




EXPERT
EXPO
EX

C# 8

Développez
des applications Windows
avec **Visual Studio 2019**

En téléchargement

 des exemples de code

 + QUIZ

Version en ligne
OFFERTE !
pendant 1 an

Jérôme HUGON



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EI8C19VIS** 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 33
 - 3.3 Configurer le projet 34
 - 3.4 La conversion de solutions 37
 - 3.5 Les projets partagés 37
 - 3.6 Les outils de refactorisation 38

Chapitre 2 L'architecture .NET

- 1. Introduction 41
- 2. CLR 42
- 3. Les bibliothèques de classes 42

4. Les types	45
4.1 Les types valeur.	46
4.2 Les types référence	47

Chapitre 3

Introduction au langage C#

1. La syntaxe	49
1.1 Les identifiants	49
1.2 Les mots-clés	49
1.3 La ponctuation	51
1.4 Les opérateurs	52
1.4.1 Les opérateurs de calcul.	52
1.4.2 Les opérateurs d'assignation	53
1.4.3 Les opérateurs de comparaison	53
1.5 La déclaration de variables	54
1.6 Les instructions de contrôle	55
1.6.1 Les instructions conditionnelles	55
1.6.2 Les instructions itératives	60
1.6.3 Les instructions de saut.	62
1.7 Les commentaires	64
2. Les espaces de noms	67
2.1 Le mot-clé using	68
2.2 Le mot-clé alias	68
2.3 Les classes statiques	69
3. Les types de base.	69
3.1 Les types numériques.	69
3.1.1 Les entiers.	70
3.1.2 Les décimaux	71
3.2 Les booléens	71
3.3 Les chaînes de caractères	71
3.4 Les types nullable	73

- 3.5 La conversion de types75
 - 3.5.1 La conversion implicite75
 - 3.5.2 La conversion explicite76
- 4. Les constantes et les énumérations76
 - 4.1 Les constantes76
 - 4.2 Les énumérations77
- 5. Les tableaux80
- 6. Les collections81
- 7. Les directives preprocessor83

Chapitre 4
La création de types

- 1. Introduction87
- 2. Les niveaux d'accès88
- 3. Les structures89
- 4. Les classes90
 - 4.1 Les champs90
 - 4.2 Les propriétés91
 - 4.3 Les méthodes93
 - 4.3.1 La surcharge95
 - 4.3.2 Les paramètres95
 - 4.3.3 Les tuples102
 - 4.4 Les constructeurs102
 - 4.5 Les destructeurs104
 - 4.6 Les classes et membres statiques104
 - 4.7 Les classes partielles105
 - 4.8 Le mot-clé this106
 - 4.9 Les indexeurs108
 - 4.10 La surcharge d'opérateurs109
 - 4.10.1 Les opérateurs arithmétiques110
 - 4.10.2 Les opérateurs de comparaison112

Chapitre 5

L'héritage

1. L'héritage de classe	115
1.1 Implémenter l'héritage	115
1.2 Les membres virtuels	117
1.3 Masquer les membres hérités.	117
1.4 Le mot-clé base	118
1.5 Les classes et membres abstraits	119
1.6 Les classes et les méthodes scellées	120
1.7 Les constructeurs dérivés	121
1.8 Le polymorphisme	123
2. Les interfaces.	125
2.1 L'implémentation d'interfaces	125
2.2 Le polymorphisme d'interface	127
2.3 L'héritage d'interfaces.	129

Chapitre 6

Types génériques

1. Introduction	131
2. La création de types génériques.	132
3. Les contraintes de type.	134
4. Les interfaces génériques	135
4.1 La variance dans les interfaces génériques	136
4.1.1 La covariance	136
4.1.2 La contravariance.	137
4.2 La création d'interfaces génériques variantes	138
4.3 L'héritage d'interfaces génériques variantes	139
5. La création de méthodes génériques	140
6. Valeur par défaut générique.	142
7. L'héritage de classe générique.	143

Chapitre 7
Délégués, événements et expressions lambda

- 1. Les délégués 145
 - 1.1 Les paramètres de méthode 146
 - 1.2 Les méthodes cibles multiples 147
 - 1.3 Les délégués génériques 148
 - 1.4 La compatibilité des délégués 148
- 2. Les événements 150
- 3. Les expressions lambda 153
 - 3.1 L'utilisation des expressions lambda 154
 - 3.2 Les délégués génériques 155
 - 3.3 La capture de variable 155
 - 3.4 Les fonctions locales 158

