Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **RI10CSHAVSC** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Introduction 1

Chapitre 1
Introduction

1. Qu'est-ce que C# ? 9
   1.1 Que peut-on réaliser avec C# ? 10
   1.2 Le langage est-il stable et pérenne ? 12
2. Préparer son environnement 13
   2.1 Installation et configuration de Visual Studio Code 14
   2.2 Installer les outils de compilation 16
3. Comment fonctionne le C# ? 19

Chapitre 2
Premier programme

1. Créer sa première application C# 23
2. Comprendre et écrire du code C# 27
   2.1 Notions de variable et constante 29
      2.1.1 Types numériques 31
      2.1.2 Types textuels 33
      2.1.3 Valeur booléenne 35
      2.1.4 Opérateurs 35

2.2 Les autres types. 38
2.2.1 Stockage des dates 38
2.2.2 Les intervalles de temps 40
3. Analyser la structure d'un projet C# 41
3.1 La notion de blocs. 42
3.2 Signification des blocs de code. 44
3.2.1 Le bloc d'espace de noms 44
3.2.2 La définition d'une classe 47
3.2.3 La définition d'une méthode 48
3.3 Déclaration "top-level" 48
4. Exécuter un programme C# 49
4.1 Lancer le programme avec Visual Studio Code 49
4.2 Lancer depuis la ligne de commande. 51
5. Exercice 54
5.1 Énoncé. 54
5.2 Corrigé. 54

Chapitre 3
Programmation orientée objet

1. Principes de la programmation orientée objet 57
1.1 Qu'est-ce qu'une classe ? 57
1.1.1 Les classes dans Visual Studio Code 59
1.1.2 L'héritage. 60
1.1.3 L'encapsulation 61
1.2 Que peut-on déclarer dans une classe ?. 62
1.2.1 Les méthodes 62
1.2.2 Déclarer une donnée. 65
1.3 Instancier une classe. 69
1.3.1 Le constructeur. 69
1.3.2 L'instanciation avec le mot-clé new 71
1.4 Le polymorphisme 73

2. Concepts avancés 75
2.1 L'héritage avancé 75
2.1.1 Méthodes virtuelles 75
2.1.2 Classe abstraite 76
2.1.3 Interface 78
2.1.4 Implémentation par défaut dans une interface 80
2.1.5 Masquage 81
2.1.6 Interdire l'héritage 82
2.2 Les différents types d'objets 83
2.2.1 Les types références 83
2.2.2 Les types valeurs 84
2.2.3 Les types nullables 87
2.2.4 Les types références nullables 88
2.2.5 Les énumérations 90
2.2.6 Les records 91
2.3 Les modificateurs de classe 93
2.3.1 La notion de static 94
2.3.2 La notion de classe partielle 95
3. Exercice 96
3.1 Énoncé 96
3.2 Corrigé 97

Chapitre 4
Algorithmique

1. Bases de l'algorithmique 101
1.1 La logique conditionnelle 101
1.1.1 Test simple : le if/else 102
1.1.2 Multiples tests avec l'instruction switch 108
1.1.3 Pattern matching 110
1.1.4 Exercice - énoncé 115
1.1.5 Exercice - corrigé 115

1.2 Les collections 116
1.2.1 L'interface IEnumerable 116
1.2.2 Les tableaux 117
1.2.3 La liste 119
1.2.4 Les dictionnaires 122
1.2.5 Les collections algorithmiques 124
1.3 Les boucles 126
1.3.1 Les généralités sur les boucles 126
1.3.2 La boucle for 127
1.3.3 La boucle while 129
1.3.4 La boucle do while 129
1.3.5 La boucle foreach 130
1.3.6 Le mot-clé yield 130
1.3.7 Exercice - énoncé 131
1.3.8 Exercice - corrigé 132
2. Gestion des erreurs 135
2.1 Concept d'une exception 135
2.2 Renvoyer une exception 136
2.3 Gérer une exception 139
2.3.1 Blocs try, catch et finally 139
2.3.2 Filtre sur bloc catch 141
2.4 Exceptions et performances 143

