteLine(file);
}
```

Pour avoir juste les noms des fichiers et pas les sous-répertoires d'un répertoire, on utilise System::IO::Directory::GetFiles()

Comment lire et écrire dans un fichier texte ?

Auteurs : abelman - nico-pyright(c)

**Nous allons ouvrir un fichier texte et le remplir s'il n'existe pas encore.
Nous afficherons ensuite son contenu à l'écran.
On utiliser pour cela les classes System::IO::StreamReader pour la lecture et System::IO::StreamWriter pour l'écriture.**

```
void FichierTexte(String ^nomFichier)
{
 StreamReader ^sr;
 StreamWriter ^sw;
 String ^line;
 try
 {
  if (! File::Exists(nomFichier))
  {
   // Le fichier n'existe pas. On le crée.
   sw = gcnew StreamWriter(nomFichier);
   sw->WriteLine("Bonjour. Nous sommes le {0} et il est {1} ",
    DateTime::Now.ToLongDateString(),
    DateTime::Now.ToLongTimeString());
   sw->Close();
   // Remarque : On peut utiliser sw = File::AppendText(NomFichier) pour ajouter
   // du texte à un fichier existant
  }
  // Ouverture du fichier et écriture du contenu du fichier sur la console
  sr = gcnew StreamReader(nomFichier);
  Console::WriteLine("Début du fichier");
  line = sr->ReadLine();
  while (line != nullptr)
  {
   Console::WriteLine(line);
   line = sr->ReadLine();
```

```
    }
    // Remarque : on peut aussi utiliser ReadToEnd pour lire tout le fichier en une seule fois
    Console::WriteLine("Fin du fichier");
   }
   finally
   {
    // Fermeture streamreader
    if (sr != nullptr) sr->Close();
    // Fermeture streamwriter
    if (sw != nullptr) sw->Close();
   }
  }
```

Comment lire et écrire dans un fichier binaire ?

Auteurs : abelman - nico-pyright(c)

Nous allons ouvrir un fichier binaire et le remplir (d'entiers) s'il n'existe pas encore.
Nous afficherons son contenu à l'écran.
On utilise pour cela les classes System::IO::BinaryReader pour la lecture et System::IO::BinaryWriter pour l'écriture.

```
void FichierBinaire(String ^NomFichier)
{
 BinaryReader ^br;
 BinaryWriter ^bw;
 FileStream ^fs;
 try
 {
  if (!File::Exists(NomFichier))
  {
   // Le fichier n'existe pas. On le crée
   bw = gcnew BinaryWriter(File::Create(NomFichier));
   for (int i=0; i<10; i++)
    bw->Write(i);
   bw->Close();
  }
  // Ouverture du contenu du fichier et écriture sur la console
  fs = File::Open(NomFichier, FileMode::Open);
  br = gcnew BinaryReader(fs);
  while (fs->Position < fs->Length)
   Console::Write(br->ReadInt32());
  Console::WriteLine("\nFin du fichier");
 }
 finally
 {
  if (br!=nullptr) br->Close();
  if (bw!=nullptr) bw->Close();
 }
}
```

Comment surveiller les modifications d'un fichier grâce aux notifications système?

Auteurs : abelman - nico-pyright(c)

Windows envoie des notifications qui permettent de surveiller les modifications apportées au système de fichier.
Cela se fait de la manière suivante en utilisant la classe System::IO::FileSystemWatcher.

```
void OnChanged(Object ^source, FileSystemEventArgs ^e)
{
 Console::WriteLine("Fichier {0} {1}", e->FullPath, e->ChangeType);
}
void OnRenamed(Object ^source, RenamedEventArgs ^e)
{
 Console::WriteLine("Fichier {0} renommé en {1}", e->OldFullPath, e->FullPath);
}

void OnError(Object ^source, ErrorEventArgs ^e)
{
    Exception ^ex = e->GetException();
 Console::WriteLine(ex->ToString());
}
void Watch(String ^path, String ^filter)
{
    // On peut utiliser les '*' avec filter

    // Création de l'objet watcher
    FileSystemWatcher ^fw = gcnew FileSystemWatcher(path, filter);

    // On ajoute les handlers pour surveiller les événements qu'on souhaite.
    fw->Changed += gcnew FileSystemEventHandler(OnChanged);
    fw->Renamed += gcnew RenamedEventHandler(OnRenamed);
    fw->Created += gcnew FileSystemEventHandler(OnChanged);
    fw->Deleted += gcnew FileSystemEventHandler(OnChanged);
    fw->Error += gcnew ErrorEventHandler(OnError);

    // On surveillera aussi les sous répertoires
    fw->IncludeSubdirectories = true;

    // Mettre EnableRaisingEvents à true démarre la surveillance des modifications.
    // Mettre à false l'arrête.
    fw->EnableRaisingEvents = true;
}
```

Et voici l'utilisation de la fonction Watch :

```
// Pour surveiller l'activité du répertoire c:\rep
Watch("c:\\rep", "*");
```

Comment récupérer le nom d'un fichier, lorsque j'ai le chemin complet ?

Auteurs : nico-pyright(c) - Thomas Lebrun

Vous devez utiliser la méthode GetFileName de la classe System::IO::Path qui vous permet de récupérer le nom du fichier.

```
String ^fileName = "C:\\toto.txt";
String ^name = System::IO::Path::GetFileName(fileName);
Console::WriteLine(name);
```

Comment afficher ou écrire correctement les caractères accentués dans mes fichiers ?

Auteurs : nico-pyright(c) - Thomas Lebrun

**Les fichiers sont ouverts avec l'encodage unicode par défaut dans .NET.
Il en résulte que les caractères accentués par exemple ne s'affichent pas correctement.
Vous devez spécifier le type d'encodage à utiliser, pour la lecture/écriture de votre fichier. Exemple pour la lecture :**

```
using namespace System::IO;
using namespace System::Text;
StreamReader ^strReader = gcnew StreamReader(fileName, Encoding::Default);
```

Comment tester l'existence d'un fichier ?

Auteurs : neo.51 - nico-pyright(c)

```
if (System::IO::File::Exists("c:\\monfichier.txt"))
 //le fichier existe
else
 //le fichier n'existe pas
```

Comment savoir si un fichier est en lecture seule ?

Auteur : Jérôme Lambert

Grâce à la classe File, il est possible de récupérer les attributs (Archive, Lecture seule, ...) d'un fichier :

```
// Récupération des attributs d'un fichier
FileAttributes Fa = File::GetAttributes("C:\\text.txt");
// Vérification si le fichier est en lecture seule
if ((Fa & FileAttributes::ReadOnly) == FileAttributes::ReadOnly)
 Console::WriteLine("Ce fichier est en lecture seule !");
else
 Console::WriteLine("Ce fichier n'est pas en lecture seule");
```

lien : http://msdn.microsoft.com/fr-fr/library/system.io.file(VS.80).aspx

Comment obtenir le répertoire d'exécution de mon application ?

Auteurs : abelman - freegreg - nico-pyright(c)

On utilise la fonction GetCommandLineArgs de la classe System::Environment.

```
// Chemin de l'exécutable
// Pour les WinForms, on peut aussi utiliser Application::ExecutablePath
String ^ exepath = Environment::GetCommandLineArgs()[0];
```

```
// Répertoire de l'exécutable
String ^ exedir = exepath->Substring(0, exepath->LastIndexOf('\\'));
```

Comment obtenir le répertoire courant ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété CurrentDirectory de la classe Environment.

```
Console::WriteLine("Le répertoire courant est : {0}", Environment::CurrentDirectory);
```

Comment obtenir le répertoire système ?

Auteurs : abelman - nico-pyright(c)

On utilise la propriété SystemDirectory de la classe Environment.

```
Console::WriteLine("Le répertoire système est : {0}", Environment::SystemDirectory);
```

Comment obtenir les chemins des répertoires spéciaux comme "Mes Documents" ?

Auteurs : abelman - nico-pyright(c)

On utilise la fonction GetFolderPath de la classe Environment avec l'énumération Environment::SpecialFolder.

```
static void PrintSpecFolder()
{
 // Répertoire spéciaux
 Console::WriteLine("Répertoire spéciaux");
 System::Array ^ sfe = Enum::GetValues(Environment::SpecialFolder::typeid);
 for each (Environment::SpecialFolder s in sfe)
  Console::WriteLine(Environment::GetFolderPath(s));
}
```

Comment obtenir la liste des lecteurs logiques ?

Auteurs : abelman - nico-pyright(c)

On utilise la fonction GetLogicalDrives de la classe Environment.

```
static void PrintLogicalDrives()
{
 // Lecteurs logiques
 Console::WriteLine("Lecteurs logiques");
 array<String ^> ^drives = Environment::GetLogicalDrives();
 for each (String ^drive in drives)
```

```
    Console::WriteLine(drive);
  }
```

Comment obtenir les informations d'un disque ?

Auteurs : merlin - nico-pyright(c)

Il faudra tout d'abord importer l'assembly System.Management. Pour récupérer une valeur, il faudra utiliser la méthode GetPropertyValue associée à un nom de propriété.
L'exemple suivant commence par lister toutes les propriétés existantes puis récupère la valeur de la propriété VolumeSerialNumber.

```
using namespace System;
using namespace System::Management;

ManagementObject ^disque = gcnew ManagementObject("win32_logicaldisk.deviceid=\"c:\"");
disque->Get();

//Afficher toutes les propriétés du disque
String ^strProp;
for each(PropertyData ^d in disque->Properties)
 strProp += d->Name + "\n";
Console::WriteLine(strProp);

//Obtenir une propriété en particulier
strProp = disque->GetPropertyValue("VolumeSerialNumber")->ToString();
Console::WriteLine("n° série volume : " + strProp);
```

Comment connaître le pourcentage d'espace disque de vos disques durs ?

Auteur : Jérôme Lambert

La classe DriveInfo peut nous donner les informations nécessaires afin de calculer le pourcentage d'espace libre de chaque disque dur :

```
// Parcours de la liste des disques durs
for each (DriveInfo ^CurrentDrive in DriveInfo::GetDrives())
{
 // Vérification qu'on a bien affaire à un disque dur de l'ordinateur
 if (CurrentDrive->DriveType == DriveType::Fixed)
 {
  // Calcul du pourcentage d'espace disque libre
  Double pourcentageLibre = safe_cast<Double>(safe_cast<Double>(CurrentDrive->AvailableFreeSpace) /
safe_cast<Double>(CurrentDrive->TotalSize) * 100);
  Console::WriteLine("Espace libre de {0} >> {1}%",
CurrentDrive->Name, Convert::ToInt16(pourcentageLibre));
 }
}
```

Comment détecter si il y a un cd/dvd dans un lecteur ?

Auteur : Jérôme Lambert

On utilise la classe DriveInfo qui fournit des informations concernant entre autres des informations concernant les lecteurs CD/DVD :

```
String ^ monDrive = "D:\\";
try
{
 DriveInfo ^monDriveInfo = gcnew DriveInfo(monDrive);
 // Vérification qu'on a bien affaire à un lecteur CD/DVD
 if (monDriveInfo->DriveType == DriveType::CDRom)
 {
  // Vérification si il y a un cd dans le lecteur
  if (monDriveInfo->IsReady == true)
   Console::WriteLine("Lecteur identifié avec CD : {0}", monDriveInfo->VolumeLabel);
  else
   Console::WriteLine("Lecteur identifié sans CD : {0}", monDriveInfo->Name);
 }
 else
  Console::WriteLine("Ce lecteur n'est pas un lecteur CD/DVD !");
}
catch (Exception ^e)
{
 Console::WriteLine("Ceci n'est pas un lecteur !");
}
```

lien : System.IO.DriveInfo

Comment lister toutes les entrées d'un répertoire ?

Auteur : neguib

L'intérêt est ici d'implémenter de façon appropriée la récursivité. L'espace de noms privilégié est System::IO, notamment sa classe Directory. Pour l'exemple nous allons explorer le répertoire particulier des "Favoris", ce qui nous permet à ce propos de nous inspirer de la Q/R d'abelman : Comment obtenir les chemins des répertoires spéciaux comme 'Mes Documents' ?.