Chapitre 8
Création de formulaires

- 1. Utiliser les formulaires 159
 - 1.1 Ajouter des formulaires au projet 159
 - 1.2 Modifier le formulaire de démarrage 162
 - 1.3 Les propriétés des formulaires 162
 - 1.4 Les méthodes des formulaires 165
 - 1.5 Les événements des formulaires 166
- 2. Utiliser les contrôles 167
 - 2.1 Les types de contrôles 167
 - 2.2 Ajouter des contrôles aux formulaires 168
 - 2.3 Les propriétés des contrôles 170
 - 2.4 Les menus 171
 - 2.5 Les conteneurs 174
 - 2.6 L'ergonomie 175
 - 2.7 Ajouter des contrôles à la boîte à outils 176

Chapitre 9**Implémentation de gestionnaires d'événements**

1. Introduction 179
2. La création de gestionnaires d'événements 180
 - 2.1 La mécanique d'un événement. 181
 - 2.2 L'ajout dynamique d'un gestionnaire d'événements 182
 - 2.3 La suppression dynamique d'un gestionnaire d'événements . . 183
3. Les gestionnaires d'événements avancés 183
 - 3.1 Un gestionnaire pour plusieurs événements. 183
 - 3.2 Plusieurs gestionnaires pour un événement 184

Chapitre 10**Validation de la saisie**

1. Introduction 187
2. La validation au niveau des champs 187
 - 2.1 Les propriétés de validation 187
 - 2.2 Les événements de validation 188
 - 2.2.1 KeyDown et KeyUp 188
 - 2.2.2 KeyPress 189
 - 2.2.3 Validating et Validated 189
3. La validation au niveau du formulaire 191
4. Les méthodes de retour à l'utilisateur 194
 - 4.1 MessageBox. 195
 - 4.2 ErrorProvider. 196

Chapitre 11**Création de contrôles utilisateurs**

1. Introduction 199
2. Les contrôles personnalisés 200
3. L'héritage de contrôles 202

4. Les contrôles utilisateurs 204

Chapitre 12
Création d'applications UWP

1. Introduction 211
2. Principes 212
3. Les outils de développement 214
4. Le langage XAML 216
5. Une première application UWP 219
 5.1 Les bases d'un projet UWP 219
 5.2 Les contrôles et événements 221
 5.3 Les styles 222

Chapitre 13
Débogage

1. Les types d'erreur 227
 1.1 Les erreurs de syntaxe 227
 1.2 Les erreurs d'exécution 228
 1.3 Les erreurs de logique 230
2. Le débogueur 230
 2.1 Contrôler l'exécution 232
 2.2 Les points d'arrêt 233
 2.2.1 Les conditions d'arrêt 234
 2.2.2 Le nombre d'accès 235
 2.2.3 Le filtrage 236
 2.2.4 Les actions 236
 2.2.5 Exécuter l'exécution jusqu'ici 237
 2.3 Les DataTips 237
 2.4 Les PerfTips 238
 2.5 Les attributs Caller 239

3. Les fenêtres	241
3.1 La fenêtre Sortie	242
3.2 La fenêtre Variables locales	242
3.3 La fenêtre Automatique	243
3.4 La fenêtre Espion	243
3.5 La fenêtre Exécution	243
3.6 Les autres fenêtres	244

Chapitre 14

Gestion des exceptions

1. La classe Exception	247
2. La création d'exceptions personnalisées	248
3. Le déclenchement des exceptions	249
4. L'interception et la gestion des exceptions	252

Chapitre 15

Monitoring

1. Le traçage	259
1.1 Les classes Debug et Trace	259
1.2 La collection d'écouteurs	262
1.2.1 La création d'écouteurs	262
1.2.2 La sauvegarde des traces	263
1.3 Les commutateurs de trace	265
1.3.1 Le fonctionnement des commutateurs de trace	265
1.3.2 La configuration des commutateurs de trace	266
2. Les journaux d'événements	267
2.1 L'interaction avec les journaux d'événements	268
2.2 La gestion des journaux d'événements	269
2.3 L'écriture d'événements	270

