

vBook

livre &
vidéo

C# 5 sous Visual Studio 2012



Les fondamentaux du langage



Les bonnes pratiques
de développement
(LINQ, génériques, délégues)

1 H 30 de vidéo

Thierry DOUCHET
Thierry GROUSSARD

Téléchargement
www.editions-eni.fr


Editions

Les éléments à télécharger sont disponibles à l'adresse suivante :

<http://www.editions-eni.fr>

Saisissez la référence ENI de l'ouvrage **RI12CSHA** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1 Présentation de la plate-forme .NET

1.	Introduction	15
1.1	Principe de fonctionnement du Common Language Runtime	17
1.2	Les services du Common Language Runtime	18
1.3	La Base Class Library	20
1.4	Les versions et évolutions de la plate-forme .NET	21
2.	Écriture, compilation et exécution d'une application	25
2.1	Écriture du code	25
2.2	Compilation du code	27
2.3	Analyse d'un fichier compilé	30
2.4	Exécution du code.....	39

Chapitre 2 Présentation de Visual Studio

1.	Installation et premier démarrage	41
1.1	Configuration nécessaire	41
1.2	Premier démarrage	46
2.	Découverte de l'environnement	48
2.1	Page de démarrage	48
2.2	Environnement Visual Studio	49

3.	Les outils disponibles	51
3.1	Les barres d'outils	51
3.2	La boîte à outils	54
3.3	L'explorateur de serveurs	55
3.4	L'explorateur de solutions	57
3.5	L'affichage de classes	58
3.6	La fenêtre de propriétés	59
3.7	La liste des tâches	60
3.8	La liste des erreurs	62
3.9	La fenêtre d'édition de code	62
3.9.1	Les Snippets	62
3.9.2	Suivi des modifications	68
3.9.3	Les outils d'édition de code	70

Chapitre 3

Organisation d'une application

1.	Les solutions	77
1.1	Présentation	77
1.2	Création d'une solution	77
1.3	Modification d'une solution	79
1.3.1	Ajouter un projet	80
1.3.2	Supprimer un projet	80
1.3.3	Renommer un projet	81
1.3.4	Décharger un projet	81
1.4	Organisation d'une solution	82
1.4.1	Création d'un dossier de solution	82
1.4.2	Créer un projet dans un dossier	82
1.4.3	Déplacer un projet dans un dossier	83
1.5	Le dossier Éléments de solution	83
1.6	Le dossier Fichiers divers	83

1.7 Configuration d'une solution	86
1.7.1 Configuration du projet de démarrage	86
1.7.2 Dépendances du projet.....	87
1.7.3 Paramètres d'analyse du code.....	89
1.7.4 Fichiers source pour le débogage	90
1.7.5 Configurations	91
2. Les projets	92
2.1 Création d'un projet.....	92
2.1.1 Les modèles de projets	93
2.1.2 Création de modèle de projet.....	99
2.1.3 Modification d'un modèle existant.....	101
2.1.4 Utilisation d'un projet existant comme modèle.....	101
2.2 Modification d'un projet	104
2.3 Propriétés des projets	107
2.3.1 Application	108
2.3.2 Générer	112
2.3.3 Événements de build	116
2.3.4 Déboguer.....	118
2.3.5 Ressources.....	119
2.3.6 Paramètres d'application	120
2.3.7 Autres paramètres de configuration	123

Chapitre 4

Bases du langage

1. Les variables, constantes et énumérations	125
1.1 Les variables	125
1.1.1 Nom des variables.....	125
1.1.2 Type des variables.....	126
1.1.3 Conversions de types	135
1.1.4 Déclaration des variables	141
1.1.5 Inférence de type	142
1.1.6 Portée des variables.....	143

