



Ressourcesinformatiques

C# 6

et Visual Studio 2015

Les fondamentaux du langage

Sébastien PUTIER

Téléchargement
www.editions-eni.fr



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RI15CSHA** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1

La plateforme .NET

1. Introduction	15
2. Historique	17
3. Le Common Language Runtime (CLR)	22
4. La Base Class Library (BCL)	24
5. Le Dynamic Language Runtime (DLR)	25
6. Évolution de la plateforme	26
6.1 .NET Core	27
6.2 .NET Compiler Platform : Roslyn	28
6.3 .NET dans le monde open source	28
7. Une première application avec Visual C#	29
7.1 Création	29
7.2 Compilation	31
7.3 Analyse de l'assembly	33
7.3.1 Structure	33
7.3.2 Exploration avec ILDASM	34

2 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

Chapitre 2 Visual Studio 2015

1. Installation et premier lancement	39
1.1 Prérequis	39
1.2 Éditions de Visual Studio	40
1.2.1 Visual Studio Express	41
1.2.2 Visual Studio Community	41
1.2.3 Éditions commerciales	42
1.3 Installation	43
1.4 Premier lancement	47
2. Description des outils	53
2.1 Barres d'outils	58
2.2 Explorateur de solutions	59
2.3 Explorateur d'objets	60
2.4 Explorateur de serveurs	61
2.5 Fenêtre de propriétés	65
2.6 Fenêtre d'édition de code	68
2.6.1 Navigation	68
2.6.2 Suivi des modifications	68
2.6.3 Mise en surbrillance des références	69
2.6.4 Refactorisation	70
2.6.5 IntelliSense	72
2.6.6 Snippets	72

Chapitre 3 L'organisation d'une application

1. Les solutions	75
1.1 Présentation	75
1.2 Création d'une solution	76
1.3 Organisation	77

1.4	Actions disponibles sur une solution	78
1.4.1	Ajout et suppression d'un projet	78
1.4.2	Création d'un dossier de solution	80
1.4.3	Chargement et déchargement d'un projet	80
1.4.4	Création d'un fichier	81
1.4.5	Génération de la solution	81
1.5	Configuration de la solution	82
1.5.1	Configuration des projets de démarrage	82
1.5.2	Dépendances du projet	84
1.5.3	Paramètres d'analyse du code	85
1.5.4	Fichiers sources pour le débogage	86
1.5.5	Configuration	87
2.	Les projets	88
2.1	Création d'un projet	88
2.2	Propriétés d'un projet	96
2.2.1	Application	97
2.2.2	Générer	100
2.2.3	Événements de build	103
2.2.4	Déboguer	105
2.2.5	Ressources	106
2.2.6	Paramètres	107

Chapitre 4
Les bases du langage

1.	Introduction	109
2.	Les variables	109
2.1	Nommage des variables	110
2.2	Type des variables	111
2.2.1	Types valeurs et types références	111
2.2.2	Types intégrés	112
2.3	Déclaration des variables	116
2.4	Portée des variables	116

4 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

2.5	Modificateurs d'accès	117
2.6	Le mot-clé var et l'inférence de type	118
3.	Les constantes.	119
4.	Les opérateurs.	119
4.1	Les opérateurs d'accès.	119
4.1.1	Accès simple : . (point).	120
4.1.2	Accès indexé : []	120
4.1.3	Accès avec nullité conditionnelle : ?	120
4.2	Les opérateurs arithmétiques.	121
4.3	Les opérateurs de comparaison	121
4.4	Les opérateurs conditionnels	122
4.4.1	Opérateur ternaire : ? ... :	122
4.4.2	Opérateur de fusion de valeur nulle : ??	123
4.5	Les opérateurs logiques.	123
4.5.1	Négation : !	123
4.5.2	ET logique : &	124
4.5.3	OU logique : 	124
4.5.4	OU exclusif : ^	124
4.5.5	ET conditionnel : &&	125
4.5.6	OU conditionnel : 	125
4.6	Les opérateurs binaires	126
4.6.1	ET binaire : &	126
4.6.2	OU binaire : 	126
4.6.3	OU exclusif : ^	127
4.6.4	Négation : ~	127
4.6.5	Décalage vers la droite : >>	127
4.6.6	Décalage vers la gauche : <<	128
5.	Les structures de contrôle	128
5.1	Les structures conditionnelles	128
5.1.1	if ... else	128
5.1.2	switch	130
5.2	Les structures d'itération	131

