



Ressourcesinformatiques

C# sous Windows Phone

Développez et publiez
vos applications sur le Store



Christopher
MANEU

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 **RIWPCSHA** 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 Windows Phone

1.	Windows Phone, le téléphone selon Microsoft	15
1.1	Une nouvelle manière de concevoir le téléphone	17
1.2	Windows Phone aujourd'hui	19
2.	L'architecture de Windows Phone	20
2.1	Le matériel	20
2.2	La plateforme	22
2.2.1	Les options de développement disponibles à partir de Windows Phone 8.0.	22
2.2.2	Windows Phone 8.1 et les applications universelles	23
2.2.3	Le développement de jeux vidéo	25
2.2.4	Les modes d'exécution	25
2.3	Windows Phone et Windows 8	27
3.	Le langage C# et le framework .NET	28
3.1	Le Common Language Runtime	30
3.2	La Base Class Library	32
3.3	.NET dans Windows Phone	33

2 _____ C# sous Windows Phone

Développez et publiez vos applications sur le Store

Chapitre 2 Présentation de Visual Studio

1.	Installation et premier démarrage.....	35
1.1	Configuration nécessaire.....	35
1.1.1	Configuration requise pour Visual Studio 2013.....	35
1.1.2	Configuration spécifique requise pour l'émulateur Windows Phone 8.....	37
1.1.3	Configuration requise pour le déploiement sur téléphone physique.....	39
1.2	Installation du kit de développement Windows Phone 8.....	40
2.	Découverte de Visual Studio 2013.....	46
2.1	Découverte de l'environnement.....	46
2.2	Les solutions et les projets.....	51
2.2.1	Les solutions.....	51
2.2.2	Les projets.....	53
2.3	Les outils de création d'interfaces graphiques.....	60
2.3.1	Designer XAML de Visual Studio.....	60
2.3.2	Blend pour Visual Studio.....	65
3.	Découverte du kit de développement Windows Phone 8.1.....	68
3.1	L'intégration à Visual Studio 2013.....	69
3.2	Utilisation de l'émulateur.....	70
3.3	Outil d'enregistrement du téléphone développeur.....	81
3.4	Outil de déploiement.....	82
3.5	Le Windows App Certification Kit.....	82
4.	Création des solutions et des projets.....	85
4.1	Les différents types de projets.....	85
4.2	Création d'un premier projet.....	87
4.3	Déploiement et test des applications.....	92
4.4	Déboguer votre application.....	93
4.4.1	Lancer une application avec le débogueur et analyser les exceptions.....	93
4.4.2	Analyser l'exécution d'une application.....	94
4.4.3	Fonctionnalités de débogage avancées.....	97

Chapitre 3 Les bases du langage C#

1. Les variables, constantes, structures et énumérations	101
1.1 Les variables	101
1.1.1 Définition d'une variable.	101
1.1.2 Types de variables	102
1.1.3 Types de données valués	103
1.1.4 Types de données références.	105
1.1.5 Portée d'une variable	108
1.1.6 Conversions	109
1.2 Les énumérations et les constantes	115
1.2.1 Les énumérations.	115
1.2.2 Les constantes	117
1.3 Les tableaux et les listes	118
1.3.1 Les tableaux	118
1.3.2 Les listes	119
2. Les opérateurs et les structures de contrôle.	120
2.1 Les opérateurs	120
2.1.1 Opérateurs d'assignation.	120
2.1.2 Opérateurs de comparaison	121
2.2 Structures de choix.	121
2.2.1 If	121
2.2.2 Switch.	122
2.3 Structures de boucles	123
2.3.1 For.	123
2.3.2 Foreach	123
2.3.3 While	124
2.3.4 Do while.	124
2.3.5 Break et continue.	124
3. Les procédures et les fonctions	125
3.1 Déclaration et usage d'une procédure	125
3.2 Déclaration et usage d'une fonction	126
3.3 Les paramètres des méthodes.	127
3.4 Les propriétés	129

4 _____ C# sous Windows Phone

Développez et publiez vos applications sur le Store

Chapitre 4

La plateforme Windows Phone

1. La programmation événementielle	131
1.1 Les événements en .NET	131
1.1.1 Traitement d'un événement	131
1.1.2 Création d'un événement	135
1.2 La programmation événementielle avec XAML	137
1.2.1 L'utilisation des événements en XAML	137
1.2.2 Les événements routés	139
2. La programmation XAML	140
2.1 Présentation de XAML	140
2.2 Les bases de XAML	141
2.2.1 Syntaxe des éléments	141
2.2.2 Structure d'un fichier XAML	143
2.2.3 Interactions avec le code C#	145
2.3 Séparer les traitements graphiques et métiers	146
3. Le modèle d'application Windows Phone	149
3.1 Le contenu d'un projet Windows Phone	149
3.1.1 Les fichiers et dossiers du projet	150
3.1.2 Le manifeste de l'application	150
3.1.3 Le fichier App.xaml	151
3.2 Le cycle de vie d'une application Windows Phone	152
3.2.1 Tombstoning et Fast app resume	153
3.2.2 Le cycle de navigation d'une application Windows Phone ..	154