```
static void WriteFileEntries(String^ folder, int indent)
{
 // Créer une indentation du texte
 StringBuilder^ tab = gcnew StringBuilder(String::Empty);
 for(int i = 0; i < indent; i++)
  tab->Append(L" ");
 // Afficher les dossiers présents et leur contenu par récursivité
 array<System::String ^>^ subdirs = Directory::GetDirectories(folder);
 // SI le dossier contient des sous-dossiers
 if (subdirs->Length != 0)
 {
  for each (String^ s in subdirs)
  {
   // Afficher le nom du dossier à explorer
   Console::WriteLine(tab->ToString() + Path::GetFileName(s));
   //Utiliser la récursivité avec une indentation
   WriteFileEntries(s, indent + 3);
```

```
    }
   }
   // Afficher les noms des fichiers présents
   array<System::String ^>^ files = Directory::GetFiles(folder);
   // SI le dossier contient des fichiers
   if (files->Length != 0)
   {
    for each (String^ s in files)
     Console::WriteLine(tab->ToString() + Path::GetFileName(s));
   }
  }
```

lien : System.Environment
lien : System.IO.Directory
lien : System.IO.Path

Comment concaténer de manière intelligente un path et un nom de fichier ?

Auteurs : Jérôme Lambert - nico-pyright(c)

La méthode statique Combine de la classe Path permet de concaténer de manière intelligente un chemin de répertoire avec un nom de fichier :

```
String ^chemin = "C:\\dev";
String ^fichier = "fichier.txt";
String ^cheminComplet = System::IO::Path::Combine(chemin, fichier);
Console::WriteLine(cheminComplet);
```

Comment calculer la taille d'un répertoire ?

Auteurs : nico-pyright(c) - trueman

Nous utiliserons la classe DirectoryInfo avec les méthodes GetFiles pour récupérer les fichiers du répertoire et GetDirectories pour les sous répertoires, le tout de manière récursive :

```
using namespace System::IO;

static Int64 TailleRepertoire(DirectoryInfo ^rep)
{
 Int64 Size = 0;
 // liste tous les fichiers du répertoire rep
 array<FileInfo ^> ^ fichiers = rep->GetFiles();
 for each (FileInfo ^fich in fichiers)
  Size += fich->Length;
 // liste tous les sous-répertoires.
 array<DirectoryInfo^>^ sousrep = rep->GetDirectories();
 for each (DirectoryInfo ^repert in sousrep)

 Size += TailleRepertoire(repert); //appelle la méthode TailleRepertoire pour calculer la taille de chaque sous
 return (Size);
}

int main(array<System::String ^> ^args)
{
 String ^path = "E:\\temp";
 Int64 TailleOctet = TailleRepertoire(gcnew DirectoryInfo(path));
 Int64 TailleKiloOctet = TailleOctet / 1024;
 Console::WriteLine("Taille du répertoire {0} : {1} ko", path, TailleKiloOctet);
 return 0;
```

```
}
```

Comment récuperer le path de l'application ?

Auteur : nico-pyright(c)

On utilise :

```
String ^pathModule = System::IO::Path::GetDirectoryName(System::Reflection::Assembly::GetEntryAssembly()->Locati
```

Attention à ne pas confondre avec :

```
System::IO::Directory::GetCurrentDirectory()
```

qui donne le répertoire courant, qui n'est pas forcément le répertoire de l'exécutable.

Sommaire > Intéraction du C++/CLI avec le framework .Net > Fichiers, Répertoires, Disques > Compression

Comment compresser un fichier en utilisant GZip ?

Auteur : Webman

Tout d'abord pensez à ajouter ces deux namespaces sans quoi il n'est pas possible d'utiliser le code donné en exemple.

```
using namespace System::IO;
using namespace System::IO::Compression;
```

Avec cet exemple il vous suffit juste de passer en paramètres le chemin complet du fichier à compresser, et ensuite le chemin de destination du fichier GZip qui va être crée.

```
bool compresserFichier(String ^cheminSource, String ^cheminDestination)
{
    /* cheminSource : chemin complet du fichier à compresser
    cheminDestination : chemin complet du fichier compressé à créer*/
    try
    {
        // Le fichier est placé dans le FileStream
        FileStream ^ monFileStream = gcnew FileStream(cheminSource, FileMode::Open);
        cli::array<unsigned char,1> ^ monBuffer =
 gcnew cli::array<unsigned char>(safe_cast<int>(monFileStream->Length));
        // Lecture de l'intégralité du FileStream
        monFileStream->Read(monBuffer, 0, safe_cast<int>(monFileStream->Length));
        // Fermeture du FileStream
        monFileStream->Close();
        // Création du fichier qui va contenir le fichier compressé
        monFileStream = gcnew FileStream(cheminDestination, FileMode::Create);
        // Compression des données
        GZipStream ^monGZipStream =
 gcnew GZipStream(monFileStream, CompressionMode::Compress, false);
        // Ecriture des données compressées dans le fichier de destination
        monGZipStream->Write(monBuffer, 0, monBuffer->Length);
        // Fermeture du GZipStream
        monGZipStream->Close();
        return true;
    }
    catch(Exception ^e)
    {
        Console::WriteLine(e->Message);
        return false;
    }
}
```

lien : System.IO.Compression

Comment décompresser un fichier GZip ?

Auteur : Webman

Tout d'abord pensez à ajouter ces deux namespaces sans quoi il n'est pas possible d'utiliser le code donné en exemple.

```
using namespace System::IO;
using namespace System::IO::Compression;
```

Avec cet exemple il vous suffit juste de passer en paramètres le chemin complet du fichier GZip à décompresser, et ensuite le chemin de destination du fichier qui va être décompressé.

```
bool decompression(String ^cheminSource, String ^ cheminDestination)
{
    /*cheminSource : chemin complet du fichier compressé
    cheminDestination : chemin complet du fichier où le fichier doit être décompressé*/
    try
    {
        // Lecture du fichier compresse
        FileStream ^monFileStream = gcnew FileStream(cheminSource, FileMode::Open);
        // Données du fichier placées dans un GZipStream
        GZipStream ^monGzipStream = gcnew GZipStream(monFileStream, CompressionMode::Decompress);
        // Tableau qui va contenir la taille du fichier
        array <unsigned char> ^tailleOctets = gcnew array<unsigned char>(4);
        // Positionnement dans le Stream pour récupérer la taille
        int position = safe_cast<int>(monFileStream->Length) - 4;
        monFileStream->Position = position;
        // Récupération de la taille du fichier
        monFileStream->Read(tailleOctets, 0, 4);
        // Repositionnement en début du Stream
        monFileStream->Position = 0;
        // Conversion de la taille du fichier en entier
        int tailleFichier = BitConverter::ToInt32(tailleOctets, 0);
        // Dimesionnement du buffer
        array<unsigned char> ^buffer = gcnew array<unsigned char>(tailleFichier + 100);
        // Offset qui permettra de se repérer dans le Stream
        int monOffset = 0;

        while (true)
        {
            // Les données sont décompressées et placées dans le buffer
            int decompressionOctets = monGzipStream->Read(buffer, monOffset, 100);
            // Tant qu'il reste des données on continue
            if (!decompressionOctets)
                break;
            //  On icrémente l'offset pour ne pas repartir de 0 à chaque fois...
            monOffset += decompressionOctets;
         }

        // Création du fichier décompressé
        monFileStream = gcnew FileStream(cheminDestination, FileMode::Create);
        // Ecriture des données décomrpessées dans le fichier
        monFileStream->Write(buffer, 0, tailleFichier);
        // Efface les données en mémoire tampon
        monFileStream->Flush();
        // Fermeture des Streams
        monFileStream->Close();
        monGzipStream->Close();
        return true;
    }
    catch(Exception ^e)
    {
        Console::WriteLine(e->Message);
        return false;
    }
}
```

lien : System.IO.Compression

Comment lire un fichier Xml avec les classes de l'espace de noms System::Xml::Xpath ?

Auteurs : nico-pyright(c) - StormimOn

Voici un exemple basique de lecture d'un fichier Xml en se servant non pas de la classe XmlDocument mais des classes de l'espace de noms System::Xml::Xpath. Deux cas sont distingués, les fichiers Xml de base et ceux avec un espace de noms.

Fichier XML basique

```
<Recordbook>
  <Records>
    <Record>
      <FirstValue>10</FirstValue>
      <SecondValue>51</SecondValue>
    </Record>
    <Record>
      <FirstValue>25</FirstValue>
      <SecondValue>38</SecondValue>
    </Record>
  </Records>
</Recordbook>
```

son implémentation

```
using namespace System::Xml::XPath;

void TraiteXml(String ^ fileName)
{
 XPathDocument ^ doc = gcnew XPathDocument(fileName);
    XPathNavigator ^ nav = doc->CreateNavigator();
    // On récupère un XPathNodeIterator sur les Record
    XPathNodeIterator ^ iter = nav->Select("Recordbook/Records/Record");
    // Pour chaque Record
    while (iter->MoveNext())
    {
        // On récupère l'info FirstValue
        String ^ firstValue = iter->Current->SelectSingleNode("FirstValue")->Value;
        // On récupère l'info SecondValue
        String ^ secondValue = iter->Current->SelectSingleNode("SecondValue")->Value;
  Console::WriteLine("{0},{1}", firstValue, secondValue);
    }
}
```

Fichier Xml avec un espace de noms

```
<rd:Recordbook xmlns:rd="http://myexemple/myschema/record">
  <rd:Records>
    <rd:Record>
      <rd:FirstValue>10</rd:FirstValue>
      <rd:SecondValue>51</rd:SecondValue>
    </rd:Record>
    <rd:Record>
      <rd:FirstValue>25</rd:FirstValue>
      <rd:SecondValue>38</rd:SecondValue>
    </rd:Record>
  </rd:Records>
</rd:Recordbook>
```

son implémentation

```
using namespace System::Xml;

void TraiteXmlEspaceNom(String ^ fileName)
{
    XPathDocument ^ doc = gcnew XPathDocument(fileName);
    XPathNavigator ^ nav = doc->CreateNavigator();
    // On ajoute la gestion des espaces de noms
    XmlNamespaceManager ^ mgr = gcnew XmlNamespaceManager(nav->NameTable);
    mgr->AddNamespace("rd", "http://myexemple/myschema/record");
    // On récupère un XPathNodeIterator sur les Record
    XPathNodeIterator ^ iter = nav->Select("rd:Recordbook/rd:Records/rd:Record", mgr);
    // Pour chaque Record
    while (iter->MoveNext())
    {
        // On récupère l'info FirstValue
        String ^ firstValue = iter->Current->SelectSingleNode("FirstValue", mgr)->Value;
        // On récupère l'info SecondValue
        String ^ secondValue = iter->Current->SelectSingleNode("SecondValue", mgr)->Value;
  Console::WriteLine("{0},{1}", firstValue, secondValue);
    }
}
```

Remarque :
Attention, XPath est sensible à la casse. Le nom des balises doit correspondre exactement.