1.1.7 Niveau d'accès des variables	144
1.1.8 Durée de vie des variables	145
1.2 Les constantes	145
1.3 Les énumérations	146
1.4 Les tableaux	148
1.5 Les chaînes de caractères	152
1.6 Les structures	156
1.6.1 Déclaration d'une structure	156
1.6.2 Utilisation des structures	158
2. Les opérateurs	160
2.1 Les opérateurs d'affectation	160
2.2 Les opérateurs arithmétiques	160
2.3 Les opérateurs binaires	161
2.4 Les opérateurs de comparaison	161
2.5 Opérateur de concaténation	162
2.6 Les opérateurs logiques	163
2.7 Ordre d'évaluation des opérateurs	164
3. Les structures de contrôle	165
3.1 Structures de décision	165
3.1.1 Structure if	165
3.1.2 Structure switch	166
3.2 Les structures de boucle	167
3.2.1 Structure while	168
3.2.2 Structure do ... while	168
3.2.3 Structure for	168
3.2.4 Structure foreach	170
3.2.5 Autres structures	171
4. Les procédures et fonctions	171
4.1 Procédure	172
4.2 Fonction	173
4.3 Procédures de propriétés	174
4.4 Les procédures opérateur	176

4.5	Les arguments des procédures et fonctions	177
4.6	Fonctions asynchrones	182
5.	Assemblies, Namespace et attributs	188
5.1	Les assemblies	188
5.2	Les Namespaces	191
5.3	Les attributs	194
5.3.1	Attributs les plus courants en Visual C#	195

Chapitre 5

Programmation objet

1.	Introduction	199
2.	Mise en œuvre avec Visual C#	202
2.1	Création d'une classe	203
2.1.1	Déclaration de la classe	203
2.1.2	Classe partielle	205
2.1.3	Création de propriétés	206
2.1.4	Création de méthodes	214
2.1.5	Constructeurs et destructeurs	221
2.1.6	Membres partagés	224
2.2	Utilisation d'une classe	225
2.2.1	Création d'une instance	225
2.2.2	Initialisation d'une instance	226
2.2.3	Destruction d'une instance	227
2.2.4	Liaison tardive, liaison précoce	228
2.3	Héritage	230
2.3.1	base et this	231
2.3.2	Classes abstraites	235
2.3.3	Classes finales	235
2.3.4	Classes anonymes	236
2.4	Interfaces	239

2.5 Les événements	243
2.5.1 Déclaration et déclenchement d'événements	244
2.5.2 Gérer les événements	246
2.6 Les délégués	248
2.6.1 Déclaration et création d'un délégué	248
2.6.2 Utilisation des délégués	250
2.6.3 Expressions lambda	250
3. Les types génériques	252
3.1 Les classes génériques	254
3.1.1 Définition d'une classe générique	254
3.1.2 Utilisation d'une classe générique	259
3.2 Interfaces génériques	261
3.2.1 Définition d'une interface générique	262
3.2.2 Utilisation d'une interface générique	262
3.3 Procédures et fonctions génériques	263
3.3.1 Création d'une procédure ou fonction générique	263
3.3.2 Utilisation d'une procédure ou fonction générique	264
3.4 Délégués génériques	265
3.5 Variance	267
3.5.1 Variance dans les interfaces génériques	267
3.5.2 Variance dans les délégués génériques	274
4. Les collections	278
4.1 Les collections prédéfinies	278
4.1.1 Array	279
4.1.2 ArrayList et List	279
4.1.3 Hashtable et Dictionary	282
4.1.4 Queue	283
4.1.5 Stack	284
4.2 Choisir un type de collection	284

Chapitre 6
Gestion des erreurs et débogage du code

1.	Les différents types d'erreurs	285
1.1	Les erreurs de syntaxe.	285
1.2	Les erreurs d'exécution.	288
2.	Traitements des exceptions.	289
2.1	Récupération d'exceptions.	289
2.1.1	Création et déclenchement d'exceptions	294
3.	Les outils de débogage	295
3.1	Contrôle de l'exécution	297
3.1.1	Démarrage de la solution	297
3.1.2	Arrêter la solution.	297
3.1.3	Interrompre la solution	298
3.1.4	Poursuivre l'exécution	298
3.2	Points d'arrêt et TracePoint.	301
3.2.1	Placer un point d'arrêt	302
3.2.2	Activer, désactiver, supprimer un point d'arrêt	308
3.3	Examen du contenu de variables	309
3.3.1	DataTips	309
3.3.2	Fenêtre Automatique	310
3.3.3	Fenêtre Variables locales	311
3.3.4	Les fenêtres Espion	312
3.3.5	La fenêtre Espion express	313
3.4	Les autres fenêtres de débogage.	314
4.	Autres techniques de débogage	315