5.2.1	for	131
5.2.2	while	132
5.2.3	do ... while	133
5.2.4	foreach	133
5.2.5	Contrôler l'exécution d'une boucle	134
5.3	Autres structures	135
5.3.1	using	135
5.3.2	goto	136
6.	Les fonctions	136
6.1	Écriture d'une fonction	137
6.2	Paramètres de fonction	138
6.3	Procédures	143
6.4	Surcharges	144
7.	Les attributs	145

Chapitre 5

La programmation orientée objet avec C#

1.	Les principes de la programmation orientée objet	147
2.	Les classes et les structures	150
2.1	Classes	150
2.1.1	Déclaration	151
2.1.2	Constructeur et destructeur	153
2.1.3	Classes partielles	157
2.2	Structures	158
2.3	Création de méthodes	159
2.3.1	Création	159
2.3.2	Méthodes partielles	161
2.3.3	Méthodes d'extension	162
2.3.4	Méthodes opérateurs	163

6 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

2.4	Création de propriétés	165
2.4.1	Lecture et écriture	165
2.4.2	Lecture seule	166
2.4.3	Écriture seule	167
2.4.4	Propriétés automatiques	167
2.4.5	Initialisation de propriétés automatiques	168
2.4.6	Propriétés automatiques en lecture seule	168
2.4.7	Propriétés indexées	169
2.5	Membres statiques	170
2.6	Utilisation des classes et structures	171
2.6.1	Instanciation	171
2.6.2	Initialisation	172
2.6.3	Types anonymes	172
3.	Les espaces de noms	175
3.1	Nomenclature	176
3.2	using	177
4.	L'héritage	178
4.1	Mise en œuvre	179
4.2	Les mots-clés this et base	180
4.3	Redéfinition et masquage	182
4.3.1	Redéfinition de méthode	182
4.3.2	Masquage de méthode	183
4.3.3	Différences entre redéfinition et masquage	183
4.4	Imposer ou interdire l'héritage	186
4.5	Le transtypage	187
5.	Les interfaces	190
5.1	Création	190
5.2	Utilisation	191
5.2.1	Implémentation implicite	192
5.2.2	Implémentation explicite	194
6.	Les énumérations	196

7. Les délégués.....	196
7.1 Création.....	197
7.2 Utilisation.....	197
7.3 Expressions lambda.....	198
8. Les événements.....	199
8.1 Déclaration et déclenchement.....	199
8.2 Gestion des événements.....	201
9. Les génériques.....	203
9.1 Classes.....	203
9.1.1 Définition d'une classe générique.....	204
9.1.2 Utilisation d'une classe générique.....	204
9.2 Interfaces.....	205
9.2.1 Définition d'une interface générique.....	205
9.2.2 Utilisation d'une interface générique.....	206
9.3 Contraintes.....	207
9.4 Méthodes.....	210
9.4.1 Définition d'une méthode générique.....	210
9.4.2 Utilisation d'une méthode générique.....	212
9.5 Événements et délégués.....	212
10. Les collections.....	214
10.1 Types existants.....	214
10.1.1 Array.....	214
10.1.2 ArrayList et List<T>.....	215
10.1.3 Hashtable et Dictionary<TKey, TValue>.....	219
10.1.4 Stack et Stack<T>.....	221
10.1.5 Queue et Queue<T>.....	222
10.2 Choisir un type de collection.....	223
11. La programmation dynamique.....	223
12. La programmation asynchrone.....	226
12.1 Les objets Task.....	226
12.2 Écrire du code asynchrone avec async et await.....	229