```
nav->Select("Recordbook/Records/Record");
```

et

```
nav->Select("Recordbook/Records/record");
```

ne représentent pas la même chose.

Comment créer un XmlNamespaceManager en se basant sur un fichier Xml ?

Auteurs : nico-pyright(c) - StormimOn

Précédemment, nous avons vu comment lire un fichier avec les classes XPath. Dans le cas de la présence de namespace dans le fichier Xml, nous avons utilisé la classe XmlNamespaceManager pour gérer ces espaces de noms. Le seul défaut c'est que nous alimentions manuellement ces données, nous allons donc maintenant créer ce XmlNamespaceManager de manière automatique.

Fichier Xml avec un espace de noms

```
<rd:Recordbook xmlns:rd="http://myexemple/myschema/record">
  <rd:Records>
    <rd:Record>
      <rd:FirstValue>10</rd:FirstValue>
      <rd:SecondValue>51</rd:SecondValue>
    </rd:Record>
    <rd:Record>
      <rd:FirstValue>25</rd:FirstValue>
      <rd:SecondValue>38</rd:SecondValue>
    </rd:Record>
  </rd:Records>
</rd:Recordbook>
```

Fichier Xml avec un espace de noms

```
using namespace System::Collections::Generic;
using namespace System::Xml;
using namespace System::Xml::XPath;

XmlNamespaceManager ^ GetXmlNamespaceManager(XPathNavigator ^ nav)
{
    XmlNamespaceManager ^ mgr = nullptr;
    nav->MoveToFirstChild();
    for each (KeyValuePair<String ^, String ^> ^ keyPair in
 nav->GetNamespacesInScope(XmlNamespaceScope::Local))
    {
        if (mgr == nullptr)
   mgr = gcnew XmlNamespaceManager(nav->NameTable);
        mgr->AddNamespace(keyPair->Key, keyPair->Value);
    }
    nav->MoveToRoot();
    return mgr;
}

void TraiteXml(String ^ fileName)
{
    XPathDocument ^ doc = gcnew XPathDocument(fileName);
    XPathNavigator ^ nav = doc->CreateNavigator();
    XmlNamespaceManager ^ mgr = GetXmlNamespaceManager(nav);
    if (mgr != nullptr)
    {
        XPathNodeIterator ^ iter = nav->Select("rd:Recordbook/rd:Records/rd:Record", mgr);
        while (iter->MoveNext())
  {
   String ^ firstValue = iter->Current->SelectSingleNode("rd:FirstValue", mgr)->Value;
            String ^ secondValue = iter->Current->SelectSingleNode("rd:SecondValue", mgr)->Value;
   Console::WriteLine("{0},{1}", firstValue, secondValue);
        }
    }
}
```

Comment valider un fichier XML avec un schéma XSD ?

Auteurs : nico-pyright(c) - StormimOn

Si vous devez valider un fichier XML avec un schéma XSD, voici la marche à suivre.

Fichier Xml exemple (books.xml)

```
<?xml version="1.0"?>
<catalog>
 <!--
 <title>2000-10-01</title>
 -->
 <book id="bk101">
  <author>Gambardella, Matthew</author>
  <title>XML Developer's Guide</title>
  <genre>Computer</genre>
  <prices>44.95</price>
  <publish_date>2000-10-01</publish_dates>
  <description>An in-depth look at creating applications with  XML.</description>
 </book>
</catalog>
```

Fichier XSD exemple (books.xsd)

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="catalog" type="CatalogData"/>
 <xsd:complexType name="CatalogData">
  <xsd:sequence>
   <xsd:element name="book" type="bookdata" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="bookdata">
  <xsd:sequence>
   <xsd:element name="author" type="xsd:string"/>
   <xsd:element name="title" type="xsd:string"/>
   <xsd:element name="genre" type="xsd:string"/>
   <xsd:element name="price" type="xsd:float"/>
   <xsd:element name="publish_date" type="xsd:date"/>
   <xsd:element name="description" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
 </xsd:complexType>
</xsd:schema>
```

```
using namespace System::Xml;
using namespace System::Xml::Schema;

public ref class ValidXml
{
private:
 static bool error;
 static void XmlValidationError(Object ^ sender, ValidationEventArgs ^ e)
 {
  // code à exécuter si erreur à la validation du Xml suivant le schéma Xsd
  // on passe autant de fois dans ce code qu'il y a d'erreurs
  Debug::WriteLine(e->Exception->Message);
  error = true;
 }
public:
 static Boolean isXmlValide(String ^xsdFile, String ^xmlFile)
 {
  error = false;
  try
  {
   XmlReaderSettings ^ settings = gcnew XmlReaderSettings();
   settings->Schemas->Add(nullptr, xsdFile);
   settings->ValidationType = ValidationType::Schema;
   settings->ValidationEventHandler += gcnew ValidationEventHandler(XmlValidationError);
   XmlReader ^ reader = XmlReader::Create(xmlFile, settings);
   while (reader->Read()) ;
  } catch (Exception ^)
  {
   return false;
  }
  return !error;
 }

};

void main () // test
{
 Console::WriteLine(ValidXml::isXmlValide("c:\\books.xsd", "c:\\books.xml"));

}
```

Comment sérialiser et désérialiser un objet simple en XML ?

Auteur : nico-pyright(c)

On utlise l'objet XmlSerializer pour effectuer une sérialisation et une désérialisation en xml.
Soit l'objet suivant :

```
public ref class Personne
{
private:
 String ^_login;
 String ^_pwd;
 int _age;
public:
 Personne(){}
 Personne(String ^login, String ^pwd, int age) : _login(login), _pwd(pwd), _age(age) {}

 property String ^Login
 {
  String ^ get() { return _login; }
  void set(String ^value) { _login = value; }
 }
 [System::Xml::Serialization::XmlIgnore()]
 property String ^Pwd
 {
  String^ get() { return _pwd; }
  void set(String ^value) { _pwd = value; }
 }
 property int Age
 {
  int get() { return _age; }
  void set(int value) { _age = value; }
 }

 virtual String^ ToString() override
 {
  return String::Format("Login : {0} - Mot de passe : {1} - Age : {2}", _login, _pwd, _age);
 }
};
```

Je peux choisir délibérément de ne pas sérialiser certaines propriétés. Par exemple, la propriété Pwd ne sera pas sérialisée grâce à l'attribut XmlIgnore.
NB : Pour être sérialisable, une classe doit-être publique, les membres à sérialiser doivent l'être également. La classe devra également posséder un constructeur par défaut et une propriété set pour chaque propriété get.

On effectue la sérialisation ainsi :

```
Personne ^moi = gcnew Personne("Nico", "Abcd", 28);
System::Xml::Serialization::XmlSerializer ^ sr =
gcnew System::Xml::Serialization::XmlSerializer(Personne::typeid);
System::IO::StreamWriter ^ writer = gcnew System::IO::StreamWriter("fichier.xml");
try
{
 sr->Serialize(writer, moi);
}
catch (Exception ^ e)
{
```

```
 Console::WriteLine("Impossible de sérialiser : " + e->Message);
}
finally
{
 writer->Close();
}
```

On utilise ici directement un StreamWriter pour enregistrer un fichier xml, mais il est envisageable d'utiliser d'autres surcharges, pour par exemple obtenir une chaine.

La désérialisation se déroule ainsi :

```
Personne ^moiClone;
System::IO::StreamReader ^reader = gcnew System::IO::StreamReader("fichier.xml");
try
{
 moiClone = (Personne^)sr->Deserialize(reader);
 Console::WriteLine(moiClone->ToString());
}
catch (Exception ^ e)
{
 Console::WriteLine("Impossible de désérialiser : " + e->Message);
}
finally
{
 reader->Close();
}
```

Notez bien sur que l'appel à ToString restitue une chaine vide pour le mot de passe car cette propriété n'a pas été sérialisée.

Comment sérialiser un objet en ignorant les références circulaires ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Prenons le cas de deux classes, classA et classB :

```
ref class classB;

[Serializable()]
public ref class classA
{
private:
 int m_MembreInt;
 classB ^m_B;
public:
 property int MembreInt;
 property classB ^B;
};

[Serializable()]
public ref class classB
{
private:
 float m_MembreFloat;
 classA ^m_A;
public:
 property float MembreFloat;
 property classA ^A;
};
```

Comme vous pouvez le voir, classA a une référence vers classB et classB a une référence vers classA. Si nous essayons de sérialiser un objet de type classA ou classB (et à condition que les références correspondantes ne soient pas nulles), nous aurons une erreur de référence circulaire et la sérialisation s'avèrera impossible :

```
classA ^objA = gcnew classA();
classB ^objB = gcnew classB();

objA->B = objB;
objB->A = objA;

// sérialisation
System::Xml::Serialization::XmlSerializer ^ sr =
 gcnew System::Xml::Serialization::XmlSerializer(classA::typeid);
System::IO::StreamWriter ^ writer = gcnew System::IO::StreamWriter("fichier.xml");
sr->Serialize(writer, objA);
writer->Close();
```

Le code ci dessous génèrera le message d'erreur suivant à l'exécution :

```
Référence circulaire détectée lors de la sérialisation d'un objet de type
 ConsoleApplication2.classA.
```

Afin de retirer les références circulaires, il suffira d'exclure les objets en question de la sérialisation en utilisant l'attribut System.Xml.Serialization.XmlIgnore sur les propriétés concernées :

```
[System::Xml::Serialization::XmlIgnore()]
property classB ^B;
```

et

```
[System::Xml::Serialization::XmlIgnore()]
property classA ^A;
```

Sommaire > Intéraction du C++/CLI avec le framework .Net > Thread, Processus

Comment créer et lancer un thread ?

Auteurs : abelman - nico-pyright(c)

Pour créer un thread, il faut utiliser la classe System::Threading::Thread.
Considérons que l'on dispose d'une Form.
Nous avons besoin de déclarer notre objet thread à l'intérieur de la Form.

```
using namespace System::Threading;
public ref class Threads : public System::Windows::Forms::Form
{
    // ........
    private : Thread ^_threadCalculs1;
}
```

La fonction exécutée par le thread a une signature imposée par le .NET Framework. Il s'agit du délégué System::Threading::ThreadStart.
C'est une fonction qui ne prend aucun paramètre et ne possède pas de valeur de retour.
On peut la déclarer ainsi dans notre Form.

```
private :void ThrFunc1()
{
    // Traitement effectué par le thread. Calculs est une fonction quelconque de notre Form
    try
    {
        Calculs(1000) ;
    }
    catch (Exception ^ex)
 {
    System::Diagnostics::Debug::WriteLine(ex->ToString());
    }
}
```

Pour démarrer le thread, on utilise la fonction Start de la classe Thread.

```
private : void StartThread()
{
    // ThrFunc1 est la fonction exécutée par le thread.
 _threadCalculs1 = gcnew Thread(gcnew ThreadStart(this, &Threads::ThrFunc1));
    // Il est parfois pratique de nommer les threads surtout si on en créé plusieurs.
    _threadCalculs1->Name = "Thread1";
    // Démarrage du thread.
    _threadCalculs1->Start();
}
```

Comment passer un ou plusieurs paramètres à un thread ?

Auteurs : abelman - nico-pyright(c) - Thomas Lebrun

Le délégué System::Threading::ThreadStart utilisé pour les fonctions de thread ne prend pas de paramètres.

Pour passer des paramètres à un thread, il vous faut créer une classe pour contenir les paramètres et la méthode du thread.

```
using namespace System;
using namespace System::Threading;

ref class ThreadParametre
{
private:
 String ^_text;
    int _entier;
public:
    // Constructeur
    ThreadParametre(String ^texte, int entier)
    {
        _text = texte;
        _entier = entier;
    }
    // Exécution de la méthode du thread
    void ExecuteThread()
    {
        for(int i = 0; i < _entier; i++)
        {
   Console::WriteLine("Index : " + i);
   Console::WriteLine("Message : " + _text);
        }
    }
};

int main(array<System::String ^> ^args)
{
 ThreadParametre ^ExempleThread = gcnew ThreadParametre("Message de test", 5);
 Thread ^t = gcnew Thread(gcnew ThreadStart(ExempleThread, &ThreadParametre::ExecuteThread));
 t->Start();

 return 0;
}
```

Comment arrêter un thread ?

