INFORMATIQUE
TECHNIQUE

Expert

C# 9

Développez des applications Windows avec Visual Studio 2019

En téléchargement

</> des exemples de code

+ QUIZ

Version en ligne **OFFERTE !** pendant 1 an

Jérôme HUGON

eni

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EI9C19VIS** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
Travailler avec Visual Studio 2019

1. Introduction 17
2. L'interface de développement 18
   - 2.1 L'éditeur de texte 20
   - 2.2 Le concepteur de vues. 25
   - 2.3 Le débogueur intégré 26
   - 2.4 Le gestionnaire d'extensions 27
   - 2.5 NuGet 29
   - 2.6 Fenêtres personnalisées 31
3. La création de solutions 31
   - 3.1 Définir le point d'entrée 32
   - 3.2 La différence entre projets et solutions 34
   - 3.3 Configurer le projet 35
   - 3.4 La conversion de solutions 38
   - 3.5 Les projets partagés 38
   - 3.6 Les outils de refactorisation 39

Chapitre 2
L'architecture .NET

1. Introduction 43
2. CLR 44
3. Les bibliothèques de classes 45

4. Les types 47
4.1 Les types valeur. 48
4.2 Les types référence 49

Chapitre 3
Introduction au langage C#

1. La syntaxe 51
1.1 Les identifiants 51
1.2 Les mots-clés 51
1.3 La ponctuation 53
1.4 Les opérateurs 54
1.4.1 Les opérateurs de calcul 54
1.4.2 Les opérateurs d'assignation. 55
1.4.3 Les opérateurs de comparaison 55
1.5 La déclaration de variables 57
1.6 Les instructions de contrôle 58
1.6.1 Les instructions conditionnelles 58
1.6.2 Les instructions itératives 62
1.6.3 Les instructions de saut 65
1.7 Les commentaires 67
2. Les espaces de noms 70
2.1 Le mot-clé using 71
2.2 Le mot-clé alias 71
2.3 Les classes statiques 72
3. Les types de base. 72
3.1 Les types numériques 72
3.1.1 Les entiers 73
3.1.2 Les décimaux. 74
3.2 Les booléens 74
3.3 Les chaînes de caractères 74
3.4 Les types nullable 76
3.5 La conversion de types 78

3.5.1 La conversion implicite 78
3.5.2 La conversion explicite 79
4. Les constantes et les énumérations 79
4.1 Les constantes 79
4.2 Les énumérations 80
5. Les tableaux 83
6. Les collections 84
7. Les directives preprocessor 86

Chapitre 4
La création de types

1. Introduction 89
2. Les niveaux d'accès 90
3. Les structures 91
4. Les classes 92
4.1 Les champs 92
4.2 Les propriétés 93
4.3 Les méthodes 95
4.3.1 La surcharge 97
4.3.2 Les paramètres 97
4.3.3 Les tuples 104
4.3.4 Les méthodes partielles 104
4.4 Les constructeurs 105
4.5 Les destructeurs 106
4.6 Les classes et membres statiques 107
4.7 Les classes partielles 107
4.8 Le mot-clé this 109
4.9 Les indexeurs 110
4.10 La surcharge d'opérateurs 111
4.10.1 Les opérateurs arithmétiques 112
4.10.2 Les opérateurs de comparaison 114

5. Les records 116

Chapitre 5
L'héritage

1. L'héritage de classe 117
   1.1 Implémenter l'héritage 117
   1.2 Les membres virtuels 119
   1.3 Masquer les membres hérités 119
   1.4 Le mot-clé base 120
   1.5 Les classes et membres abstraits 121
   1.6 Les classes et les méthodes scellées 122
   1.7 Les constructeurs dérivés 123
   1.8 Le polymorphisme 125
2. Les interfaces 127
   2.1 L'implémentation d'interfaces 127
   2.2 Le polymorphisme d'interface 129
   2.3 L'héritage d'interfaces 131