- 3. Les compteurs de performance 271
 - 3.1 La création de compteurs de performance 272
 - 3.1.1 Depuis Visual Studio..... 272
 - 3.1.2 Depuis le code 273
 - 3.2 L'utilisation de compteurs de performance..... 275
 - 3.3 L'analyse de compteurs de performance 278

Chapitre 16
Tests unitaires

- 1. Introduction aux tests unitaires 281
 - 1.1 La création du projet 281
 - 1.2 Les classes de tests unitaires..... 282
- 2. La mise en place d'une série de tests 284
 - 2.1 La création de tests au projet..... 284
 - 2.2 Le déroulement des tests 285

Chapitre 17
Création du modèle de données

- 1. Introduction 289
- 2. La création d'un modèle 290
- 3. La création d'entités 291
- 4. La génération de la base de données 297
- 5. La création d'entités à partir du code (Code First) 302

Chapitre 18
Présentation d'Entity Framework

- 1. Introduction 307
- 2. Le mappage 308
 - 2.1 La couche logique 308
 - 2.2 La couche conceptuelle..... 310

2.3	La couche de mappage	313
3.	Travailler avec les entités	314
3.1	Les entités	315
3.2	La classe DbContext	317
3.3	Les relations	318
3.3.1	Le concept de table par type	318
3.3.2	Le concept de table par hiérarchie	318

Chapitre 19

Présentation de LINQ

1.	Les requêtes LINQ	321
1.1	La syntaxe	321
1.2	Les méthodes d'extension	322
2.	Les opérateurs de requêtes	324
2.1	Filtrer	324
2.1.1	Where	324
2.1.2	OfType<TResult>	324
2.1.3	SelectMany	325
2.1.4	Skip et Take	325
2.2	Ordonner	326
2.2.1	OrderBy	326
2.2.2	ThenBy	327
2.3	Grouper	327
2.3.1	GroupBy	327
2.3.2	Join	328
2.4	Agréger	328
2.5	Convertir	329
3.	Les requêtes parallèles	329
3.1	Partitionner une requête	330
3.2	Annuler une requête	331

Chapitre 20
LINQ to Entities

- 1. Introduction 333
- 2. Extraire les données 334
 - 2.1 L'extraction simple 334
 - 2.2 L'extraction conditionnelle 335
- 3. Ajouter, modifier et supprimer des données 336
 - 3.1 Ajouter des données 336
 - 3.2 Modifier des données 337
 - 3.3 Supprimer des données 337
 - 3.4 L'envoi des modifications 337

Chapitre 21
LINQ to SQL

- 1. La création de classes LINQ to SQL 339
- 2. L'objet DataContext 342
 - 2.1 La méthode ExecuteQuery 343
 - 2.2 Utiliser des transactions 343
 - 2.3 Les autres membres de DataContext 344
- 3. Exécuter des requêtes avec LINQ 345
 - 3.1 Les requêtes simples 345
 - 3.2 Les requêtes filtrées 346
 - 3.3 Les requêtes de jointure 346
- 4. Les procédures stockées 346
 - 4.1 L'ajout de procédures stockées au modèle 347
 - 4.2 L'exécution de procédures stockées 348

Chapitre 22

LINQ to XML

1. Les objets XML	349
1.1 XDocument	349
1.2 XElement	350
1.3 XNamespace	351
1.4 XAttribute	352
1.5 XComment	352
2. Exécuter des requêtes avec LINQ	353
2.1 Les requêtes simples	353
2.2 Les requêtes filtrées	354
2.3 Les requêtes de jointure	354

Chapitre 23

Le système de fichiers

1. Les classes de gestion du système de fichiers	355
1.1 DriveInfo	355
1.2 Directory et DirectoryInfo	357
1.3 File et FileInfo	359
1.4 Path	362
2. Travailler avec le système de fichiers	365
2.1 Les objets Stream	365
2.2 La classe FileStream	365
2.3 Lire un fichier texte	367
2.3.1 Lire avec la classe File	367
2.3.2 Lire avec la classe StreamReader	368
2.4 Écrire dans un fichier texte	370
2.4.1 Écrire avec la classe File	370
2.4.2 Écrire avec la classe StreamWriter	371