Auteurs : abelman - nico-pyright(c)

Le meilleur moyen de d'arrêter un thread est de laisser sa fonction se terminer.
Si une fonction de thread s'exécute en continu dans une boucle, il est nécessaire d'écrire un code qui prévoit une condition pour sortir de la boucle. Cette condition doit pouvoir être modifiée par d'autres threads.
Reprenons l'exemple de notre Form (voir Q/R création d'un thread).
Pour signaler au thread que nous souhaitons qu'il s'arrête, nous allons utiliser un objet de la classe System::Threading::AutoResetEvent.
Dans la boucle de la fonction du thread, nous faisons attendre le thread pendant un court laps de temps. Si l'objet AutoResetEvent passe à l'état signalé, alors on sort du thread.

```
using namespace System;
using namespace System::Threading;

public ref class Threads : public System::Windows::Forms::Form
{
 private : Thread ^ _threadCalculs1;
 // Evènement de signal de fin de thread
 private : AutoResetEvent ^_endThreadCalculsEvent;
```

```
public:
 Threads(void)
 {
  InitializeComponent();
  _endThreadCalculsEvent = gcnew AutoResetEvent(false);
 }

 private :void ThrFunc1()
 {
  // Traitement effectué par le thread. Calculs est une fonction quelconque de notre Form
  try
  {
   Calculs(1000) ;
  }
  catch (Exception ^ex)
  {
   System::Diagnostics::Debug::WriteLine(ex->ToString());
  }
 }

 private : void StartThread()
 {
  // ThrFunc est la fonction exécutée par le thread.
  _threadCalculs1 = gcnew Thread(gcnew ThreadStart(this, &Threads::ThrFunc1));
  // Il est parfois pratique de nommer les threads surtout si on en créé plusieurs.
  _threadCalculs1->Name = "Thread1";
  // Démarrage du thread.
  _threadCalculs1->Start();
 }
 void Calculs(int tempo)
 {
  // Si l'événement est à l'état signalé, WaitOne renvoie true et la boucle se termine.
  while (! _endThreadCalculsEvent->WaitOne(tempo, false) )
  {
   // C'est ici ou notre thread fait son travail
   // .....
  }
 }

 // pour arreter le thread
 private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e)
 {
  // L'événement passe à l'état signalé
  _endThreadCalculsEvent->Set();
  // On attend la fin du thread.
  _threadCalculs1->Join();
 }
};
```

**Il existe un moyen plus radical d'arrêter un thread, c'est l'utilisation de la fonction Thread::Abort.
Lorsque vous appelez Abort, le Runtime lève une exception ThreadAbortException que le thread peut alors intercepter.
C'est aussi pourquoi il est déconseillé d'utiliser Abort car on ne peut prévoir où en est le thread dans son traitement. Lever une exception peut interrompre le thread alors qu'il est dans une partie du code qu'il doit terminer avant de sortir.
Un des exemples où on peut utiliser Abort sans risque : la fonction du thread est bloquée infiniment sur un appel (une attente de connexion socket par exemple).**

```
// Forcer la fin du thread
void AbortThread()
{
    _threadCalculs1->Abort();  // On demande au runtime d'arrêter le Thread
    _threadCalculs1->Join();   //  On attend la fin du thread.
```

```
}
```

Un thread terminé ne peut plus être relancé. Il faut instancier un nouvel objet Thread pour chaque démarrage de thread.

Comment changer le nom du thread courant ?

Auteur : sam_XIII

Tout d'abord pensez à la clause:

```
using namespace System::Threading;
```

Pour changer le nom du thread courant, ajoutez la ligne de code suivante :

```
Thread::CurrentThread->Name = "MainThread";
```

Ici on donne le nom "MainThread" au thread courant.
Si vous devez réaliser cela pour le thread principal, alors il faut ajouter cette ligne avant le "Application::Run(...)".

Comment ne lancer qu'une seule instance de mon application ?

Auteurs : abelman - nico-pyright(c)

Il arrive souvent de souhaiter interdire à une application d'avoir plusieurs instances lancées.
Voici une petite classe qui lors du démarrage de l'application, s'assure qu'elle n'est pas déjà en cours d'exécution.
Elle utilise un objet mutex nommé, donc potentiellement visible par tous les autres processus.

```
ref class SingleInstanceApp
{
private:
 System::Threading::Mutex ^_siMutex;
 bool _siMutexOwned;
public:
 SingleInstanceApp(String ^name)
 {
  _siMutex = gcnew System::Threading::Mutex(false, name);
  _siMutexOwned = false;
 }

 ~SingleInstanceApp()
 {
  // Libération du mutex si il a été acquis
  if (_siMutexOwned)
   _siMutex->ReleaseMutex();
 }

 // Application déjà lancée ?
 bool IsRunning()
```

```
 {
  // Acquisition du mutex.
  // Si _siMutexOwned vaut true, l'application acquiert le mutex car il est "libre"
  // Sinon le mutex a déjà été acquis lors du lancement d'une instance précédente
  _siMutexOwned =  _siMutex->WaitOne(0, true);
  return !(_siMutexOwned);
 }
};
```

Pour utiliser notre classe, il suffit de procéder ainsi dans le Main de notre application.

```
SingleInstanceApp ^app = gcnew SingleInstanceApp("{123456789 - ABCD - EFEG - XXXX}");
if (app->IsRunning())
 MessageBox::Show("Application déjà lancée");
else
 Application::Run(gcnew Form1());
delete app;
```

Important :
Si une application lambda en cours d'exécution crée un mutex ayant le même nom que celui de notre application, cette dernière ne pourra plus se lancer.
Elle se comportera comme si une autre instance de l'application était déjà en cours.
Il existe une technique pour l'éviter mais cela sort de notre sujet. Veuillez donc à choisir un nom assez compliqué pour votre mutex.

Remarque : On appelle explicitement le destructeur avec delete car on ne peut pas libérer le mutex dans le finalizer.

Comment lancer un processus ?

Auteurs : abelman - freegreg - nico-pyright(c)

Pour lancer un processus depuis notre application, on utilise la classe System::Diagnostics::Process.
Exemple : lancer une instance de internet explorer qui ouvre www.developpez.com

```
void StartProcess()
{
 // Instance de la classe Process
 System::Diagnostics::Process ^proc = gcnew System::Diagnostics::Process();
 // Nom de l'exécutable à lancer
 proc->StartInfo->FileName = "iexplore.exe";
 // Arguments à passer à l'exécutable à lancer
 proc->StartInfo->Arguments="http://www.developpez.com";
 // Démarrage du processus
 proc->Start() ;
 // On libère les ressources dont on n'a plus besoin.
 proc->Close(); // Attention Close ne met pas fin au processus.
```

```
}
```

Comment ouvrir un fichier avec l'application associée à son extension ?

Auteurs : abelman - nico-pyright(c)

On peut ouvrir des documents dont l'extension est connue du shell windows comme les .txt ou les .doc avec la classe System::Diagnostics::Process.
Exemple : Ouverture d'un fichier texte .txt.

```
//Instance de la classe System.Diagnostics.Process
System::Diagnostics::Process ^ proc = gcnew System::Diagnostics::Process();
//Nom du fichier dont l'extension est connue du shell à ouvrir
proc->StartInfo->FileName = "monfichier.txt";
//Démarrage du processus. Notepad (si il est associé aux fichiers .txt) sera alors lancé et ouvrira le fichier
proc->Start();
//On libère les ressources
proc->Close();
```

Comment rediriger la sortie standard d'un processus ?

Auteurs : abelman - nico-pyright(c)

Il est possible de rediriger la sortie standard d'un processus et de l'afficher dans un TextBox multiligne par exemple.

```
String ^ RedirectStdOutput(String ^nomProcess)
{
 System::Diagnostics::Process ^ proc = gcnew System::Diagnostics::Process();
 // On désactive le shell
 proc->StartInfo->UseShellExecute = false;
 // On redirige la sortie standard
 proc->StartInfo->RedirectStandardOutput = true;
 // On définit la commande
 proc->StartInfo->FileName = nomProcess;
 // Démarrage de la commande
 proc->Start();
 // Attente de la fin de la commande
 proc->WaitForExit();
 // Lecture de la sortie de la commande
 String ^ output = proc->StandardOutput->ReadToEnd();
 // Libération des ressources
 proc->Close();
 return output;
}
```

Comment lister les processus en cours d'exécution ?

Auteurs : abelman - nico-pyright(c)

Pour lister les processus en cours on utilise la fonction :

```
// Pour tous les processus en cours sur l'ordinateur local
array<System::Diagnostics::Process ^> ^ prc  = Process::GetProcesses();
// Pour tous les processus notepad en cours sur l'ordinateur local
array<System::Diagnostics::Process ^> ^ prc = Process::GetProcessesByName("notepad");
```

Comment arrêter un processus ?

Auteurs : abelman - nico-pyright(c)

Pour arrêter un processus, il faut disposer d'un objet System::Diagnostics::Process qui représente le processus.

```
// Pour les applications consoles
proc->Kill();
// Libération des ressources
proc->Close();
```

Pour les applications WinForm il est préférable d'utiliser CloseMainWindow afin que l'application reçoive le message de fermeture et se ferme correctement.

```
// Pour les applications avec une interface graphique (et donc une pompe de messages)
// Si l'appel échoue, on peut alors forcer la fermeture avec Kill.
proc->CloseMainWindow();
// Libération des ressources
proc->Close();
```

Comment vérifier la validité d'une adresse IP ?

Auteurs : abelman - nico-pyright(c) - Louis Guillaume Morand

Avec les expressions régulières.
On utilise le namespace System::Text::RegularExpressions.

```
bool CheckIpAddr(String ^ipAddress)
{
 String ^ re = "^(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)(\\.(25[0-5]|2[0-4]\\d|[0-1]?\\d?\\d)){3}$";
 return System::Text::RegularExpressions::Regex::IsMatch(ipAddress, re);
}
```

Version sans expression régulière.
On se sert de la fonction Split de la classe String pour parser la chaine, puis on analyse les différentes sous-chaînes.

```
bool CheckIpAddrNoRegex( String ^ipAddress )
{
 if ( ipAddress == nullptr || ipAddress == "" )
  return false;
 array<String^> ^ ipPartList = ipAddress->Split('.');
 if ( ipPartList->Length != 4 )
  return false;
 try
 {
  unsigned char ipPartNumber = Convert::ToByte(ipPartList[0]);
  ipPartNumber = Convert::ToByte(ipPartList[1]);
  ipPartNumber = Convert::ToByte(ipPartList[2]);
  ipPartNumber = Convert::ToByte(ipPartList[3]);
 }
 catch ( Exception ^) { return false; }
 return true;
}
```

Comment obtenir les adresses IP d'un ordinateur ?

Auteurs : abelman - freegreg - nico-pyright(c)

Pour obtenir les adresses IP d'un ordinateur on utilise la fonction Resolve de la Classe System::Net::Dns.
Voici une petite fonction qui le fait :

```
array<String ^> ^ GetIPaddresses(String ^computername)
{
 array<String ^> ^ saddr;
 array<IPAddress ^> ^ addr = Dns::GetHostEntry(computername)->AddressList;

 if (addr->Length > 0)
 {
  saddr = gcnew array<String^>(addr->Length);
  for (int i = 0; i < addr->Length; i++)
   saddr[i] = addr[i]->ToString();
 }
 return saddr;
}
```

Comment télécharger et afficher le contenu d'une page web ?

Auteurs : abelman - nico-pyright(c) - Louis Guillaume Morand

On utilise les classes HttpWebRequest, HttpWebResponse qui encapsulent la communication socket du protocole HTTP. On utilise ensuite la classe StreamReader pour lire le flux de la réponse.

```
HttpWebResponse ^HttpWResponse;
StreamReader ^sr;
try
{
 HttpWebRequest ^ HttpWRequest = safe_cast<HttpWebRequest ^>(WebRequest::Create("http://
www.developpez.com"));
 HttpWRequest->CachePolicy =
 gcnew Cache::HttpRequestCachePolicy(Cache::HttpRequestCacheLevel::Reload);
 HttpWResponse = safe_cast<HttpWebResponse^>(HttpWRequest->GetResponse());
 sr = gcnew StreamReader(HttpWResponse->GetResponseStream());
 richTextBox1->Text= sr->ReadToEnd();
}
catch (Exception ^ex)
{
 Console::WriteLine(ex->Message);
}
finally
{
 if (HttpWResponse != nullptr)
  HttpWResponse->Close();
 if (sr != nullptr)
  sr->Close();
}
```

Remarque : on utilise ici la propriété Reload du namespace Cache pour ne pas se servir du cache.

Comment pinguer une machine ?