Chapitre 6
Types génériques

1. Introduction 133
2. La création de types génériques 134
3. Les contraintes de type 136
4. Les interfaces génériques 137
   4.1 La variance dans les interfaces génériques 138
      4.1.1 La covariance 138
      4.1.2 La contravariance 140
   4.2 La création d'interfaces génériques variantes 141
   4.3 L'héritage d'interfaces génériques variantes 142
5. La création de méthodes génériques 142

6. Valeur par défaut générique 145
7. L'héritage de classe générique 145

Chapitre 7
Délégués, événements et expressions lambda

1. Les délégués 147
   1.1 Les paramètres de méthode 148
   1.2 Les méthodes cibles multiples 149
   1.3 Les délégués génériques 150
   1.4 La compatibilité des délégués 150
2. Les événements 152
3. Les expressions lambda 155
   3.1 L'utilisation des expressions lambda 156
   3.2 Les délégués génériques 157
   3.3 La capture de variable 158
   3.4 Les fonctions locales 161

Chapitre 8
Création de formulaires

1. Utiliser les formulaires 163
   1.1 Ajouter des formulaires au projet 163
   1.2 Modifier le formulaire de démarrage 166
   1.3 Les propriétés des formulaires 166
   1.4 Les méthodes des formulaires 169
   1.5 Les événements des formulaires 170
2. Utiliser les contrôles 171
   2.1 Les types de contrôles 171
   2.2 Ajouter des contrôles aux formulaires 173
   2.3 Les propriétés des contrôles 174
   2.4 Les menus 176
   2.5 Les conteneurs 179

2.6 L'ergonomie 180
2.7 Ajouter des contrôles à la boîte à outils 181

Chapitre 9
Implémentation de gestionnaires d'événements

1. Introduction 183
2. La création de gestionnaires d'événements 184
   2.1 La mécanique d'un événement 186
   2.2 L'ajout dynamique d'un gestionnaire d'événements 186
   2.3 La suppression dynamique d'un gestionnaire d'événements . . 187
3. Les gestionnaires d'événements avancés 188
   3.1 Un gestionnaire pour plusieurs événements 188
   3.2 Plusieurs gestionnaires pour un événement 189

Chapitre 10
Validation de la saisie

1. Introduction 191
2. La validation au niveau des champs 191
   2.1 Les propriétés de validation 191
   2.2 Les événements de validation 192
      2.2.1 KeyDown et KeyUp 192
      2.2.2 KeyPress 193
      2.2.3 Validating et Validated 193
3. La validation au niveau du formulaire 195
4. Les méthodes de retour à l'utilisateur 198
   4.1 MessageBox 199
   4.2 ErrorProvider 200

Chapitre 11
Création de contrôles utilisateurs

1. Introduction 203
2. Les contrôles personnalisés 204
3. L'héritage de contrôles 206
4. Les contrôles utilisateurs 208

Chapitre 12
Création d'applications UWP

1. Introduction 215
2. Principes 216
3. Les outils de développement 218
4. Le langage XAML 221
5. Une première application UWP 223
   5.1 Les bases d'un projet UWP 223
   5.2 Les contrôles et événements 225
   5.3 Les styles 227

Chapitre 13
Débogage

1. Les types d'erreur 231
   1.1 Les erreurs de syntaxe 231
   1.2 Les erreurs d'exécution 232
   1.3 Les erreurs de logique 234
2. Le débogueur 234
   2.1 Contrôler l'exécution 236
   2.2 Les points d'arrêt 237
      2.2.1 Les conditions d'arrêt 238
      2.2.2 Le nombre d'accès 239
      2.2.3 Le filtrage 240

2.2.4 Les actions 240
2.2.5 Exécuter l'exécution jusqu'ici 241
2.3 Les DataTips 241
2.4 Les PerfTips 242
2.5 Les attributs Caller 243
3. Les fenêtres 245
3.1 La fenêtre Sortie 246
3.2 La fenêtre Variables locales 246
3.3 La fenêtre Automatique 247
3.4 La fenêtre Espion 247
3.5 La fenêtre Exécution 248
3.6 Les autres fenêtres 249