Chapitre 5
LINQ

1. Fonctionnement de base 145
2. Variables anonymes 148
3. Principes des opérateurs LINQ 148
3.1 Opérateurs de production 152
3.2 Opérateurs de sélection 163
3.3 Opérateurs de génération 169

4. Expression de requête LINQ 169
4.1 Le mot-clé into 170
4.2 Le mot-clé let 172
5. Exercice 173
5.1 Énoncé 173
5.2 Corrigé 174

Chapitre 6
Sérialisation

1. Sérialisation en C# 175
2. Sérialisation binaire 176
2.1 Utilisation des attributs 177
2.2 Utilisation de l'interface ISerializable 181
3. Sérialisation XML 183
3.1 XmlSerializer 183
3.2 XDocument, XElement et XAttribute 186
4. Sérialisation JSON 190
4.1 Utf8JsonReader et Utf8JsonWriter 191
4.2 JsonDocument 194
4.3 JsonSerializer 195
5. Exercice 199
5.1 Énoncé 199
5.2 Corrigé 201

Chapitre 7
Concepts avancés

1. Asynchronisme 203
   1.1 Fonctionnement de base 203
   1.2 Thread et asynchronisme 205
   1.3 Asynchronisme en C# 206
   1.4 Les mots-clés async et await 207
   1.5 Flux asynchrones 210
2. Algorithmique avancée 212
   2.1 Programmation événementielle 212
      2.1.1 Les delegates 212
      2.1.2 Les events 214
   2.2 Les types génériques 217
      2.2.1 Utilisation standard 217
      2.2.2 Contraintes sur type générique 219
   2.3 Gestion de la mémoire 220
      2.3.1 Le destructeur 221
      2.3.2 IDisposable et IAsyncDisposable 222
   2.4 Paramètres de méthodes avancés 223
      2.4.1 Paramètre optionnel 223
      2.4.2 Mots-clés de paramètres 224
      2.4.3 Nommage de paramètres 227
      2.4.4 Paramètres variables 228
   2.5 Extension du fonctionnement d'un type 228
      2.5.1 Méthodes d'extension 229
      2.5.2 Définition des opérateurs 230
   2.6 Tuples et déconstruction 233
      2.6.1 Les tuples en C# 7 233
      2.6.2 Déconstruction de type 235
   2.7 Fonction locale 237

Chapitre 8
Créer des applications

1. Application web 241
   1.1 Applications web graphiques 241
      1.1.1 ASP.NET MVC 242
      1.1.2 ASP.NET Razor Pages 247
      1.1.3 Blazor 252
   1.2 API 256
2. Application de bureau 262
   2.1 WinForms 263
   2.2 Windows Presentation Foundation (WPF) 269
   2.3 Universal Windows Platform (UWP) 273
3. Application mobile 278
   3.1 Installation 279
      3.1.1 Installation à partir de la ligne de commande 279
      3.1.2 Installation avec Visual Studio 2022 283
   3.2 Code 288
4. Conclusion 290

Chapitre 9
Référence

1. Introduction 291
2. Mots-clés de type 291
3. Mots-clés de programmation orientée objet 293
4. Mots-clés algorithmiques 297

Index 303

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EI2ECR** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 1
Introduction

1. Avant-propos 9
2. Les risques de la suroptimisation 10
3. Le principe du profilage et du benchmarking 12
4. L'enjeu financier et écologique 15
5. Méthodologie 17
6. Environnement nécessaire 18

Chapitre 2
Principes du profilage

1. Une activité strictement régulée 19
2. Stabilité de la plate-forme 26
   - 2.1 Pourquoi cette règle ? 26
   - 2.2 Comment appliquer cette règle ? 26
   - 2.3 Extraire les cas particuliers 27
   - 2.4 Lien avec la maintenabilité 29
3. Neutralité de l'environnement 31
   - 3.1 Le principe 31
   - 3.2 Virtualisation 32
     - 3.2.1 Virtualisation lourde 32
     - 3.2.2 Virtualisation légère 33
   - 3.3 Déport d'affichage 34
   - 3.4 Effets de caches externes 35
   - 3.5 Processus externe 36