Auteur : Jérôme Lambert

Avec la version 2 du Framework, la classe Ping est apparue permettant de faire la même chose que la commande ping utilisée sous Dos :

```
// Pinguer un machine
Ping ^monPing = gcnew Ping();
PingReply ^Reply = monPing->Send("192.168.0.1");
Console::WriteLine("Statut du ping : {0}", Reply->Status);
```

lien : Voir l'article de Webman
lien : System.Net.NetworkInformation.Ping

Sommaire > Intéraction du C++/CLI avec le framework .Net > ADO.NET

Qu'est-ce que ADO.NET ?

Auteur : Jérôme Lambert

Cette définition est tirée de MSDN
ADO.NET propose un accès cohérent à des sources de données telles que Microsoft SQL Server, ainsi qu'à des sources de données exposées via OLE DB et XML. Des applications grand public de partage de données peuvent utiliser ADO.NET pour se connecter à des sources de données et extraire, manipuler et mettre à jour des données.

lien : Vue d'ensemble d'ADO.NET

Quelles classes utiliser pour me connecter à ma base de donnée ?

Auteur : abelman

Le .NET Framework dispose de plusieurs namespaces permettant de se connecter à divers SGBD.

- **System::Data::SqlClient pour SQL Server**
- **System::Data::Odbc pour les SGBD fournissant un pilote ODBC**
- **Oracle::DataAcess (de ORACLE) pour ORACLE. Disponible en installant Oracle Data Provider .NET (ODP .NET)**
- **System::Data::Oracle::Client (de Microsoft) pour Oracle**
- **System::Data::OleDb pour tous les SGBD ayant un fournisseur OLE DB**

Comment se connecter à une base de données ?

Auteurs : abelman - nico-pyright(c)

Voici un exemple pour SQL Server :

```
using namespace System::Data::SqlClient;
try
{
    // Chaîne de connexion
    String ^connectString = "database=test_paresco;server=am01;User ID=BACK;pwd=xxxxxx";
    // Objet connection
    SqlConnection ^connection = gcnew SqlConnection(connectString);
    // Ouverture
    connection->Open();
    // Fermeture
    connection->Close();
}
catch (Exception ^ex)
{
    System::Diagnostics::Debug::WriteLine(ex->ToString());
}
```

Comment exécuter une requête SELECT ?

Auteurs : abelman - nico-pyright(c)

On utilise les objets Command et DataReader.

Exemple avec SQL Server :

```
// Chaîne de connexion
String ^connectString = "database=test_paresco;server=am01;User ID=BACK;pwd=xxxxxx";
// Objet connection
SqlConnection ^connection = gcnew SqlConnection(connectString);
// Ouverture
connection->Open();
// Objet Command
SqlCommand ^command = gcnew SqlCommand("SELECT * FROM usr_contract", connection);
// Objet DataReader
SqlDataReader ^reader = command->ExecuteReader();
array<Object ^> ^row = gcnew array<Object ^>(reader->FieldCount);
while (reader->Read())
{
 reader->GetValues(row);
 for (int i=0; i<row->GetLength(0); i++)
 {
  if (row[i] != DBNull::Value)
   Console::Write(row[i]);
  else
   Console::Write("NULL");
  if (i<row->GetUpperBound(0))
   Console::Write("|");
 }
 Console::WriteLine();
}
// Fermeture reader
reader->Close();
// Fermeture connection
connection->Close();
```

Comment exécuter une requête non SELECT ?

Auteurs : abelman - nico-pyright(c)

On utilise la méthode ExecuteNonQuery de l'objet Command.
Exemple avec SqlServer :

```
// Chaîne de connexion
String ^connectString = "database=test_paresco;server=am01;User ID=BACK;pwd=xxxxxx";
// Objet connection
SqlConnection ^connection = gcnew SqlConnection(connectString);
// Ouverture
connection->Open();
// Objet Command
SqlCommand ^command = gcnew SqlCommand("UPDATE usr_contract set ctr_n = ctr_n + 1", connection);
// Execution
int affectedrows = command->ExecuteNonQuery();
Console::WriteLine("Nombre de lignes affectées {0}", affectedrows);
// Fermeture connection
connection->Close();
```

Comment exécuter une requête paramétrée ?

Auteurs : abelman - nico-pyright(c)

Il est possible de passer des paramètres à des requêtes SQL.
Exemple avec SQL Server :

```
using namespace System::Data::SqlClient;
using namespace System::Data;

// Chaîne de connexion
String ^connectString = "database=test_paresco;server=am01;User ID=BACK;pwd=xxxxxx";
// Objet connection
SqlConnection ^connection = gcnew SqlConnection(connectString);
// Ouverture
connection->Open();
// Objet Command
SqlCommand ^command = gcnew SqlCommand("SELECT * FROM usr_contract WHERE " +
    "ctr_ref = @contract AND ctr_exg_ref = @exg",
    connection);
// Paramètres
command->Parameters->Add(gcnew SqlParameter("@contract", SqlDbType::VarChar, 5));
command->Parameters["@contract"]->Value = "FTE";
command->Parameters->Add(gcnew SqlParameter("@exg", SqlDbType::VarChar, 8));
command->Parameters["@exg"]->Value = "SBF";
// Object datareader
SqlDataReader ^reader = command->ExecuteReader();
array<Object ^> ^row = gcnew array<Object ^>(reader->FieldCount);
while (reader->Read())
{
 reader->GetValues(row);
 for (int i=0; i<row->GetLength(0); i++)
 {
  if (row[i] != DBNull::Value)
   Console::Write(row[i]);
  else
   Console::Write("NULL");
  if (i<row->GetUpperBound(0))
   Console::Write("|");
 }
 Console::WriteLine();
}
// Fermeture reader
reader->Close();
// Fermeture base
connection->Close();
```

Comment exécuter une requête dont le texte comprend une quote simple ?

Auteurs : abelman - nico-pyright(c)

Pour exécuter une requête contenant une quote simple, il faut utiliser les requêtes paramétrées.
On peut aussi faire plus simple en doublant les quotes avant d'exécuter la requête.

```
String ^sql = "SELECT * FROM pasta WHERE name = 'aujourd''hui'";
// Ou
```

```
String ^name = "aujourd'hui";
sql = "SELECT * FROM pasta WHERE name = " + name->Replace("'", "''");
```

Comment puis-je exécuter une procédure stockée ?

Auteurs : nico-pyright(c) - Thomas Lebrun

Pour pouvoir exécuter une procédure stockée, vous devez utiliser un objet SqlCommand et indiquer à sa propriété CommandType que vous désirez utiliser une procédure stockée :

```
// Objet SqlCommand
SqlCommand ^ cmd = gcnew SqlCommand();
// On indique que l'on souhaite utiliser une procédure stockée
cmd->CommandType = CommandType::StoredProcedure;
// On donne le nom de cette procédure stockée
cmd->CommandText = "CustOrderHist";
```

Comment écrire le contenu de ma table dans un fichier XML ?

Auteurs : nico-pyright(c) - Thomas Lebrun

En utilisant un DataSet et sa méthode WriteXml, vous avez la possibilité d'écrire le contenu d'une table dans un fichier XML.

```
// Création de la chaîne de connexion
String ^_ConnectionString =  "Server=Srv1;Database=Northwind;User ID=sa;Password=asdasd";
// Création de la connexion
SqlConnection ^_SqlConnection = gcnew SqlConnection();
_SqlConnection->ConnectionString = _ConnectionString;
// Création du SqlDataAdapter
SqlDataAdapter ^ da = gcnew SqlDataAdapter("Select * from Customers", _SqlConnection);
// Création d'un DataSet
DataSet ^ds = gcnew DataSet();
// Remplissage du DataSet avec le SqlDataAdapter
da->Fill(ds, "Customers");
// Ecriture du fichier XML au moyen de la méthode WriteXml
ds->WriteXml("C:\\TestXml.xml");
```

Voici, après traitement, le contenu du fichier TestXml.xml :

```
 <?xml version="1.0" standalone="yes" ?>
- <NewDataSet>
- <myTable>
<CustomerID>ALFKI</CustomerID>
 <CompanyName>Alfreds Futterkiste</CompanyName>
 <ContactName>Maria Anders</ContactName>
  <ContactTitle>Sales Representative</ContactTitle>
  <Address>Obere Str. 57</Address>
  <City>Berlin</City>
  <PostalCode>12209</PostalCode>
  <Country>Germany</Country>
  <Phone>030-0074321</Phone>
  <Fax>030-0076545</Fax>
</myTable>
```

Comment créer une instance d'un type dynamiquement ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Pour instancier une classe à partir d'un type, il vous faudra utiliser la classe Activator :

```
// Récupération du type à  instancier dynamiquement
Type ^objType = Test::typeid;
// Instance dynamique à  partir du type donné
Object ^objInstanceDynamique = Activator::CreateInstance(objType);
// Casting de l'objet
Test ^objTest = (Test^)objInstanceDynamique;
```

lien : System.Activator

Comment créer une instance d'un type dynamiquement à partir d'une chaîne de caractères ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Pour instancier un classe à partir d'une chaîne de caractères, il vous faudra dans un premier temps transformer cette châine en un objet de type Type. A partir de l'objet Type obtenu, vous pourrez créer une instance grâce à la classe Activator :

```
// Récupération du type à instancier dynamiquement à partir d'une chaîne de caractères
Type ^objType = Type::GetType("Test");
// Vérification si le type a bien été retrouvé
if (objType != nullptr)
{
 // Instance dynamique à partir du type donné
 Object ^objInstanceDynamique = Activator::CreateInstance(objType);
 // Casting de l'objet
 Test ^objTest = (Test ^)objInstanceDynamique;
}
else
{
 Console::WriteLine("Le type semble être incorrect!");
}
```

lien : System.Activator
lien : System.Type

Comment mon application .NET peut interagir avec les applications et les documents de la suite Microsoft Office ?

Auteur : Skalp

Certains éditeurs fournissent des interfaces permettant de communiquer avec leurs applications par programmation. Cette capacité s'appelle l'interopérabilité. La technologie s'appelle l'automation.
Microsoft fournit des interfaces permettant de piloter les applications Office depuis une application .NET (Qu'est-ce que l'automation Office ?) . Elles sont appelées Primary Interop Assemblies (Que sont les Primary Interop Assemblies (PIA) ?).
Cette citation, tirée de l'Aide et Support Microsoft, illustre bien les possibilités offertes par l'automation Office :

```
« Avec Automation, vous pouvez utiliser la fonctionnalité de publipostage de Microsoft Word pour
 générer des lettres types à partir de données
d'une base de données sans que l'utilisateur se rende compte que Word est impliqué.
Vous pouvez même utiliser toutes les fonctionnalités de graphiques et d'analyse de données de
 Microsoft Excel à l'aide d'Automation.
Vous n'avez pas besoin d'écrire votre propre moteur de calcul pour fournir la multitude de fonctions
 mathématiques, financières et techniques fournies par Excel.
Il vous suffit d'automatiser Microsoft Excel pour « emprunter » ces fonctionnalités et les
 incorporer dans votre propre application. »
```

Un autre moyen de communiquer avec un document Office, est de le considérer comme une base de données. C'est valable évidemment pour Access, mais aussi, et c'est moins connu, pour Excel.
Une troisième méthode est de manipuler directement le format natif des fichiers Office. C'est le cas du projet SourceForge Koogra qui permet de lire un fichier Excel à partir de son format natif BIFF (Binary Interchange File Format).
Depuis la version 2007 d'Office, un nouveau format a vu le jour, c'est le format OpenXML qui n'est autre qu'un fichier XML et donc aisément lisible et modifiable par programmation. Cependant, ce format n'est compatible que pour Office 2007 et éventuellement Office 2003, après avoir installé un patch permettant de lire ce format.
Enfin, sachez qu'il existe des librairies payantes de pilotage des applications Office. Nul besoin d'en dire plus, vous pourrez trouver toutes les informations nécessaires sur votre moteur de recherche préféré, avec des mots-clés tels que : read, write, excel, word, powerpoint, files, .net,…
Il est aussi possible de développer un Add-in pour Office ou de développer une application à partir d'un document Office. C'est ce que permettent les Visual Studio Tools pour Office (Que sont les Visual Studio Tools for Office (VSTO) ?).