Chapitre 5

La programmation orientée objet

1. Introduction à la programmation orientée objet	155
2. Les classes	156
2.1 Déclaration d'une classe	156
2.1.1 Les variables	157
2.1.2 Les méthodes	158
2.1.3 Le constructeur	159
2.1.4 Le destructeur	160

- 2.1.5 Les propriétés 163
- 2.1.6 Indexeurs et opérateurs 166
- 2.2 Utilisation d'une classe 169
- 2.3 Membres statiques 170
- 3. L'héritage 170
 - 3.1 Création d'une classe héritée 171
 - 3.2 Classes abstraites et finales 177
- 4. Les interfaces 179
 - 4.1 Déclaration d'une interface 179
 - 4.2 Utilisation d'une interface dans une classe 180
- 5. La gestion des exceptions 180
 - 5.1 La structure try-catch 181
 - 5.2 La levée et la création d'exceptions 184
 - 5.3 La gestion des exceptions avec Windows Phone 185
- 6. Les expressions lambda et les types anonymes 185

Chapitre 6
La création de l'interface utilisateur

- 1. Organiser et positionner les contrôles 187
 - 1.1 Retour sur XAML 187
 - 1.2 Les propriétés liées au positionnement 192
 - 1.2.1 Width et Height 192
 - 1.2.2 Les alignements 193
 - 1.2.3 Margin 195
 - 1.2.4 Un mot sur les pixels logiques 195
 - 1.2.5 Padding 197
 - 1.2.6 ZIndex 197
 - 1.3 Les conteneurs 198
 - 1.3.1 Grid 198
 - 1.3.2 StackPanel 201
 - 1.3.3 Canvas 201
 - 1.3.4 Border 202
 - 1.3.5 ScrollViewer 203

6 **C# sous Windows Phone**

Développez et publiez vos applications sur le Store

1.4	La fenêtre popup	204
1.4.1	MessageBox	204
1.4.2	Popup	205
2.	Utiliser les contrôles standards	206
2.1	Les contrôles de texte	206
2.1.1	TextBlock	206
2.1.2	TextBox	208
2.1.3	PasswordBox	210
2.2	Les boutons et contrôles de sélection.	210
2.2.1	Button.	210
2.2.2	CheckBox	211
2.2.3	RadioButton	212
2.2.4	Slider	213
2.2.5	ProgressBar	214
2.3	Les listes.	214
2.3.1	ListBox	214
2.3.2	LongListSelector.	217
2.4	Les contrôles multimédias	224
2.4.1	Image	224
2.4.2	MediaElement	225
2.4.3	WebBrowser	226
3.	Personnaliser l'apparence de son application	228
3.1	Les ressources	229
3.1.1	Déclarer et utiliser des ressources	229
3.1.2	Regrouper des ressources dans des dictionnaires de ressources	231
3.2	Les styles	232
3.3	Les modèles	233
3.4	La typographie.	236
3.4.1	Les propriétés typographiques.	236
3.4.2	Utiliser une police personnalisée.	237
3.4.3	Utiliser des polices de symbole	238
3.5	Adapter l'interface à l'orientation et aux différentes résolutions.	240
3.5.1	Gérer les différentes orientations	240
3.5.2	Gérer les différentes résolutions d'écran.	243

- 4. Gérer la navigation au sein de son application 247
 - 4.1 Naviguer entre les différentes pages 247
 - 4.1.1 Le système de navigation de Windows Phone 247
 - 4.1.2 Passage de paramètres entre les pages 248
 - 4.2 Personnaliser la navigation 250
 - 4.2.1 Revenir à la page précédente 250
 - 4.2.2 Supprimer l'historique de navigation 250
 - 4.2.3 Empêcher le retour en arrière 251
 - 4.3 Intégrer la navigation à son application 251
 - 4.3.1 Utiliser les HyperLinkButton 251
 - 4.3.2 Personnaliser les liens avec un UriMapper 252
- 5. Animer l'interface utilisateur 253
 - 5.1 Animer son interface avec les storyboards 253
 - 5.1.1 Création des storyboards en XAML 253
 - 5.1.2 Création des storyboards en C# 256
 - 5.1.3 Animations et performances 258
 - 5.2 Animer un contrôle avec les états visuels 258
 - 5.2.1 Personnaliser les états visuels de contrôles existants 258
 - 5.2.2 Création d'états visuels pour les pages 261
 - 5.3 Utiliser les animations signatures de Windows Phone 261
 - 5.3.1 Utilisation du Tilt Effect 261
 - 5.3.2 Animer les transitions de pages 263