Chapitre 14
Gestion des exceptions

1. La classe Exception 251
2. La création d'exceptions personnalisées 252
3. Le déclenchement des exceptions 253
4. L'interception et la gestion des exceptions 256

Chapitre 15
Monitoring

1. Le traçage 263
1.1 Les classes Debug et Trace 263
1.2 La collection d'écouteurs 266
1.2.1 La création d'écouteurs 266
1.2.2 La sauvegarde des traces 267
1.3 Les commutateurs de trace 269
1.3.1 Le fonctionnement des commutateurs de trace 269
1.3.2 La configuration des commutateurs de trace 270

2. Les journaux d'événements 271
2.1 L'interaction avec les journaux d'événements. 272
2.2 La gestion des journaux d'événements 273
2.3 L'écriture d'événements 274
3. Les compteurs de performance 275
3.1 La création de compteurs de performance 276
3.1.1 Depuis Visual Studio 276
3.1.2 Depuis le code 277
3.2 L'utilisation de compteurs de performance. 279
3.3 L'analyse de compteurs de performance 282

Chapitre 16
Tests unitaires

1. Introduction aux tests unitaires 285
1.1 La création du projet 285
1.2 Les classes de tests unitaires. 286
2. La mise en place d'une série de tests 288
2.1 La création de tests au projet 288
2.2 Le déroulement des tests 289

Chapitre 17
Création du modèle de données

1. Introduction 293
2. La création d'un modèle 294
3. La création d'entités 295
4. La génération de la base de données 301
5. La création d'entités à partir du code (Code First) 307

Chapitre 18
Présentation d'Entity Framework

1. Introduction 313
2. Le mappage 314
   2.1 La couche logique 314
   2.2 La couche conceptuelle 316
   2.3 La couche de mappage 319
3. Travailler avec les entités 320
   3.1 Les entités 321
   3.2 La classe DbContext 323
   3.3 Les relations 324
      3.3.1 Le concept de table par type 324
      3.3.2 Le concept de table par hiérarchie 324

Chapitre 19
Présentation de LINQ

1. Les requêtes LINQ 327
   1.1 La syntaxe 327
   1.2 Les méthodes d'extension 328
2. Les opérateurs de requêtes 330
   2.1 Filtrer 330
      2.1.1 Where 330
      2.1.2 OfType<TResult> 330
      2.1.3 SelectMany 331
      2.1.4 Skip et Take 331
   2.2 Ordonner 332
      2.2.1 OrderBy 332
      2.2.2 ThenBy 333
   2.3 Grouper 333
      2.3.1 GroupBy 333
      2.3.2 Join 334

2.4 Agréger 334
2.5 Convertir 335
3. Les requêtes parallèles 335
3.1 Partitionner une requête 336
3.2 Annuler une requête 337

Chapitre 20
LINQ to Entities

1. Introduction 339
2. Extraire les données 340
2.1 L'extraction simple 340
2.2 L'extraction conditionnelle 341
3. Ajouter, modifier et supprimer des données 342
3.1 Ajouter des données 342
3.2 Modifier des données 343
3.3 Supprimer des données 343
3.4 L'envoi des modifications 343

Chapitre 21
LINQ to SQL

1. La création de classes LINQ to SQL 345
2. L'objet DataContext 348
2.1 La méthode ExecuteQuery 349
2.2 Utiliser des transactions 349
2.3 Les autres membres de DataContext 350
3. Exécuter des requêtes avec LINQ 351
3.1 Les requêtes simples 351
3.2 Les requêtes filtrées 352
3.3 Les requêtes de jointure 352

4. Les procédures stockées 352
4.1 L'ajout de procédures stockées au modèle. 353
4.2 L'exécution de procédures stockées 354

Chapitre 22
LINQ to XML

1. Les objets XML 355
1.1 XDocument 355
1.2 XElement. 356
1.3 XNamespace 357
1.4 XAttribute. 358
1.5 XComment 358
2. Exécuter des requêtes avec LINQ 359
2.1 Les requêtes simples 359
2.2 Les requêtes filtrées 360
2.3 Les requêtes de jointure 360