3.6 Services divers 37
3.7 Comparaison avec la nano-technologie 38
4. Objectifs fixés avant l'analyse 39
5. Améliorations mesurables 42
5.1 Pourquoi cette règle ? 42
5.2 Comment appliquer cette règle ? 42
6. Granularité descendante 44
6.1 Commençons par une parabole 44
6.2 Un exemple 44
6.3 Avertissement 45
6.4 Qui est responsable ? 47

Chapitre 3
Profilage d'une application .NET

1. Gestion de la mémoire par .NET 49
1.1 Principes de base 49
1.2 Gestion de mémoire automatisée et performances 50
1.3 Le cas particulier du temps réel 50
1.3.1 Lever un malentendu 50
1.3.2 Non-déterminisme des ramasse-miettes 52
1.4 Affectation de la mémoire 55
1.5 Comment .NET nous aide 61
1.6 Types valeurs et références 62
1.6.1 Fonctionnement d'une pile 63
1.6.2 Retour sur le code IL 69
1.6.3 Différence entre valeur et référence 71
1.6.4 Cas particulier des structures 76
1.6.5 Lien à la performance 78
1.7 Calcul de la taille mémoire à affecter 79
1.7.1 Cas du code 100 % géré 79
1.7.2 Cas des objets sujets à interopérabilité 80

1.8 Collecte de la mémoire 82
1.8.1 Critères de collecte de la mémoire 82
1.8.2 Mécanisme de compactage 84
1.8.3 Mémoire fixée et fragmentation 85
1.8.4 Déclenchement et exécution du ramasse-miettes 87
1.8.5 Notion de génération 88
1.8.6 Impact sur le codage pour la performance 90
1.8.7 Choix du ramasse-miettes 93
1.9 Boxing, unboxing et performances associées 95
1.9.1 Quel est le problème ? 95
1.9.2 Le boxing/unboxing et les performances 100
1.9.3 Le remède 102
1.10 Gestion de mémoire et finalisation 106
1.10.1 Laissons faire .NET ! 106
1.10.2 Relâcher des ressources externes lors du passage du GC 107
1.10.3 Fonctionnement du finaliseur 110
1.10.4 Relâcher les ressources au plus tôt 112
1.10.5 Combiner les deux opérations 115
1.11 Une dernière remarque 119
2. Particularité des fonctions inline 120
2.1 Mécanisme des fonctions inline 120
2.2 Problématiques de performance et fonctions inline 125
2.3 Impact sur les profileurs 129
3. Impact de la gestion mémoire sur la performance 132
3.1 Une grande diversité dans les impacts 132
3.2 Utilisation de la mémoire virtuelle 133
3.3 Fuites mémoire 137
3.4 Tas spécial pour les gros objets et fragmentation mémoire . . . 143
4. Les autres ressources à surveiller 146
4.1 La mémoire n'est pas tout 146
4.2 Le CPU 146
4.3 Les entrées/sorties 150

4.4 L'espace disque disponible 152
4.5 La bande passante 153

Chapitre 4
Application de test

1. Préambule 155
1.1 Une migration historique 155
2. Critères de choix 156
2.1 Pourquoi cette application ? 156
2.2 Utiliser des retours d'expérience 157
2.3 Le choix de la transparence 158
2.4 Les limites de la transparence 159
3. Application choisie 160
3.1 Domaine d'utilisation 160
3.2 Architecture 160
3.3 Interface 161
3.4 Description du métier 161
4. Détail de l'application 162
4.1 Trouver l'application 162
4.2 Installation de la base de données 162
4.2.1 Création manuelle 163
4.2.2 Utilisation des scripts de génération 164
4.3 Installation de l'application 165
4.3.1 Ouverture de la solution 165
4.3.2 Configuration des projets 165
4.3.3 Exécution de l'application 166
4.4 Détails des assemblages 167
4.5 Architecture du client 168
4.6 Structure des services web 168
4.7 Structure de la base de données 169
5. Explication de la lourdeur de l'application 169