Chapitre 24
Sérialisation

- 1. Introduction 373
- 2. La sérialisation binaire 374
 - 2.1 Les bases 374
 - 2.2 Contrôler la sérialisation 376
 - 2.2.1 Le contrôle par attribut 376
 - 2.2.2 Le contrôle par interface 378
- 3. La sérialisation XML 381
 - 3.1 Les bases 382
 - 3.2 Contrôler la sérialisation 385
 - 3.3 La sérialisation XML SOAP 386

Chapitre 25
Expressions régulières

- 1. Introduction 389
- 2. Une première expression régulière 390
- 3. Les options de recherche 391
- 4. Les caractères d'échappement 392
- 5. Les ensembles 392
- 6. Les groupes 394
- 7. Les ancrs 395
- 8. Les quantifieurs 396

Chapitre 26
Multithreading

- 1. Introduction 397
- 2. La classe Thread 398
 - 2.1 Créer un thread 398
 - 2.2 Suspendre ou annuler un thread 399

2.3	Échanger des données avec un thread	400
2.4	Verrouiller un thread	403
2.5	Priorité des threads	404
3.	Fonctions asynchrones	405
3.1	Task et Task<TResult>	406
3.2	async et await	408
4.	Le composant BackgroundWorker	410

Chapitre 27

Globalisation et localisation

1.	Introduction	413
2.	La culture	413
3.	La globalisation	416
4.	La localisation	418

Chapitre 28

Sécurité

1.	Introduction	421
2.	Les éléments de base	422
2.1	L'interface IPermission	422
2.2	La classe CodeAccessPermission	422
2.3	L'interface IPrincipal	423
3.	Implémentation de la sécurité	424
3.1	La sécurité basée sur les rôles	424
3.1.1	Sécurité impérative	425
3.1.2	Sécurité déclarative	426
3.2	La sécurité basée sur les droits d'accès	427
3.2.1	Sécurité impérative	427
3.2.2	Sécurité déclarative	428
4.	Introduction à la cryptographie	429

Chapitre 29
Pour aller plus loin

- 1. Le dessin avec GDI+ 433
 - 1.1 La classe Graphics 434
 - 1.1.1 Les coordonnées 434
 - 1.1.2 Les formes 435
 - 1.2 La structure Color et les classes Brush et Pen 437
 - 1.2.1 La structure Color 437
 - 1.2.2 La classe Brush 438
 - 1.2.3 La classe Pen 438
 - 1.2.4 Les paramètres système 439
 - 1.3 Les exemples 439
 - 1.3.1 L'affichage de texte 439
 - 1.3.2 Redimensionner une image 441
- 2. Le remoting 442
 - 2.1 Le principe 442
 - 2.2 L'implémentation 443
 - 2.2.1 La couche commune 443
 - 2.2.2 L'application serveur 444
 - 2.2.3 L'application cliente 446
- 3. Reflection 450
 - 3.1 La classe System.Type 450
 - 3.2 Charger un assemblage dynamiquement 452
 - 3.2.1 L'énumération des types 452
 - 3.2.2 L'instanciation d'objets 453
 - 3.2.3 L'utilisation des membres 454

Chapitre 30**Assemblages et configurations**

1. Introduction	457
2. Les assemblages privés	457
3. Les assemblages partagés	460
4. Les fichiers de configuration	462

Chapitre 31**Déploiement**

1. Introduction	465
2. Les projets de déploiement	466
2.1 XCOPY	466
2.2 Projet CAB	467
2.3 Projet de module de fusion	468
2.4 Projet d'installation	468
3. L'assistant Installation	469
4. Configuration du projet	473
4.1 Les propriétés du projet	473
4.2 Les éditeurs de configuration	476
4.2.1 Éditeur du système de fichiers	477
4.2.2 Éditeur du registre	478
4.2.3 Éditeur des types de fichiers	479
4.2.4 Éditeur de l'interface utilisateur	481
4.2.5 Éditeur des actions personnalisées	483
4.2.6 Éditeur des conditions de lancement	484

Index	487
-----------------	-----

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 :

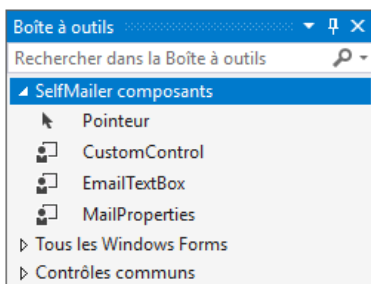
```
■ base.OnValidating(e);
```

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 :