Chapitre 23
Le système de fichiers

1. Les classes de gestion du système de fichiers 361
1.1 DriveInfo. 361
1.2 Directory et DirectoryInfo. 363
1.3 File et FileInfo 365
1.4 Path 368
2. Travailler avec le système de fichiers 371
2.1 Les objets Stream 371
2.2 La classe FileStream 371
2.3 Lire un fichier texte 373
2.3.1 Lire avec la classe File 373
2.3.2 Lire avec la classe StreamReader 374

2.4 Écrire dans un fichier texte 376
2.4.1 Écrire avec la classe File 376
2.4.2 Écrire avec la classe StreamWriter 377

Chapitre 24
Sérialisation

1. Introduction 379
2. La sérialisation binaire 380
2.1 Les bases 380
2.2 Contrôler la sérialisation 382
2.2.1 Le contrôle par attribut 382
2.2.2 Le contrôle par interface 384
3. La sérialisation XML 387
3.1 Les bases 388
3.2 Contrôler la sérialisation 391
3.3 La sérialisation XML SOAP 392

Chapitre 25
Expressions régulières

1. Introduction 395
2. Une première expression régulière 396
3. Les options de recherche 397
4. Les caractères d'échappement 398
5. Les ensembles 398
6. Les groupes 400
7. Les ancres 401
8. Les quantifieurs 402

Chapitre 26
Multithreading

1. Introduction 403
2. La classe Thread 404
   2.1 Créer un thread 404
   2.2 Suspendre ou annuler un thread 405
   2.3 Échanger des données avec un thread 406
   2.4 Verrouiller un thread 409
   2.5 Priorité des threads 410
3. Fonctions asynchrones 411
   3.1 Task et Task<TResult> 412
   3.2 async et await 414
4. Le composant BackgroundWorker 416

Chapitre 27
Globalisation et localisation

1. Introduction 419
2. La culture 420
3. La globalisation 422
4. La localisation 424

Chapitre 28
Sécurité

1. Introduction 429
2. Les éléments de base 430
   2.1 L'interface IPermission 430
   2.2 La classe CodeAccessPermission 431
   2.3 L'interface IPrincipal 431
3. Implémentation de la sécurité 433
   3.1 La sécurité basée sur les rôles 433

3.1.1 Sécurité impérative 433
3.1.2 Sécurité déclarative 435
3.2 La sécurité basée sur les droits d'accès 435
3.2.1 Sécurité impérative 436
3.2.2 Sécurité déclarative 437
4. Introduction à la cryptographie 438

Chapitre 29
Pour aller plus loin

1. Le dessin avec GDI+ 441
1.1 La classe Graphics 442
1.1.1 Les coordonnées 442
1.1.2 Les formes 443
1.2 La structure Color et les classes Brush et Pen 445
1.2.1 La structure Color 445
1.2.2 La classe Brush 446
1.2.3 La classe Pen 446
1.2.4 Les paramètres système 447
1.3 Les exemples 447
1.3.1 L'affichage de texte 447
1.3.2 Redimensionner une image 449
2. Le remoting 450
2.1 Le principe 450
2.2 L'implémentation 451
2.2.1 La couche commune 451
2.2.2 L'application serveur 452
2.2.3 L'application cliente 454
3. Reflection 458
3.1 La classe System.Type 458
3.2 Charger un assemblage dynamiquement 460
3.2.1 L'énumération des types 460
3.2.2 L'instanciation d'objets 461