Chapitre 6

Débogage et gestion des erreurs

1. Les différents types d'erreurs	231
1.1 Erreurs de compilation	231
1.2 Erreurs d'exécution	233
2. Utilisation des exceptions	234
2.1 Création et déclenchement d'exceptions	234
2.1.1 La classe Exception	234
2.1.2 Le mot-clé throw	235
2.1.3 Exceptions spécialisées	235
2.2 Gérer les exceptions	236
2.2.1 La structure try ... catch	236
2.2.2 Les filtres d'exception	239
2.2.3 Le bloc finally	240
3. Les outils fournis par Visual Studio	242
3.1 Contrôle de l'exécution	242
3.1.1 Démarrage	243
3.1.2 Arrêt	244
3.1.3 Pause	244
3.1.4 Reprise	245
3.2 Points d'arrêt	245
3.3 Visualiser le contenu des variables	252
3.3.1 DataTips	252
3.3.2 Fenêtres Espion	253
3.3.3 Fenêtre Espion express	254
3.3.4 Fenêtre Variables locales	255
3.4 Compilation conditionnelle	255

Chapitre 7
Développement d'applications Windows

- 1. Présentation de WPF. 259
 - 1.1 Structure d'une application WPF. 260
 - 1.2 XAML 261
 - 1.2.1 Templates 262
 - 1.2.2 Espaces de noms 263
 - 1.3 Contexte de données et binding 264
- 2. Utilisation des contrôles. 268
 - 2.1 Ajout de contrôles 269
 - 2.2 Positionnement et dimensionnement des contrôles 272
 - 2.3 Ajout d'un gestionnaire d'événements à un contrôle 275
- 3. Les principaux contrôles. 277
 - 3.1 Contrôles de fenêtrage 278
 - 3.1.1 Window. 278
 - 3.1.2 NavigationWindow 280
 - 3.2 Contrôles de disposition. 282
 - 3.2.1 Grid 282
 - 3.2.2 StackPanel 286
 - 3.2.3 DockPanel 286
 - 3.2.4 WrapPanel 288
 - 3.2.5 Canvas. 290
 - 3.3 Contrôles d'affichage de données 290
 - 3.3.1 TextBlock 291
 - 3.3.2 Label 292
 - 3.3.3 Image. 293
 - 3.3.4 ScrollViewer 294
 - 3.3.5 ItemsControl. 294
 - 3.3.6 StatusBar. 298
 - 3.3.7 ToolTip 298

10 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

3.4	Contrôles d'édition de texte.....	299
3.4.1	TextBox.....	299
3.4.2	RichTextBox.....	300
3.4.3	PasswordBox.....	302
3.5	Contrôles de sélection.....	302
3.5.1	RadioButton.....	302
3.5.2	CheckBox.....	303
3.5.3	ComboBox.....	303
3.5.4	ListBox.....	304
3.5.5	ListView.....	306
3.5.6	TreeView.....	308
3.5.7	Slider.....	313
3.5.8	Calendar.....	313
3.5.9	DatePicker.....	314
3.6	Contrôles d'action.....	315
3.6.1	Button.....	315
3.6.2	Menu.....	315
3.6.3	ContextMenu.....	318
3.6.4	ToolBar.....	319
4.	Interactions clavier et souris.....	320
4.1	Événements clavier.....	320
4.2	Événements souris.....	323
4.3	Glisser-déposer.....	324
5.	Aller plus loin avec WPF.....	328
5.1	Introduction à l'utilisation de Blend.....	328
5.1.1	L'interface.....	329
5.1.2	Ajout et modification de contrôles visuels.....	337
5.2	Introduction à MVVM.....	340
5.2.1	Présentation.....	340
5.2.2	Les interfaces INotifyPropertyChanged et INotifyCollectionChanged.....	341
5.2.3	Commandes.....	342
5.2.4	Mise en œuvre.....	342

Chapitre 8
Accès aux données

- 1. Principes d'une base de données 359
 - 1.1 Terminologie..... 359
 - 1.2 Le langage SQL 360
 - 1.2.1 Recherche d'enregistrements 361
 - 1.2.2 Ajout d'enregistrements..... 363
 - 1.2.3 Mise à jour d'informations..... 363
 - 1.2.4 Suppression d'informations 364
- 2. ADO.NET 364
 - 2.1 Présentation 364
 - 2.2 Les fournisseurs de données..... 365
 - 2.2.1 SQL Server..... 366
 - 2.2.2 Oracle 366
 - 2.2.3 OLE DB 367
 - 2.2.4 ODBC 367
- 3. Utiliser ADO.NET en mode connecté..... 368
 - 3.1 Connexion à une base de données..... 368
 - 3.1.1 Chaînes de connexion 368
 - 3.1.2 Pools de connexions 371
 - 3.1.3 Gestion de la connexion..... 373
 - 3.2 Création et exécution de commandes 375
 - 3.2.1 Définition et création d'une commande 375
 - 3.2.2 Sélection de données 376
 - 3.2.3 Actions sur les données 377
 - 3.2.4 Paramétrage d'une commande..... 378
 - 3.2.5 Exécution de procédures stockées 381
- 4. Utiliser ADO.NET en mode déconnecté..... 383
 - 4.1 DataSet et DataTable..... 383
 - 4.1.1 Description 383
 - 4.1.2 Remplissage d'un DataSet
à partir d'une base de données 384