Qu'est-ce que l'automation Office ?

Auteur : Skalp

Cette définition de l'automation est tirée du Support Microsoft :

```
« Automation (anciennement OLE Automation) est une technologie qui vous permet de tirer parti d'une
 fonctionnalité ou du contenu d'un
programme existant et de l'incorporer à vos propres applications. Automation est basé sur le
 composant COM (Component Object Model).
COM est une architecture logicielle standard basée sur des interfaces, et conçue pour séparer le
 code en objets autonomes ou composants.
Chaque composant expose un jeu d'interfaces par lesquelles toute la communication avec le composant
 est gérée.
Automation est constitué d'un serveur et d'un client. Le serveur Automation est l'application qui
 expose
```

```
ses fonctionnalités au travers d'interfaces COM à d'autres applications, appelées clients
 Automation. »
```

L'automation Office est la technologie automation appliquée au modèle objet Office.

lien : L'automation Office 97 et Office 2000
lien : Automation
lien : COM (Component Object Model)

Que sont les Primary Interop Assemblies (PIA) ?

Auteur : Skalp

Les Primary Interop Assemblies (PIAs) sont des assemblys de code managé capables de communiquer avec les assemblys (de code non managé) du modèle objet Office. Cela est rendu possible parce que les assemblys du modèle objet Office exposent des interfaces COM.
Le mot "Primary" est utilisé lorsque ces assemblys sont fournis par l'éditeur. En l'occurrence, pour les assemblys Office, l'éditeur est Microsoft.
Attention, chaque version d'Office a ses propres versions de PIAs. Mais chaque version de PIAs est compatible avec les versions précédentes.
Il est possible de générer soi-même ces assemblys grâce à l'outil .NET TLBIMP. Ils s'appellent dans ce cas Interop assemblies (IAs). Mais il faut savoir que les PIAs sont optimisés par Microsoft. De plus, les types définis dans les IAs ne seront pas les mêmes que ceux définis dans les PIAs.
Exemple : le type Document d'un PIA est différent du type Document d'un IA ; qui lui-même est différent du type Document d'un autre IA.

lien : Assemblys PIA (Primary Interop Assemblies)
lien : Primary Interop Assemblies (PIAs)

Comment installer les Primary Interop Assemblies ?

Auteur : Skalp

Les PIAs sont disponibles dans les installations des applications Office version 2003 et supérieures. Ces modules sont intitulés *Prise en charge de la programmabilité .NET* (*.NET Programmability support*).
Certains PIAs sont aussi disponibles au téléchargement :
- Office XP
- Office 2003
Les PIAs s'installent dans le Global Assembly Cache (GAC) du poste.
Pour les versions 97 et 2000, les IAs sont générés par TLBIMP. Cette génération est automatisée dans Visual Studio lorsque les références sont ajoutées dans un projet.

lien : Comment : installer les assemblys PIA (Primary Interop Assembly) d'Office
lien : Office 2003 Update: Redistributable Primary Interop Assemblies
lien : Office XP Primary Interop Assemblies (PIAs)

Que sont les Visual Studio Tools for Office (VSTO) ?

Auteur : Skalp

Les outils Visual Studio pour Office permettent d'utiliser la technologie .NET à partir d'un document Office. Autrement dit, presque tout ce qu'il est possible de faire en winform ou webform est possible au sein du document Office.
Les outils Visual Studio pour Office ne sont valables qu'avec Office 2003 et 2007.

Attention : Les modèles de projet VSTO ne sont pas disponibles dans les versions Express de Visual Studio.

lien : Visual Studio 2005 Tools for Microsoft Office: L'automation Office en .NET
lien : Library : Visual Studio Tools pour Office
lien : Office Developper Center : Visual Studio Tools for Office

Comment installer les Visual Studio Tools for Office ?

Auteur : Skalp

Pour pouvoir développer et exécuter des solutions VSTO, il faut installer le Runtime VSTO (VSTOR), dans sa version 2.0 pour Office 2003 et 3.0 pour Office 2007. Celui-ci est disponible sur le centre de téléchargement Microsoft avec le mot-clé : VSTOR.
Les autres composants indispensables sont :
- Le framework .NET
- Office 2003 ou 2007, dont :
*** La programmabilité .NET (PIAs)**
*** Visual Basic pour Applications**

lien : Installation de Visual Studio Tools pour Office
lien : Centre de téléchargement Microsoft

Où puis-je trouver de la documentation sur l'automation Office ?

Auteur : Skalp

1. Les ressources developpez.com
Les articles developpez.com sont un bon point de départ pour l'automation Office : les meilleurs cours et tutoriels C#.NET avec OFFICE.
2. La MSDN.
La MSDN regorge d'informations concernant l'automation Office et les modèles objet Office. Voici quelques points d'entrée intéressants :
Visual Studio Tools pour Office
Understanding the Excel Object Model from a Visual Studio 2005 Developer's Perspective
Understanding the Word Object Model from a Visual Studio 2005 Developer's Perspective
3. Les macros VBA.
Enregistrer une macro Office puis visualiser le code VBA généré permet de découvrir les objets Office utilisés et leurs paramètres. C'est très pratique lorsqu'on connaît la fonctionnalité à programmer mais qu'on ne connaît pas l'objet Office correspondant.
4. La documentation du modèle objet.
Office fournit une aide décrivant le modèle objet des applications avec des exemples d'utilisation en VBA. Cette aide se trouve dans le répertoire d'installation Office. Pour retrouver ces documents, voir : How to find and use Office object model documentation sur le centre d'aide et support.

lien : Les meilleurs cours et tutoriels C#.NET avec OFFICE
lien : Visual Studio Tools pour Office
lien : Understanding the Excel Object Model from a Visual Studio 2005 Developer's Perspective

lien : How to find and use Office object model documentation

Comment l'automation dans mon application peut-elle être compatible avec plusieurs versions d'Office ?

Auteur : Skalp

Chaque version d'Office a ses propres PIAs. Mais, de la même façon qu'Office, chaque version des PIAs est compatible avec les versions précédentes.
Ainsi, pour être sûr d'être compatible avec un maximum de versions Office, le mieux est d'utiliser les PIAs les plus anciens. Tout ce qui fonctionne avec une version ancienne, fonctionnera avec une version plus récente d'Office.
D'un autre côté, une version ancienne ne permet pas de bénéficier des fonctions apparues dans les versions récentes d'Office.
En général, les assemblys de la version 9 d'Office (Office 2000) suffisent amplement.
Il faut savoir que le format des documents Office a été profondément modifié entre la version 8 (Office 97) et 9 (Office 2000). Par conséquent, l'automation Office avec des PIAs d'Office 97 pour des versions supérieures est déconseillée.
Tableau récapitulatif des versions d'Office et de leur appellation commerciale :

Version	Appellation commerciale
8.0	Office 97
9.0	Office 2000
10.0	Office XP ou Office 2002
11.0	Office 2003
12.0	Office 2007

Comment libérer les ressources d'une automation Office ?

Auteurs : Skalp - Thomas Lebrun

Nombreux sont ceux qui rencontrent ce problème : lorsque vous quittez une automation, vous utilisez (à quelques détails près) cette portion de code :

```
monAppli->Quit();
monAppli = nullptr;
```

Le plus souvent, ceci ne suffit pas à fermer le processus office : vous pouvez toujours le voir dans le gestionnaire des tâches. Ainsi, lorsque vous ouvrez le document, avec lequel vous avez travaillé par automation, depuis Windows, il refuse de l'ouvrir au titre qu'il est en cours d'utilisation par… vous-même !
Voilà donc quelques astuces pour libérer les ressources automation efficacement :
1. Déclarer les objets avec des variables indépendantes :
Par exemple, ne faites pas :

```
Workbook ^monClasseur = monAppli->Workbooks->Add();
```

Mais plutôt :

```
Workbooks ^mesClasseurs = monappli->Workbooks;
Workbook ^monClasseur = mesClasseurs->Add();
```

Pour une automation simple, cela peut passer ; mais pour une automation complexe, cela peut s'avérer extrêmement fastidieux !
2. Libérer les références aux objets COM :
Utilisez la méthode suivante pour libérer les références à vos objets COM :

```
System::Runtime::InteropServices::Marshal::ReleaseComObject(monAppli);
```

3. Utiliser le Garbage Collector (GC) :

```
GC::Collect();
GC::WaitForPendingFinalizers();
```

Il peut être utile de doubler cette portion, dans le cas où la référence mémoire du process est conservée après le premier appel.
4. Récupérer l'identifiant du processus office :
La méthode la plus définitive est de tuer le processus à la fin de l'automation.
Il faut pour cela, dans un premier temps, récupérer et stocker l'identifiant du processus pendant tout le temps que doit durer l'automation. Exemple pour une automation Word :

```
int processId = 0;
// 1. Additionner les identifiants des processus winword avant l'ouverture de l'automation :
for each (Process ^item in Process::GetProcessesByName("winword"))
 processId -= item.Id;

// 2. Ouvrir l'automation :
ApplicationClass ^monAppli = gcnew ApplicationClass();

// 3. Ajouter les identifiants des processus winword après l'ouverture de l'automation,
// la différence (après - avant) donnera l'identifiant du processus ouvert :
foreach (Process ^item in Process::GetProcessesByName("winword"))
 processId += item.Id;

// Automation...

// Enfin, lorsque l'automation est terminée, il suffit de tuer le processus à partir de son identifiant :
Process::GetProcessById(processId)->Kill();
```

lien : Office application does not quit after automation from Visual Studio .NET client

Peut-on utiliser le Compact Framework avec le C++/CLI ?

Auteurs : Jérôme Lambert - nico-pyright(c)

Cette définition est tirée de MSDN
Le .NET Compact Framework est un environnement indépendant du matériel permettant d'exécuter des programmes sur divers périphériques informatiques à ressources limitées : assistants numériques personnels (PDA, Personal Data Assistant) tels que le Pocket PC, téléphones mobiles, décodeurs, périphériques informatiques automobiles et périphériques personnalisés intégrés au système d'exploitation Windows CE .NET.
Le .NET Compact Framework est un sous-ensemble de la bibliothèque de classes du .NET Framework, mais contient également des classes spécialement conçues à son intention. Il hérite la totalité de l'architecture de Common Language Runtime et d'exécution de code managé du .NET Framework.

Malheureusement, à l'heure actuelle, il n'est pas possible d'utiliser le C++/CLI pour utiliser le Compact Framework. Les mécanismes d'intéropérabilités (It Just Works) n'étant pas disponibles dans le CF, il a été jugé inutile de permettre l'utilisation du C++/CLI, mettant en avant plutôt l'utilisation de C#.
Mais il n'est pas totalement exclu que les équipes VC++ et CF se mettent à travailler ensemble dans l'avenir, même si cela parait peu probable.

lien : It Just Works

Comment calculer la différence de temps entre deux dates ?

Auteur : Jérôme Lambert

Grâce à la redéfinition de l'opérateur - (mais aussi de +, ==, !=, >, >=, <, <=) avec la classe TimeSpan, il est possible de faire la différence entre 2 objets DateTime
Calculons le nombre de jours écoulés depuis la création de cette question :

```
DateTime DateCourante = DateTime::Now;
DateTime DateCreationquestion = DateTime(2007, 1, 3);

TimeSpan ^Ts = DateCourante - DateCreationquestion;
Console::WriteLine("Il s'est écoulé {0} jour(s) depuis la création de cette question !", Ts->Days);
```

Comment mesurer un intervalle de temps avec précision ?

Auteur : Jérôme Lambert

Avec la version 2 du Framework, il est apparu la classe StopWatch qui permet de mesurer un intervalle de temps avec grande précision :

```
Stopwatch ^maMesureDeTemps = gcnew Stopwatch();
// Démarrage de l'intervalle de temps
maMesureDeTemps->Start();
// ...
// Fin de l'intervalle de temps
maMesureDeTemps->Stop();
```

```
Console::WriteLine("L'exécution du code a pris : {0}", maMesureDeTemps->Elapsed.ToString());
```

lien : Voir l'article de Webman
lien : System.Diagnostics.StopWatch
lien : Voir aussi avec l'API Win32

Comment modifier le texte de la barre de titre (fenêtre et console)?