Chapitre 7

Applications Windows

1.	Les applications Windows	317
2.	Les fenêtres	318
2.1	Dimension et position des fenêtres.	321
2.2	Couleurs et Police utilisées sur les fenêtres.	327
2.3	Les fenêtres MDI	329
3.	Les événements clavier et souris	336
3.1	Les événements clavier.	336
3.2	Les événements souris	339
3.3	Le Drag and Drop	343
3.3.1	Démarrage du Drag and Drop	344
3.3.2	Configuration des contrôles pour la réception	345
3.3.3	Récupération de l'élément accroché	346
4.	Les boîtes de dialogue.	347
4.1	La boîte de message	348
4.2	Les boîtes de dialogue de Windows.	351
4.2.1	Dialogue d'ouverture de fichier	351
4.2.2	Dialogue d'enregistrement de fichier	353
4.2.3	Dialogue de choix de répertoire	353
4.2.4	Dialogue de choix d'une couleur	354
4.2.5	Dialogue de choix d'une police	357
4.2.6	Dialogue de mise en page.	360
4.2.7	Dialogue de configuration d'impression.	362
4.3	Boîte de dialogue personnalisée	363
5.	Utilisation des contrôles	364
5.1	Ajout de contrôles	365
5.2	Position et dimension des contrôles	367
5.3	Passage du focus entre contrôles	374
5.4	Raccourcis-clavier	376

6.	Les contrôles	379
6.1	La classe Control	379
6.1.1	Dimensions et position	379
6.1.2	Apparence des contrôles.	382
6.1.3	Comportement des contrôles	384
6.2	Les contrôles d'affichage d'informations	388
6.2.1	Le contrôle Label.	388
6.2.2	Le contrôle LinkLabel	390
6.2.3	Le contrôle StatusStrip.	392
6.2.4	Le contrôle ToolTip	393
6.2.5	Le Contrôle ErrorProvider	394
6.2.6	Le contrôle NotifyIcon.	395
6.2.7	Le contrôle HelpProvider	396
6.2.8	Le contrôle ProgressBar	397
6.3	Les contrôles d'édition de texte	399
6.3.1	Le contrôle TextBox	399
6.3.2	Le contrôle MaskedTextBox	403
6.3.3	Le contrôle RichTextBox	404
6.4	Les contrôles de déclenchement d'actions	406
6.4.1	Le contrôle Button	406
6.4.2	Le contrôle ToolStrip	407
6.4.3	Le menu ContextMenuStrip	412
6.4.4	Le contrôle ToolStrip	413
6.4.5	Le contrôle ToolStripContainer.	414
6.5	Contrôles de sélection	416
6.5.1	Le contrôle CheckBox.	416
6.5.2	Le contrôle RadioButton	418
6.5.3	Le contrôle ListBox.	420
6.5.4	Le contrôle NumericUpDown	424
6.5.5	Le contrôle TrackBar	425
6.5.6	Le contrôle DomainUpDown	425
6.5.7	Le contrôle CheckedListBox.	426
6.5.8	Le contrôle ComboBox.	426

6.5.9 Le contrôle TreeView	428
6.5.10 Le contrôle ListView	431
6.6 Les contrôles de regroupement	436
6.6.1 Le contrôle GroupBox	436
6.6.2 Le contrôle Panel	436
6.6.3 Le contrôle TabControl	438
6.6.4 Le contrôle SplitContainer	440
6.6.5 Le contrôle FlowLayoutPanel	441
6.6.6 Le contrôle TableLayoutPanel	442
6.7 Les contrôles graphiques	445
6.7.1 Le contrôle PictureBox	445
6.7.2 Le contrôle ImageList	447
6.8 Les contrôles de gestion du temps	449
6.8.1 Le contrôle DateTimePicker	449
6.8.2 Le contrôle MonthCalendar	450
6.8.3 Le contrôle Timer	452
6.8.4 Le composant BackGroundWorker	453
7. L'héritage de formulaires	457

Chapitre 8

Accès aux bases de données

1. Principe de fonctionnement d'une base de données	461
1.1 Terminologie	461
1.2 Le langage SQL	463
1.2.1 Recherche d'informations	463
1.2.2 Ajout d'informations	465
1.2.3 Mise à jour d'informations	466
1.2.4 Suppression d'informations	466
2. Présentation d'ADO.NET	467
2.1 Mode connecté	467
2.2 Mode non connecté	468
2.3 Architecture d'ADO.NET	469