3.2.3 L'utilisation des membres. 462

Chapitre 30
Assemblages et configurations

1. Introduction 465
2. Les assemblages privés 465
3. Les assemblages partagés 468
4. Les fichiers de configuration 470

Chapitre 31
Déploiement

1. Introduction 473
2. Les projets de déploiement. 474
   2.1 XCOPY 474
   2.2 Projet CAB 475
   2.3 Projet de module de fusion. 476
   2.4 Projet d'installation 476
3. L'assistant Installation 477
4. Configuration du projet 481
   4.1 Les propriétés du projet 481
   4.2 Les éditeurs de configuration. 484
      4.2.1 Éditeur du système de fichiers. 485
      4.2.2 Éditeur du registre 486
      4.2.3 Éditeur des types de fichiers 487
      4.2.4 Éditeur de l'interface utilisateur. 489
      4.2.5 Éditeur des actions personnalisées. 491
      4.2.6 Éditeur des conditions de lancement 492

Index 495

Chapitre 11
Création de contrôles utilisateurs

1. Introduction

Le développement d'applications est principalement basé sur les contrôles ; ils fournissent des fonctionnalités distinctes sous une forme visuelle permettant à l'utilisateur d'interagir avec eux. Tous ces contrôles dérivent à un niveau plus ou moins lointain de la classe de base `System.Windows.Forms.Control`. Visual Studio propose l'intégration de contrôles tiers par l'ajout à la boîte à outils. Mais si le besoin est très spécifique, il est possible de créer ses propres contrôles.

La classe de base des contrôles, `Control`, fournit les fonctionnalités de base qui sont nécessaires, notamment pour les entrées utilisateurs via le clavier et la souris. Cela implique donc des propriétés, des méthodes et des événements communs à tous les contrôles. Néanmoins, cette classe de base ne fournit pas la logique d'affichage du contrôle.

Il existe trois modes de création de contrôle :

– Les contrôles personnalisés.
– L'héritage de contrôles.
– Les contrôles utilisateurs.

La création de contrôles s'inscrit dans le principe de réutilisation du code. La logique est créée en un seul endroit et peut être utilisée plusieurs fois. L'avantage est d'autant plus important pour la maintenance d'application car pour changer le comportement de ce contrôle, il n'y aura qu'un fichier à modifier.

2. Les contrôles personnalisés

Ces contrôles offrent les plus grandes possibilités de personnalisation tant au niveau graphique que logique. Un contrôle personnalisé hérite directement de la classe `Control`. Il est donc nécessaire d'écrire toute la logique d'affichage ce qui, suivant le résultat attendu, peut être une phase très longue et compliquée. Les méthodes, propriétés et événements doivent également être définis par le développeur.

La classe de base `Control` expose l'événement `Paint`. C'est celui-ci qui est levé lorsque le contrôle est généré et cela implique l'exécution du gestionnaire de l'événement par défaut `OnPaint`. Cette méthode reçoit un paramètre unique du type `PaintEventArgs` contenant les informations requises sur la surface de dessin du contrôle. Le type `PaintEventArgs` possède deux propriétés, `Graphics` du type `System.Drawing.Graphics` et `ClipRectangle` du type `System.Drawing.Rectangle`. Pour ajouter la logique de dessin au contrôle, il faut surcharger la méthode `OnPaint` et y ajouter le code de dessin :

```
protected override void OnPaint
                        (System.Windows.Forms.PaintEventArgs e)
{
    // Code de dessin du contrôle
}
```

La propriété `Graphics` de l'objet `PaintEventArgs` représente la surface du contrôle tandis que la propriété `ClipRectangle` représente la zone devant être dessinée. Lors de la première représentation du contrôle, la propriété `ClipRectangle` représente les limites du contrôle. Ces limites peuvent ensuite être modifiées, par exemple si un contrôle au-dessus en cache une partie de telle sorte que le contrôle ait besoin d'être redessiné. La partie `ClipRectangle` représentera la région à modifier.