Auteur : Jérôme Lambert

Pour une application Windows, c'est la propriété Text de la la classe Form qui contient le texte de la barre de titre :

```
this->Text = "Mon titre";
```

Pour ce qui est d'une application console, il faut passer par la classe Console avec la propriété Title :

```
Console::Title = "Mon titre";
```

Comment utiliser le cryptage avec l'algorithme Rijndael ?

Auteur : nico-pyright(c)

**Le framework.net fourni tout une série de classe pour faciliter le cryptage / décryptage de données.
Voici un exemple de génération de clé à partir d'une chaîne, ainsi que le cryptage et décryptage de fichiers.
Notez les méthodes pour crypter/décrypter des chaînes, ainsi que l'utilisation de BinaryReader et BinaryWritter pour crypter/décrypter un fichier.**

```
using namespace System;
using namespace System::Text;
using namespace System::Security::Cryptography;
using namespace System::IO;


void GenerateKey(String ^SecretPhrase, array<unsigned char> ^&Key, array<unsigned char> ^&IV)
{
 array<unsigned char> ^bytePhrase = Encoding::ASCII->GetBytes(SecretPhrase);
 SHA384Managed ^sha384 = gcnew SHA384Managed();
 sha384->ComputeHash(bytePhrase);
 array<unsigned char> ^result = sha384->Hash;
 for (int loop = 0; loop < 24; loop++)
  Key[loop] = result[loop];
 for (int loop = 24; loop < 40; loop++)
  IV[loop - 24] = result[loop];
}

array<unsigned char> ^ Crypter(array<unsigned char> ^encrypted, String ^keyPhrase)
{
    array<unsigned char> ^Key = gcnew array<unsigned char>(24);
    array<unsigned char> ^IV = gcnew array<unsigned char>(16);

    GenerateKey(keyPhrase, Key, IV);

    ASCIIEncoding ^textConverter = gcnew ASCIIEncoding();
    RijndaelManaged ^myRijndael = gcnew RijndaelManaged();
```

```
    myRijndael->Key = Key;
    myRijndael->IV = IV;

    ICryptoTransform ^encryptor = myRijndael->CreateEncryptor(Key, IV);
    MemoryStream ^msEncrypt = gcnew MemoryStream();
 CryptoStream ^csEncrypt = gcnew CryptoStream(msEncrypt, encryptor, CryptoStreamMode::Write);

    csEncrypt->Write(encrypted, 0, encrypted->Length);
    csEncrypt->FlushFinalBlock();

    encrypted = msEncrypt->ToArray();
 return encrypted;
}

array<unsigned char> ^ Decrypter(array<unsigned char> ^encrypted, String ^keyPhrase)
{
    array<unsigned char> ^Key = gcnew array<unsigned char>(24);
    array<unsigned char> ^IV = gcnew array<unsigned char>(16);
    GenerateKey(keyPhrase, Key, IV);

    array<unsigned char> ^fromEncrypt;
    RijndaelManaged ^myRijndael = gcnew RijndaelManaged();
    ASCIIEncoding ^textConverter = gcnew ASCIIEncoding();

    myRijndael->Key = Key;
    myRijndael->IV = IV;

    ICryptoTransform ^decryptor = myRijndael->CreateDecryptor(Key, IV);
    MemoryStream ^msDecrypt = gcnew MemoryStream(encrypted);
 CryptoStream ^csDecrypt = gcnew CryptoStream(msDecrypt, decryptor, CryptoStreamMode::Read);
    fromEncrypt = gcnew array<unsigned char>(encrypted->Length);

    csDecrypt->Read(fromEncrypt, 0, fromEncrypt->Length);

 return fromEncrypt;
}

String ^ Crypter(String ^original, String ^keyPhrase)
{
    array<unsigned char> ^Key = gcnew array<unsigned char>(24);
    array<unsigned char> ^IV = gcnew array<unsigned char>(16);

    GenerateKey(keyPhrase, Key, IV);

    ASCIIEncoding ^textConverter = gcnew ASCIIEncoding();
    RijndaelManaged ^myRijndael = gcnew RijndaelManaged();
    array<unsigned char> ^encrypted;
    array<unsigned char> ^toEncrypt;

    myRijndael->Key = Key;
    myRijndael->IV = IV;

    ICryptoTransform ^encryptor = myRijndael->CreateEncryptor(Key, IV);
    MemoryStream ^msEncrypt = gcnew MemoryStream();
 CryptoStream ^csEncrypt = gcnew CryptoStream(msEncrypt, encryptor, CryptoStreamMode::Write);

    toEncrypt = textConverter->GetBytes(original);

    csEncrypt->Write(toEncrypt, 0, toEncrypt->Length);
    csEncrypt->FlushFinalBlock();

    encrypted = msEncrypt->ToArray();
 return Convert::ToBase64String(encrypted);
}

String ^ Decrypter(String ^ encryptedString, String ^keyPhrase)
{
```

```
    array<unsigned char> ^Key = gcnew array<unsigned char>(24);
    array<unsigned char> ^IV = gcnew array<unsigned char>(16);
    GenerateKey(keyPhrase, Key, IV);

 array<unsigned char> ^encrypted = Convert::FromBase64String(encryptedString);
    array<unsigned char> ^fromEncrypt;
    RijndaelManaged ^myRijndael = gcnew RijndaelManaged();
    ASCIIEncoding ^textConverter = gcnew ASCIIEncoding();

    myRijndael->Key = Key;
    myRijndael->IV = IV;

    ICryptoTransform ^decryptor = myRijndael->CreateDecryptor(Key, IV);
    MemoryStream ^msDecrypt = gcnew MemoryStream(encrypted);
 CryptoStream ^csDecrypt = gcnew CryptoStream(msDecrypt, decryptor, CryptoStreamMode::Read);
    fromEncrypt = gcnew array<unsigned char>(encrypted->Length);

    csDecrypt->Read(fromEncrypt, 0, fromEncrypt->Length);

    return textConverter->GetString(fromEncrypt);
}

int main(array<System::String ^> ^args)
{
 FileStream ^fs = gcnew FileStream("c:\\test.zip", FileMode::Open);
 BinaryReader ^br = gcnew BinaryReader(fs);
 FileStream ^fsw = gcnew FileStream("c:\\test.cry", FileMode::CreateNew);
 BinaryWriter ^bw = gcnew BinaryWriter(fsw);
 try
 {
  bw->Write(Crypter(br->ReadBytes((int)fs->Length), "code secret"));
 }
 catch (Exception^)
 {
 }
 finally
 {
  br->Close();
  fs->Close();
  bw->Close();
  fsw->Close();
 }
 fs = gcnew FileStream("c:\\test.cry", FileMode::Open);
 br = gcnew BinaryReader(fs);
 fsw = gcnew FileStream("c:\\test2.zip", FileMode::CreateNew);
 bw = gcnew BinaryWriter(fsw);
 try
 {
  bw->Write(Decrypter(br->ReadBytes((int)fs->Length), "code secret"));
 }
 catch (Exception^)
 {
 }
 finally
 {
  br->Close();
  fs->Close();
  bw->Close();
  fsw->Close();
 }
    return 0;
```

```
}
```

Comment travailler avec les fichiers de configuration ?

Auteur : nico-pyright(c)

Les fichiers de configuration sont des fichiers XML qui contiennent la configuration de notre exécutable. Ils doivent se situer dans le même répertoire que l'exécutable.
Visual C++ ne gère pas automatiquement les fichiers de configuration comme app.config. Tant est si bien que quand on essaie de les utiliser, à chaque chargement de valeur, on obtient une chaine vide.
Ceci est expliqué par le fait que Visual C++ ne copie pas automatiquement le fichier app.config dans le répertoire de l'exécutable (debug par exemple).
Il faut donc le faire manuellement ou bien se servir des événements après génération.
Aller dans les propriétés du projet -> événement de génération -> événement après génération. Et modifier la ligne de commande par :

```
copy app.config "$(TargetPath).config"
```

Comment lire une valeur dans un fichier de configuration ?

Auteur : nico-pyright(c)

Tout d'abord, il faut créer le fichier de configuration app.config. Le plus simple ensuite est de travailler avec la section appSettings qui est gérée par le ConfigurationManager.
Dans le fichier, ajouter des clés et des valeurs comme ceci :

```
<configuration>
  <appSettings>
    <add key="nom" value="pyright"/>
    <add key="prenom" value="nico"/>
  </appSettings>
</configuration>
```

Ensuite, on peut y accéder ainsi :

```
String ^nom = Configuration::ConfigurationManager::AppSettings["nom"];
String ^prenom = Configuration::ConfigurationManager::AppSettings["prenom"];
Console::WriteLine("Je m'appelle {0} {1}", prenom, nom);
```

N'oubliez pas d'ajouter la référence à System.Configuration.

Comment écrire une valeur dans un fichier de configuration ?

Auteur : nico-pyright(c)

Il peut être utile de pouvoir modifier le fichier de configuration depuis son application. Voici comment faire :

```
String ^prenom = Configuration::ConfigurationSettings::AppSettings["prenom"];
Console::WriteLine(prenom);
```

```
Configuration::Configuration ^config = Configuration::ConfigurationManager::OpenExeConfiguration(Configuration
config->AppSettings->Settings->Remove("prenom");
config->AppSettings->Settings->Add("prenom", "Nouveau prenom");
config->Save(Configuration::ConfigurationSaveMode::Modified);
Configuration::ConfigurationManager::RefreshSection("appSettings");

prenom = Configuration::ConfigurationSettings::AppSettings["prenom"];
Console::WriteLine(prenom);
```

Comment crypter un fichier de configuration automatiquement en RSA ?

Auteur : nico-pyright(c)

On connait les fichiers de configuration d'application (app.config) (Comment lire une valeur dans un fichier de configuration ?).
Il est très pratique d'y stocker les informations de configuration de l'application. Il peut être tentant d'y stocker par exemple des informations sensibles (un login/mot de passe, ou une chaine de connexion de base de données par exemple).
Cependant, ces informations doivent être distribuées avec l'exécutable et donc librement consultable sous format XML.
Comment faire pour utiliser ce principe de configuration tout en conservant une sécurité adéquate ?

Plusieurs solutions viennent à l'esprit, comme d'implémenter un système de cryptage, à l'aide d'une clé d'encryption, comme par exemple Comment utiliser le cryptage avec l'algorithme Rijndael ?. Ceci implique d'avoir une méthode qui encrypte et qui décrypte les informations.

On peut également utiliser un mécanisme prévu par le framework.net pour un système d'encryption automatique en RSA.
Pour ce faire, on va déplacer l'écriture de la configuration dans le fichier machine.config. En utilisant toujours appsettings, ouvrez le fichier machine.config et si la section n'existe pas, créez la ainsi :

```
<appSettings>
  <add key="login" value="nico"/>
  <add key="pwd" value="abcd"/>
</appSettings>
```

Pour lire ces informations, on peut utiliser ce bout de code :

```
AppSettingsSection ^section = ConfigurationManager::OpenMachineConfiguration()->AppSettings;
String ^login = "";
String ^pwd = "";
if (section->Settings["login"] != nullptr)
  login = section->Settings["login"]->Value;
if (section->Settings["pwd"] != nullptr)
  pwd = section->Settings["pwd"]->Value;
Console::WriteLine("Login : {0} - Mot de passe : {1}", login, pwd);
```

La méthode statique OpenMachineConfiguration nous permet de lire dans le machine.config.
Lancez ensuite la commande :

```
aspnet_regiis.exe -pe "appSettings" -pkm
```

On se retrouve avec un fichier machine.config, dont la section appSettings ressemble désormais à ça :

```
<appSettings configProtectionProvider="RsaProtectedConfigurationProvider">
  <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
  xmlns="http://www.w3.org/2001/04/xmlenc#">
  <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
  <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
  <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
  <KeyName>Rsa Key</KeyName>
  </KeyInfo>
  <CipherData>
  <CipherValue>mCMSEOwI/sofISqaMxpaRM53IsBizjlBfsSzDAkzrGihm41P6R072
 FdkTkurkKBmyMjsXAI5oXlwYno7Ob8jws5i75yOjYWcy6y+uqEDTwfgIFt4kguVpEPbMlo6iR3dC0ipwmk0Eoz/TenDnSCt/e
+cmO2FEQXfqYulNRq9bDA=</CipherValue>
  </CipherData>
  </EncryptedKey>
  </KeyInfo>
  <CipherData>
  <CipherValue>TE6AddoMEjIwcAPi/mF0BBZHXfN2DZbaXzqK3SajSlv++HvGGdB
 ohgRPCboO69d5PLxPo3NjgMB5rEzqiGL49CTeapUtfNAzffa1owXzAjzQJnH/2WP3nrRci8qLhzOUYU3urGokAbN1zwiGY2adkA==</CipherVa
  </CipherData>
  </EncryptedData>
  </appSettings>
```

qui est tout de suite moins compréhensible.
Et ce qui est très pratique, c'est que sans aucune modification de code, mon application va être capable de déchiffrer cette section, et retrouver mon mot de passe par exemple.
Relancez l'application, et le tour est joué.