2.4	Les fournisseurs de données	470
2.4.1	SQL Server	471
2.4.2	OLE DB	471
2.4.3	ODBC	472
2.4.4	Oracle	472
2.5	Rechercher les fournisseurs disponibles	472
2.6	Compatibilité du code	473
3.	Utilisation du mode connecté	475
3.1	Connexion à une base	476
3.1.1	Chaîne de connexion	477
3.1.2	Pool de connexions	479
3.1.3	Événements de connexion	480
3.2	Exécution d'une commande	481
3.2.1	Création d'une commande	481
3.2.2	Lecture d'informations	482
3.2.3	Modification des informations	484
3.2.4	Utilisation de paramètres	485
3.2.5	Exécution de procédure stockée	489
4.	Utilisation du mode non connecté	491
4.1	Remplir un DataSet à partir d'une base de données	492
4.1.1	Utilisation d'un DataAdapter	492
4.1.2	Ajout de contraintes existantes à un DataSet	495
4.2	Configurer un DataSet sans base de données	496
4.3	Manipuler les données dans un DataSet	498
4.3.1	Lecture des données	498
4.3.2	Création de contraintes sur une DataTable	499
4.3.3	Ajout de relations entre les DataTables	501
4.3.4	Parcourir les relations	502
4.3.5	État et versions d'une DataRow	503
4.3.6	Ajout de données	505
4.3.7	Modification de données	506
4.3.8	Suppression de données	508
4.3.9	Valider ou annuler les modifications	509

4.3.10	Filtrer et trier des données	509
4.3.11	Rechercher des données	513
4.4	Mettre à jour la base de données	515
4.4.1	Génération automatique de commandes	516
4.4.2	Utilisation de commandes personnalisées	521
4.4.3	Gestion des accès concurrents	521
4.5	Les transactions	526

Chapitre 9

Présentation de LINQ

1.	Présentation de LINQ	531
2.	Syntaxe du langage LINQ	532
2.1	Premières requêtes LINQ	534
2.2	Les opérateurs de requête	537
2.2.1	Tri de données	537
2.2.2	Opérations sur des ensembles de données	538
2.2.3	Filtrage de données	539
2.2.4	Projections	539
2.2.5	Partitionnement	540
2.2.6	Jointures et regroupements	542
2.2.7	Agrégations	543
3.	LINQ vers SQL	544
3.1	Le mappage objet relationnel	545
3.1.1	SQLMetal	545
3.1.2	Concepteur Objet/Relationnel	552
3.1.3	Utilisation de requêtes LINQ vers SQL	561
3.1.4	Mise à jour des données	564
3.1.5	Conflits des mises à jour	568

Chapitre 10
Utilisation de XML

1.	Présentation	571
2.	Structure d'un document XML	572
2.1	Constituants d'un document XML.....	573
2.2	Document bien formé et document valide.....	578
2.2.1	Document bien formé	578
2.2.2	Document valide	579
3.	Manipulation d'un document XML	579
3.1	Utilisation de DOM	582
3.2	Utilisation de XPath.....	585
3.2.1	Recherche dans un document XML	585
3.2.2	Modification des données d'un document XML	587
3.2.3	Ajout de nœud à un document XML	588

Chapitre 11
Déploiement de composants et d'applications

1.	Introduction	591
2.	Déploiement avec Windows Installer	592
2.1	Installation de InstallShield Limited Edition	593
2.2	Création d'un projet d'installation	597
2.2.1	Informations générales.....	598
2.2.2	Éléments pré-requis	599
2.2.3	Fichiers de l'application	600
2.2.4	Raccourcis vers l'application	602
2.2.5	Informations de la base de registre	603
2.2.6	Configuration des boîtes de dialogue	604

3.	Déploiement avec ClickOnce	605
3.1	Principe de fonctionnement de ClickOnce	607
3.2	Les différentes méthodes de déploiement	608
3.3	Les mises à jour de l'application	610
3.4	Mise en œuvre de la publication ClickOnce	612
	Index	627

Chapitre 4

Bases du langage

1. Les variables, constantes et énumérations

1.1 Les variables

Les variables vont vous permettre de mémoriser, pendant l'exécution de votre application, différentes valeurs utiles pour le fonctionnement de votre application. Une variable doit obligatoirement être déclarée avant son utilisation dans le code. Lors de la déclaration d'une variable, vous fixez ses caractéristiques.