6. Méthode recommandée 170

Chapitre 5
Présentation des outils

1. Choix des outils 173
2. Visual Studio 2022 174
   2.1 Fenêtre de diagnostic 175
   2.2 Session de profilage. 177
3. Compteurs de performance 181
   3.1 Terminologique 181
   3.2 Windows 182
   3.3 Linux et macOS 186
4. BenchmarkdotNet 188
   4.1 Création du projet de benchmark 189
   4.2 Création d'un benchmark. 189
   4.3 Exécuter le benchmark 191
5. Outils alternatifs. 192

Chapitre 6
Profilage

1. Par où commencer ? 193
2. Scénarios de profilage 194
   2.1 Premier scénario : afficher le détail d'une personne 194
   2.2 Deuxième scénario : afficher et éditer un contrat 197
3. Lancement du profilage 198
   3.1 Profilage de l'API : premier scénario 199
      3.1.1 Exécution avec profiler 201
      3.1.2 Première optimisation 206
      3.1.3 Test de charge 209
      3.1.4 Seconde optimisation 215

3.2 Refactoring suite aux optimisations 218
3.2.1 Mélange des responsabilités 219
3.2.2 Suppression du static 220
3.2.3 Utilisation de l'injection de dépendances 221
3.2.4 Profilage de l'API : premier scénario, après refactoring . 227
3.3 Profilage de l'API : deuxième scénario 229
3.3.1 Analyse de la consommation mémoire 229
3.3.2 Première optimisation 235
3.3.3 Benchmark du dézippage du fichier contrat 239
3.3.4 Seconde optimisation 242
3.3.5 Mise en place du pooling 246
4. Conclusion 250

Chapitre 7
Au-delà du profilage

1. Introduction 253
2. Pistes d'amélioration restantes 254
2.1 Introduction 254
2.2 Amélioration du ressenti 255
2.3 Temps de chargement de l'application 259
2.4 Asynchronisme des traitements 266
2.5 Marquer les changements d'application 269
2.6 Quelques dernières problématiques 269
2.6.1 Gestion correcte des traces 270
2.6.2 Duplication des requêtes SQL 270
2.6.3 Éviter la sur-architecture 272
2.6.4 Pagination des résultats 273
2.6.5 Le ramasse-miettes prend du temps 274
2.6.6 Limiter les exceptions 275
2.6.7 Fonctions Equals et GetHashCode 276
2.6.8 AddRange 277
2.6.9 Concaténation de chaînes 279

3. Tuning 282
3.1 Caveat 282
3.2 Compiler en release 283
3.3 Le curseur de la consistance 284
3.3.1 BASE au lieu d'ACID 284
3.3.2 Un second exemple 286
3.3.3 Passer le code de PROFI en BASE 287
3.4 Asynchronisme globalisé 292
3.5 Utiliser des références faibles 293
3.6 Attention au tuning extrême 294
3.6.1 Limites du tuning 294
3.6.2 Struct au lieu de class 294
3.6.3 Instanciation tardive et déréférencement précoce 296
3.6.4 Byte au lieu de int dans Enum ? 297
4. Aller plus loin en réarchitecturant 298
4.1 Problématique 298
4.2 Scalabilité 299
4.2.1 Concept 299
4.2.2 Modes de scalabilité 299
4.2.3 Parallélisation des traitements 300
4.2.4 Bonnes pratiques pour la scalabilité 301
4.2.5 Parallel Linq 301
4.3 Institutionnaliser le cache 304
4.4 Penser Lean/Agile 304
4.4.1 IMDB 304
4.4.2 NoSQL 304
4.4.3 CQRS 305
4.4.4 Prévalence 305

4.5 Performance des nouvelles architectures 306
4.5.1 Problématique 306
4.5.2 Scale Out 307
4.5.3 Parallélisation 307
4.5.4 Mobilité 307
4.5.5 SOA/EDA/ESB 308
4.6 Et pour aller encore plus loin 310

Conclusion

1. Tout peut poser problème 311
2. Checklist 312
3. Les causes des erreurs 314
4. Coder léger 316
5. Conclusion 317

Index 319