12 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

4.1.3	Remplissage d'un DataSet sans base de données	387
4.2	Manipulation des données hors connexion	390
4.2.1	Lecture des données	391
4.2.2	Création de contraintes	391
4.2.3	Relations entre DataTables	395
4.2.4	État et versions d'une DataRow	397
4.2.5	Modification de données	398
4.2.6	Suppression de données	400
4.2.7	Valider ou annuler des modifications	400
4.2.8	Filtrage et tri à l'aide d'une DataView	401
4.2.9	Recherche de données	404
4.3	Valider les modifications au niveau de la base de données . . .	406
4.3.1	Générer des commandes de mise à jour automatiquement.	407
4.3.2	Commandes de mise à jour personnalisées.	409
4.3.3	Gestion des accès concurrentiels	410
5.	Utiliser les transactions	412

Chapitre 9

LINQ

1.	Présentation de LINQ.	415
2.	Syntaxe	416
2.1	Une première requête LINQ	419
2.2	Les opérateurs de requête.	422
2.2.1	Projection	422
2.2.2	Filtrage.	424
2.2.3	Triage	426
2.2.4	Partitionnement	427
2.2.5	Jointure et regroupement.	428
2.2.6	Agrégation.	431

- 3. LINQ to SQL.....433
 - 3.1 Le mappage objet-relationnel.....433
 - 3.1.1 Utilisation de SQLMetal434
 - 3.1.2 Utilisation du concepteur objet/relationnel.....440
 - 3.2 Utilisation de LINQ to SQL.....449
 - 3.2.1 Récupération de données.....449
 - 3.2.2 Mise à jour de données.....451
 - 3.2.3 Gestion des conflits453

Chapitre 10
XML

- 1. Présentation457
- 2. Structure d'un fichier XML458
 - 2.1 Constituants d'un document XML.....458
 - 2.2 Document bien formé et document valide.....462
- 3. Manipuler un document XML.....463
 - 3.1 Utilisation de DOM.....464
 - 3.2 Utilisation de XPath.....469
 - 3.3 Utilisation de LINQ to XML.....472

Chapitre 11
Déploiement

- 1. Introduction477
- 2. Windows Installer478
 - 2.1 Installation de InstallShield Limited Edition479
 - 2.2 Création d'un projet d'installation482
 - 2.2.1 Informations sur l'application483
 - 2.2.2 Prérequis d'installation.....484
 - 2.2.3 Fichiers de l'application485
 - 2.2.4 Raccourcis488
 - 2.2.5 Valeurs de la base de registre489

14 _____ C# 6 et Visual Studio 2015

Les fondamentaux du langage

2.2.6 Boîtes de dialogue	490
3. ClickOnce	491
3.1 La technologie ClickOnce	491
3.1.1 Principes de fonctionnement	492
3.1.2 Méthodes de déploiement disponibles	493
3.1.3 Les mises à jour d'applications avec ClickOnce	494
3.2 La publication ClickOnce	496

Chapitre 12

Aide-mémoire

1. Introduction	505
Index	525

Chapitre 4

Les bases du langage

1. Introduction

Comme tous les langages de programmation, C# impose certaines règles au développeur. Ces règles se manifestent au travers de la syntaxe du langage mais elles couvrent le large spectre fonctionnel proposé par C#. Avant d'explorer en profondeur les fonctionnalités du langage au chapitre suivant, nous étudierons donc les notions essentielles et fondamentales de C# : la création de données utilisables par une application et le traitement de ces données.