1.1.1 Nom des variables

Voyons les règles à respecter pour nommer les variables :

- Le nom d'une variable commence obligatoirement par une lettre.
- Il peut être constitué de lettres, de chiffres ou du caractère souligné (_).
- Il peut contenir un maximum de 1023 caractères (pratiquement, il est préférable de se limiter à une taille plus raisonnable).
- Il y a distinction entre minuscules et majuscules (la variable `AgeDuCapitaine` est différente de la variable `ageducapitaine`).
- Les mots-clés du langage ne doivent pas être utilisés (c'est malgré tout possible mais dans ce cas, le nom de la variable doit être précédé du caractère @).

Par exemple, une variable nommée `if` sera utilisée dans le code sous cette forme `@if=56 ;)`.

1.1.2 Type des variables

En spécifiant un type pour une variable, nous indiquons quelles informations nous allons pouvoir stocker dans cette variable.

Deux catégories de types de variables sont disponibles :

- Les types valeur : la variable contient réellement les informations.
- Les types référence : la variable contient l'adresse mémoire où se trouvent les informations.

Les différents types de variables disponibles sont définis au niveau du Framework lui-même. Vous pouvez également utiliser les alias définis au niveau de Visual C#, peut-être plus explicites. Ainsi, le type **System.Int32** défini au niveau du framework peut être remplacé par le type **int** dans Visual C#.

Les différents types peuvent être classés en six catégories.

Les types numériques entiers

Types entiers signés			
sbyte	-128	127	8 bits
short	-32768	32767	16 bits
int	-2147483648	2147483647	32 bits
long	-9223372036854775808	9223372036854775807	64 bits

Types entiers non signés			
byte	0	255	8 bits
ushort	0	65535	16 bits
uint	0	4294967295	32 bits
ulong	0	18446744073709551615	64 bits

Lorsque vous choisissez un type pour vos variables entières, vous devez prendre en compte les valeurs minimale et maximale que vous envisagez de stocker dans la variable afin d'optimiser la mémoire utilisée par la variable. Il est, en effet, inutile d'utiliser un type Long pour une variable dont la valeur n'excédera pas 50, un type byte est dans ce cas suffisant.

■ Remarque

L'économie de mémoire semble dérisoire pour une variable unique mais devient appréciable lors de l'utilisation de tableaux de grande dimension.

Si, par contre, vous souhaitez optimiser la vitesse d'exécution de votre code, il est préférable d'utiliser le type int.

Les types décimaux

float	-3.40282347E+38	3.40282347E+38	4 octets
double	-1.7976931348623157E+308	1.7976931348623157E+308	8 octets
decimal	-79228162514264337593543950335	79228162514264337593543950335	16 octets

Les mêmes considérations d'optimisation que pour les variables entières doivent être prises en compte. Dans ce cas, une rapidité d'exécution maximale est obtenue avec le type double. Le type decimal est plus spécialement recommandé pour les calculs financiers pour lesquels les erreurs d'arrondis sont prohibées, mais au détriment de la rapidité d'exécution du code.

Les types caractères

Le type char (caractères) est utilisé pour stocker un caractère unique. Une variable de type char utilise deux octets pour stocker le code Unicode du caractère. Dans jeu de caractère Unicode, les 128 premiers caractères sont identiques au jeu de caractère ASCII, les caractères suivants jusqu'à 255 correspondent aux caractères spéciaux de l'alphabet latin (par exemple, les caractères accentués), le reste est utilisé pour des symboles ou pour les caractères d'autres alphabets.

L'affectation d'une valeur à une variable de type char doit être effectuée en encadrant la valeur par des caractères ''. Certains caractères ayant une signification spéciale pour le langage doivent être utilisés avec une séquence d'échappement. Cette séquence d'échappement commence toujours par le caractère \. Le tableau suivant résume les différentes séquences disponibles.