Créez un dossier **Controls** à la racine du projet et ajoutez une nouvelle classe nommée **CustomControl** définie de la manière suivante :

```
using System.Drawing;

namespace SelfMailer.Controls
{
    public class CustomControl : System.Windows.Forms.Control
    {
        protected override void OnPaint
                        (System.Windows.Forms.PaintEventArgs e)
        {
            Rectangle R = new Rectangle(0, 0,
                            this.Size.Width, this.Size.Height);
            e.Graphics.FillRectangle(Brushes.Green, R);
        }
    }
}
```

Dans le constructeur du formulaire **MailServerSettings**, ajoutez le code d'instanciation du contrôle personnalisé :

```
Controls.CustomControl C = new Controls.CustomControl();
C.Location = new System.Drawing.Point(0, 0);
C.Size = this.Size;
this.Controls.Add(C);
```

Ce code instancie un nouveau contrôle du type `CustomControl`, lui affecte la position en haut à gauche et définit sa taille à celle du formulaire. Pour finir, le contrôle est ajouté à la collection des contrôles du formulaire.

Lancez l'application ([F5]) et ouvrez le formulaire des paramètres de serveur mail pour voir que le contrôle, qui représente un simple rectangle vert, remplit le formulaire comme une couleur de fond.

Remarque

Les possibilités de dessin avec GDI+ seront abordées plus loin dans cet ouvrage, à la section Le dessin avec GDI+ du chapitre Pour aller plus loin.

Il suffit ensuite d'ajouter les membres requis pour la logique du contrôle afin de le finaliser.

3. L'héritage de contrôles

Si le but est d'étendre les fonctionnalités d'un contrôle existant, que ce soit un contrôle du Framework .NET ou d'un éditeur tiers, la manière la plus rapide est d'hériter de ce contrôle. Le nouveau contrôle possède ainsi tous les membres et la représentation visuelle de sa classe parente. Il n'y a plus qu'à rajouter la logique de traitement. Au même titre que les contrôles personnalisés, il reste possible de surcharger la méthode `OnPaint` pour modifier l'aspect visuel du contrôle.

Si une application comporte plusieurs formulaires qui requièrent un e-mail comme champ de saisie, il serait préférable de créer un contrôle héritant de la classe `TextBox` et d'y implémenter la logique de validation puis d'ajouter ce contrôle aux formulaires de manière à ne pas répéter le code de validation dans chacun d'eux.

La création d'un contrôle hérité se fait de la même manière qu'un contrôle personnalisé, en créant une classe qui va hériter du contrôle ayant le comportement de base souhaité. Créez la classe `EmailTextBox` dans le dossier **Controls** et faites-la hériter de la classe `TextBox` :

```
public class EmailTextBox : System.Windows.Forms.TextBox
{
}
```

Ajoutez une surcharge de la méthode `OnValidating` pour effectuer les vérifications sur le format et ajoutez le code de la méthode `FromEmail_Validating` du formulaire **MailServerSettings** :

```
protected override void
          OnValidating(System.ComponentModel.CancelEventArgs e)
{
    base.OnValidating(e);
    string pattern = @"^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\." +
                     @"[0-9]{1,3}\.[0-9]{1,3}\.)|" +
                     @"(([a-zA-Z0-9\-]+\.)+))" +
                     @"([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";
    Regex reg = new Regex(pattern);
    if (!reg.IsMatch(this.Text))
    {
        this.BackColor = Color.Bisque;
        e.Cancel = true;
```

```
    }
    else
        this.BackColor = this.PreviousBackColor;
}
```

Des modifications sont à apporter car on n'accède plus à la propriété `Text` du contrôle à partir du formulaire. Il faut donc remplacer :

```
this.FromEmail.Text
```

par un accès direct :

```
this.Text
```

La première instruction :

```
base.OnValidating(e);
```

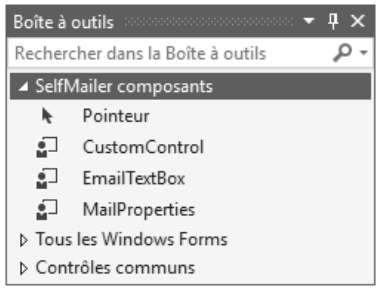
permet d'appeler la méthode de validation de la classe de base. Ainsi, il y a une première validation par la classe de base puis il y a validation du contrôle par le code supplémentaire.