Comment retirer les accents d'une chaîne de caractères ?

Auteurs : Jérôme Lambert - nico-pyright(c)

```
static String ^RemplaceCaracteresAccentues(String ^input)
{
 array<unsigned char>^ objBytes = System::Text::Encoding::GetEncoding(1251)->GetBytes(input);
 return System::Text::Encoding::ASCII->GetString(objBytes);
}

int main()
{
 Console::Write("Entrez votre chaîne de carcatères : ");
 String ^strResultat = RemplaceCaracteresAccentues(Console::ReadLine());
 Console::WriteLine("Résultat : " + strResultat);
 return 0;
}
```

Sommaire > IDE

Comment commenter un bloc de code en une seule fois ?

Auteur : abelman

Si vous n'avez pas changé les paramétrages par défaut de visual studio.
Pour la sélection :
Après avoir sélectionné le texte, faire Ctrl+K puis Ctrl+C.
Pour la désélection :
Après avoir sélectionné le texte, faire Ctrl+K puis Ctrl+U.

Comment changer les raccourcis clavier de visual studio ?

Auteur : nico-pyright(c)

Si comme moi, vous êtiez habitués à certains raccourcis de visual c++ 6, par exemple le F7 pour faire un build et que désormais, la combinaison de touches par défaut est ctrl+alt+b, alors vous pourrez tirer avantage de ce changement de configuration.

On redéfinit les raccourcis clavier de visual studio en allant dans le menu : tools --> options --> environnment -> keyboard.
On observe ensuite une liste d'action pour lesquelles on peut définir un raccourci clavier.
On pourra par exemple modifier le raccourci associé à la commande Build.BuildSolution, mais plein d'autres que je vous laisse découvrir.
Remarque: L'IDE nous propose déjà une série de configuration types que nous pouvons utiliser simplement.

Comment spécifier les arguments de la ligne de commande en mode DEBUG ?

Auteur : abelman

- **Menu Projet**
- **Propriétés de "nom du projet" (tout en bas)**
- **Assurer vous que la configuration active est bien DEBUG. ComboBox en haut à gauche**
- **Clic sur propriétés de configuration dans la liste de gauche**
- **Clic sur Débogage toujours dans la liste de gauche.**
- **A droite choisir Arguments de ligne de commande et mettre vos arguments**

Comment détecter la compilation avec un mode /clr ?

Auteur : nico-pyright(c)

Il suffit d'utiliser la macro __cplusplus_cli. Cela permet notamment de détecter le mode /clr:oldSyntax.

```
#ifdef  __cplusplus_cli
    String ^ str = gcnew String("Chaine C++/CLI");
#else
    String* str = new String(S"Chaine oldSyntax");
#endif
```

Comment référencer une assembly externe dans son projet (#using) ?

Auteur : nico-pyright(c)

Afin de pouvoir utiliser des objets ou des structures définies dans des assemblys externe, il faut les référencer dans notre projet.
Pour ceci, deux solutions :

- **Soit en allant dans les propriétés du projet --> common properties --> references --> add new reference.**
  Ici il est important de constater qu'on peut référencer une librairie .Net ou COM. Il est également possible de référencer directement un projet interne à notre solution.
- **Soit en utilisant la directive suivante :**

```
#using "monAssembly.dll"
```

Remarque : Par défaut, l'assembly System.dll est référencée, ce qui permet en l'occurence d'accéder aux objets usuels du framework .Net.

NB : L'assembly se décrivant toute seule, il n'y a nul besoin d'inclure de .h pour l'utiliser.

Comment activer la numérotation des lignes dans un fichier source ?

Auteur : Thomas Lebrun

Allez dans le menu Tools et cliquez sur Options... Dans la liste de gauche, allez dans Text Editor. Si vous désirez activer la numérotation pour tous les languages, allez dans All Languages et cochez Line Numbers. Sinon, sélectionnez le language qui vous intéresse et cochez la case Line Numbers.

Compilation sans support du CLR

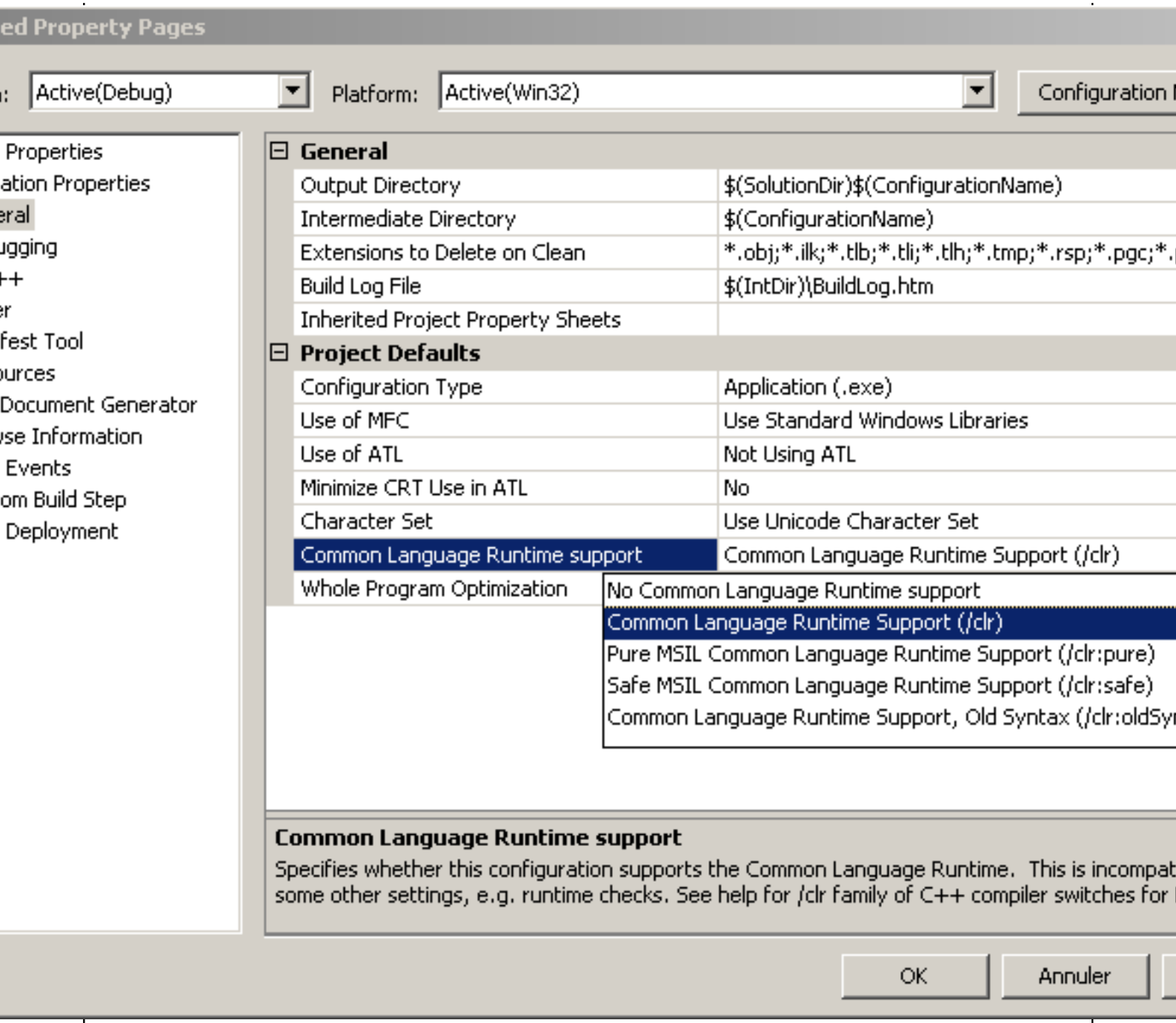
Auteur : nico-pyright(c)

Il s'agit du mode *No Common Language Runtime support*.
Ce mode (sans option /clr) est, vous l'aurez compris, le mode classique qui ne permet pas d'utiliser les extensions managées du framework .net. Le compilateur génère uniquement du langage machine dans ce mode là.
Autrement dit, le compilateur va générer un exécutable Win32 classique, indépendant de .Net qui ne pourra donc pas accéder aux fonctions des bibliothèques du framework .Net.

Compilation avec support mixte du CLR (/clr)

Auteur : nico-pyright(c)

Il s'agit du mode *Common Language Runtime Support*.
Cette option /clr permet de compiler du code C++ dans un mode mixte. C'est à dire que nous pourrons faire cohabiter du code natif avec du code managé. Le compilateur génère alors du MSIL. A partir du moment où on utilise un mode de compilation qui utilise /clr, le compilateur génère uniquement du MSIL.
Plus précisément, il pourra générer du langage machine lorsqu'il ne saura pas générer de MSIL, mais il génère prioritairement du MSIL.
La principale force de ce mode mixte est ce qu'on appelle IJW (It Just Work). Ce concept représente la capacité du compilateur à créer un exécutable "managé" à partir d'une application non managée.



Compilation avec support pur du CLR (/clr:pure)

Auteur : nico-pyright(c)

Il s'agit du mode *Pure MSIL Common Language Runtime Support*.
On utilise /clr:pure lorsque l'on n'utilise aucune classe non managée dans son application. Ce mode empêche la compilation de code natif mais autorise l'écriture de code qui dérogerait aux règles du CLS. Ce mode ne permet pas d'utiliser la technologie IJW.

Un des avantages de /clr:pure est que le code est sûr et possédant des performances meilleures qu'avec /clr uniquement. En effet, les assemblys purs contiennent uniquement du MSIL, il n'y a pas de fonctions natives et, par conséquent, aucune transition managée/non managée n'est nécessaire (hors appels P/Invoke).

Compilation avec support vérifiable du CLR (/clr:safe)

Auteur : nico-pyright(c)

Il s'agit du mode *Safe MSIL Common Language Runtime Support*.
Le mode /clr:safe génère des assemblys vérifiables, en se conformant aux exigences du CLS qui permettent au Common Language Runtime (CLR) de garantir que le code ne viole aucun paramètre de sécurité actuel.

Sachez dès à présent que les futures versions de Windows exigeront de plus en plus que les composants et les applications soient vérifiables ; donc commencer à écrire ses assemblys en mode /clr:safe est une bonne idée. (Par contre, cela rend impossible l'interopérabilité).

Compilation avec support de la syntaxe du framework 1.x du CLR (/clr:oldSyntax)

Auteur : nico-pyright(c)

Il s'agit du mode *Common Language Runtime Support, Old Syntax* .
Le mode /clr:oldsyntax est utilisé pour avoir une compatibilité avec les extensions managées du framework 1.x. Il est utile pour migrer en douceur son code pour se conformer à la syntaxe du framework 2.0. Cette syntaxe indigeste est à bannir au plus vite et à remplacer avantageusement par l'utilisation du C++/CLI.