Séquence d'échappement	caractère
\'	Simple quote
\"	Double quote
\\\	Backslash
\0	Caractère nul
\a	Alerte
\b	Backspace
\f	Saut de page
\n	Saut de ligne
\r	Retour chariot
\t	Tabulation horizontale
\v	Tabulation verticale

Ces séquences d'échappement peuvent également être utilisées dans une chaîne de caractères. Chacune d'elles représente un caractère unique.

Pour pouvoir stocker des chaînes de caractères, il convient d'utiliser le type string, qui représente une suite de zéro à 2147483648 caractères. Les chaînes de caractères sont invariables car, lors de l'affectation d'une valeur à une chaîne de caractères, de l'espace est réservé en mémoire pour le stockage. Si, par la suite, cette variable reçoit une nouvelle valeur, le système lui assigne un nouvel emplacement en mémoire. Heureusement, ce mécanisme est transparent pour nous et la variable fera toujours automatiquement référence à la valeur qui lui a été assignée. Avec ce mécanisme, les chaînes de caractères peuvent avoir une taille variable. L'espace occupé en mémoire est automatiquement ajusté à la longueur de la chaîne de caractères.

Pour affecter une chaîne de caractères à une variable, le contenu de la chaîne doit être saisi entre " ", comme dans l'exemple ci-dessous :

Exemple

■ NomDuCapitaine = "Crochet" ;

Si des caractères spéciaux doivent apparaître dans une chaîne, ils doivent être spécifiés par une séquence d'échappement. Il existe cependant une autre possibilité qui permet parfois de rendre le code plus lisible. Cette solution consiste à faire précéder la chaîne de caractères du symbole @. Le compilateur considère alors que tous les caractères contenus entre les doubles quotes doivent être utilisés tels quel, y compris les éventuels retours chariot. La seule limitation concerne le caractère " qui, s'il doit faire partie de la chaîne, doit être doublé.

Les deux déclarations de chaînes suivantes sont identiques :

```
chaine = "Que dit il ?\ril dit \"bonjour\"";  
chaine = @"Que dit il ?  
il dit ""bonjour""";
```

Lorsqu'elles sont affichées sur la console, elles donnent le résultat suivant :



The screenshot shows a terminal window titled 'Sortie'. The command 'Afficher la sortie à partir de: Déboguer' is selected. The output window displays the following text:
Que dit il ?
il dit "bonjour"
The window has a standard Windows-style title bar and scroll bars. At the bottom, there are tabs labeled 'Sortie', 'Variables locales', and 'Espion 1', with 'Sortie' being the active tab.

■ Remarque

De nombreuses fonctions de la classe *string* permettent la manipulation des chaînes de caractères et seront détaillées plus loin dans ce chapitre.

Le type bool

Le type `bool` permet d'utiliser une variable qui peut prendre deux états vrai/faux, oui/non, on/off.

L'affectation se fait directement avec les valeurs `true` ou `false`, comme dans l'exemple suivant :

```
Disponible=true;  
Modifiable=false;
```

Le type Object

C'est peut-être le type le plus universel de Visual C#. Dans une variable de type `Object`, vous pouvez stocker n'importe quoi. En fait, ce type de variable ne stocke rien. La variable va contenir non pas la valeur elle-même, mais l'adresse, dans la mémoire de la machine, où l'on pourra trouver la valeur de la variable. Rassurez-vous, tout ce mécanisme est transparent et vous n'aurez jamais à manipuler les adresses mémoire directement.

Remarque

Une variable de type `Object` pourra donc faire référence à n'importe quel autre type de valeur y compris des types numériques simples. Cependant, le code sera moins rapide du fait de l'utilisation d'une référence.

Le type dynamic

Depuis sa première version, le langage C# est un langage statiquement typé. Chaque variable utilisée doit être déclarée avec un type défini. Cette contrainte permet au compilateur de vérifier que vous ne réalisez avec cette variable que des opérations compatibles avec son type. Ceci impose bien sûr de connaître le type de la variable au moment de la conception de l'application. Il arrive cependant parfois que le type de la variable ne soit connu qu'au moment de l'exécution de l'application. Il est, dans ce cas, possible d'utiliser le mot-clé **dynamic** comme type pour la variable concernée. Pour les variables déclarées avec ce type, le compilateur ne fait aucune vérification de compatibilité concernant les opérations exécutées avec cette variable. Ces opérations de vérification sont effectuées seulement au moment de l'exécution de l'application.