2. Les variables

Les données utilisables dans un programme C# sont représentées par des variables. Une variable est un espace mémoire réservé auquel on assigne arbitrairement un nom et dont le contenu est une valeur dont le type est fixé. On peut manipuler ce contenu dans le code en utilisant le nom de la variable.

2.1 Nommage des variables

La spécification du langage C# établit quelques règles à prendre en compte lorsque l'on nomme une variable :

- Le nom d'une variable ne peut comporter que des chiffres, des caractères de l'alphabet latin accentués ou non, le caractère ç ou les caractères spéciaux `_` et `µ`.
- Le nom d'une variable ne peut en aucun cas commencer par un chiffre. Il peut en revanche en comporter un ou plusieurs à toute autre position.
- Le langage est sensible à la casse, c'est-à-dire qu'il fait la distinction entre majuscules et minuscules : les variables `unevariable` et `uneVariable` sont donc différentes.
- Un nom de variable ne peut pas excéder 511 caractères. Il n'est évidemment pas recommandé d'avoir des noms de variable aussi longs, le maximum en pratique étant plus souvent de l'ordre de la trentaine de caractères.
- Le nom d'une variable ne peut pas être un mot-clé du langage. Il est toutefois possible de préfixer un mot-clé par un caractère autorisé ou par le caractère `@` pour utiliser un nom de variable similaire.

Les noms de variables suivants sont acceptés par le compilateur C# :

- `maVariable`
- `maVariableNumerol`
- `@void` (`void` est un mot-clé de C#)
- `µn3_VàRiãbl3`

De manière générale, il est préférable d'utiliser des noms de variables explicites, c'est-à-dire permettant de savoir à quoi correspond la valeur stockée dans la variable, comme `nomClient`, `montantAchat` ou `ageDuCapitaine`.

2.2 Type des variables

Une des caractéristiques de C# est la notion de typage statique : chaque variable correspond à un type de données et ne peut en changer. De plus, ce type doit être déterminable au moment de la compilation.

2.2.1 Types valeurs et types références

Les différents types utilisables en C# peuvent être décomposés en deux familles : les types valeurs et les types références. Cette notion peut être déconcertante au premier abord, puisqu'une variable représente justement une donnée et donc une valeur. Ce concept est en fait lié à la manière dont est stockée l'information en mémoire.

Lorsque l'on utilise une variable de type valeur, on accède directement à la zone mémoire stockant la donnée. Au moment de la création d'une variable de type valeur, une zone mémoire de la taille correspondant au type est allouée. Chaque octet de cette zone est automatiquement initialisé avec la valeur binaire 00000000. Notre variable aura donc une suite de 0 pour valeur binaire.

Dans le cas d'une variable de type référence, le comportement est différent. La zone mémoire allouée à notre variable contient une adresse mémoire à laquelle est stockée la donnée. On passe donc par un intermédiaire pour accéder à notre donnée. L'adresse mémoire est initialisée avec la valeur spéciale `null`, qui ne pointe sur rien, tandis que la zone mémoire contenant les données n'est pas initialisée. Elle sera initialisée lorsque la variable sera instanciée. Dans le même temps, l'adresse mémoire stockée dans notre variable sera mise à jour.

Une variable de type référence pourra donc ne contenir aucune donnée, tandis qu'une variable de type valeur aura forcément une valeur correspondant à une suite de 0 binaires.

Cette différence de fonctionnement a une conséquence importante : la copie de variable se fait par valeur ou par référence, ce qui signifie qu'une variable de type valeur sera effectivement copiée, tandis que pour un type référence, c'est l'adresse que contient la variable qui sera copiée, et il sera donc possible d'agir sur la donnée réelle indifféremment à partir de chacune des variables pointant sur ladite donnée.

2.2.2 Types intégrés

Le framework .NET embarque plusieurs milliers de types différents utilisables par les développeurs. Parmi ces types, nous en avons une quinzaine que l'on peut considérer comme fondamentaux : ce sont les types intégrés (aussi nommés types primitifs). Ce sont les types de base à partir desquels sont construits les autres types de la BCL ainsi que ceux que le développeur crée dans son propre code. Ils permettent de définir des variables contenant des données très simples.