La dernière chose notable est que le composant **ErrorProvider** n'est plus au même niveau. Pour signaler à l'utilisateur que le champ est invalide, au lieu d'afficher une icône, la couleur de fond du contrôle est modifiée. La propriété `PreviousBackColor`, initialisée avec la couleur de fond de base dans le constructeur, permet de la conserver afin de la réaffecter au contrôle en cas de succès de la validation :

```
protected Color PreviousBackColor { get; set; }

public EmailTextBox()
{
    this.PreviousBackColor = this.BackColor;
}
```

Le gestionnaire d'événements `FromEmail_Validating` est devenu inutile puisque la validation se fait au sein du contrôle. De plus, vous pouvez supprimer le contrôle de type `TextBox` pour la saisie de l'e-mail de l'expéditeur et le remplacer par un contrôle de type `EmailTextBox` qui a été ajouté par Visual Studio dans la boîte à outils sous le groupe **SelfMailer composants** (où **SelfMailer** représente le nom du projet).

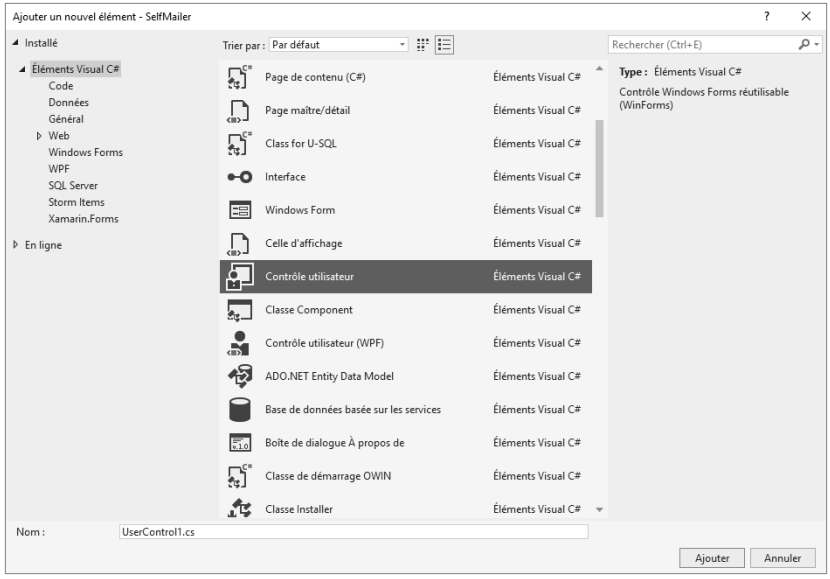


Lancez l'application ([F5]) pour tester la fonctionnalité.

4. Les contrôles utilisateurs

Le but d'un contrôle utilisateur est de regrouper de manière logique des contrôles afin d'obtenir une entité réutilisable. La création se fait par l'ajout au projet d'un **Contrôle utilisateur** depuis la fenêtre d'ajout d'un nouvel élément.

Ajoutez un contrôle utilisateur nommé **MailProperties** dans le dossier **Controls** du projet :



Les contrôles utilisateurs peuvent être construits avec le concepteur de vue au même titre que les formulaires. Ils agissent à la manière d'un conteneur. Il suffit donc de faire glisser les contrôles depuis la boîte à outils sur le concepteur de vue pour ajouter des contrôles.

Ajoutez les contrôles suivants au contrôle utilisateur **MailProperties** :

Type	Propriété	Valeur
`Label`	`Name`	lblSendType
	`Text`	Type d'envoi :
`ComboBox`	`Name`	SendType
	`DropDownStyle`	DropDownList
`GroupBox`	`Name`	MailContent
	`Anchor`	Top, Left, Right
	`Text`	Contenu
	`Enabled`	False
`Label`	`Name`	lblSubject
	`Text`	Sujet :
`TextBox`	`Name`	Subject
	`Anchor`	Top, Left, Right
`Label`	`Name`	lblBody
	`Text`	Corps :
`Button`	`Name`	LoadBody
	`Text`	Parcourir ...
`Button`	`Name`	PreviewBody
	`Text`	Aperçu