Ces types ont comme particularité d'avoir chacun un alias intégré à C#.

Types numériques

Ces types permettent de définir des variables numériques entières ou décimales. Ils couvrent des plages de valeurs différentes et ont chacun une précision spécifique. Certains types seront donc plus adaptés pour les calculs entiers, d'autres pour les calculs dans lesquels la précision décimale est très importante, comme les calculs financiers.

Les différents types numériques sont énumérés ci-dessous avec leurs alias ainsi que les plages de valeurs qu'ils couvrent.

Type	Alias C#	Plage de valeurs couverte	Taille en mémoire
System.Byte	byte	0 à 255	1 octet
System.SByte	sbyte	-128 à 127	1 octet
System.Int16	short	-32768 à 32767	2 octets
System.UInt16	ushort	0 à 65535	2 octets
System.Int32	int	-2147483648 à 2147483647	4 octets
System.UInt32	uint	0 à 4294967295	4 octets
System.Int64	long	-9223372036854775808 à 9223372036854775807	8 octets
System.UInt64	ulong	0 à 18446744073709551615	8 octets
System.Single	float	±1,5e-45 à ±3,4e38	4 octets

Type	Alias C#	Plage de valeurs couverte	Taille en mémoire
System.Double	double	$\pm 5,0e-324$ à $\pm 1,7e308$	8 octets
System.Decimal	decimal	$\pm 1,0e-28$ à $\pm 7,9e28$	16 octets

Les types numériques primitifs sont tous des types valeurs. Une variable de type numérique et non initialisée par le développeur aura pour valeur par défaut 0.

■ Remarque

.NET 4 a apporté le type `System.Numerics.BigInteger` afin de manipuler des entiers d'une taille arbitraire. Ce type est aussi un type valeur, mais il ne fait pas partie des types intégrés.

Types textuels

Il existe dans la BCL deux types permettant de manipuler des caractères Unicode et des chaînes de caractères Unicode : `System.Char` et `System.String`. Ces types ont respectivement pour alias `char` et `string`.

Le type `char` est un type valeur encapsulant les mécanismes nécessaires au traitement d'un caractère Unicode. Par conséquent, une variable de type `char` est stockée en mémoire sur deux octets.

Les valeurs de type `char` doivent être encadrées par les caractères `'` : `'a'`.

Certains caractères ayant une signification particulière pour le langage doivent être utilisés avec le caractère d'échappement `\` afin d'être correctement interprétés. D'autres caractères n'ayant pas de représentation graphique doivent être aussi déclarés avec des séquences spécifiques. Le tableau suivant résume les séquences qui peuvent être utilisées.

Séquence d'échappement	Caractère associé
<code>\'</code>	Simple quote <code>'</code>
<code>\"</code>	Double quote <code>"</code>
<code>\\</code>	Backslash <code>\</code>

Séquence d'échappement	Caractère associé
\a	Alerte sonore
\b	Retour arrière
\f	Saut de page
\n	Saut de ligne
\r	Retour chariot
\t	Tabulation horizontale
\v	Tabulation verticale
\0	Caractère nul
\uXXXX	Caractère Unicode dont le code hexadécimal est XXXX

Le type `string` manipule en interne des tableaux d'objets de type `char` pour permettre le traitement de chaînes pouvant représenter jusqu'à 4 Go, soit 2 147 483 648 caractères. Contrairement aux types intégrés vus jusqu'ici, le **type `string` est un type référence**, ce qui signifie donc qu'une variable de ce type peut avoir la valeur `null`.

Une chaîne de caractère s'écrit entre les caractères `" "` : "une chaîne de caractères sympathique et un peu longue". Tout comme pour les `char`, certains caractères nécessitent d'utiliser une séquence d'échappement.

Il est à noter que les chaînes de caractères sont invariables, c'est-à-dire qu'elles ne peuvent pas être modifiées. Heureusement, le framework nous permet de le faire mais à un certain prix. Chaque modification effectuée sur une chaîne de caractères entraîne la création d'une nouvelle chaîne incluant la modification. Ce comportement, transparent pour nous, peut entraîner des problèmes de performance lorsqu'il est question de traitements importants sur ce type d'objets. Il est donc important de connaître cette subtilité afin d'éviter ce type de problèmes.