Chapitre 7
La gestion des données

- 1. Lier les données métiers à l'interface utilisateur 265
 - 1.1 Utiliser le binding et le contexte de données 265
 - 1.1.1 Principes du binding 265
 - 1.1.2 Créer une classe source supportant la liaison 267
 - 1.1.3 Convertir ses données pour l'affichage 269
 - 1.1.4 Les options de gestion du binding 271
 - 1.2 Afficher des données dans les outils de conception 271
 - 1.2.1 Utiliser un fichier XAML pour les données 272
 - 1.2.2 Créer une classe de données 275

8 **C# sous Windows Phone**

Développez et publiez vos applications sur le Store

2.	Utiliser le système de fichiers et gérer les paramètres	276
2.1	Manipuler les fichiers	276
2.1.1	Lire et écrire des fichiers.	276
2.1.2	Gérer les fichiers et les dossiers	279
2.1.3	Synchroniser des fichiers entre les périphériques d'un utilisateur	284
2.1.4	Accéder aux fichiers de l'utilisateur.	285
2.2	Sécuriser les données.	291
2.3	Gérer les paramètres de l'utilisateur.	294
2.3.1	Paramètres locaux des utilisateurs	294
2.3.2	Synchroniser les paramètres utilisateurs	296
2.4	Partager les données avec d'autres applications	297
2.4.1	Partager des données via le presse-papiers	298
2.4.2	Partager des données via le contrat de partage.	298
3.	Interagir avec les services web.	302
3.1	Effectuer des requêtes HTTP	303
3.1.1	Effectuer une requête de type GET	303
3.1.2	Effectuer une requête de type POST	305
3.1.3	Effectuer une requête personnalisée	307
3.1.4	Communiquer avec des services web SOAP	307
3.2	Analyser les réponses des services web	308
3.2.1	Analyser les réponses au format JSON.	308
3.2.2	Analyser les réponses au format XML	312
3.3	S'authentifier auprès des services web.	314
3.3.1	Utiliser l'authentification basique	314
3.3.2	Utiliser l'authentification par en-têtes HTTP	315
3.3.3	Utiliser l'authentification OAuth	315
4.	Utiliser une base de données embarquée	316
4.1	Ajouter le support des bases de données SQLite au projet.	317
4.2	Se connecter à la base et créer des tables.	321
4.2.1	Préparer les classes représentant les tables	321
4.2.2	Se connecter à la base et créer les tables	322
4.2.3	Exécuter des requêtes SQL brutes.	323
4.3	Effectuer les opérations CRUD sur la base	323
4.3.1	Ajouter un enregistrement	324
4.3.2	Modifier un enregistrement	324
4.3.3	Supprimer un enregistrement	325

4.3.4 Lire l'ensemble des enregistrements d'une table 325
4.3.5 Requête certains enregistrements d'une table 326

Chapitre 8

Gérer le cycle de vie d'une application

1. Gérer les différents états de l'application 327
1.1 Gérer l'activation et la désactivation d'une application 327
1.1.1 Le lancement de l'application 328
1.1.2 La désactivation de l'application 329
1.1.3 L'activation de l'application 331
1.2 Gérer le Fast App Resume 332
1.3 Détecter et modifier la mise en veille 334
2. Structurer son code avec MVVM 336
2.1 Un peu de théorie autour de MVVM 336
2.1.1 Pourquoi MVVM ? 336
2.1.2 L'architecture MVVM 337
2.1.3 Quelques détails sur l'architecture MVVM 337
2.2 Mettre en place l'architecture MVVM 338
2.2.1 Création du modèle 338
2.2.2 Création de la vue 340
2.2.3 Mise en place du ViewModel 343
2.2.4 Lier la vue au ViewModel 345
2.2.5 Réagir aux interactions des utilisateurs avec les commandes 346
3. Exécuter du code en arrière-plan 349
3.1 Utiliser un agent de tâches planifiées 349
3.1.1 Les types d'agents de tâches planifiées 349
3.1.2 L'implémentation d'un agent de tâches planifiées 350
3.1.3 Les bonnes pratiques du développement
d'agents de tâches planifiées 353
3.2 Télécharger des données en arrière-plan 354
3.2.1 Télécharger un fichier en arrière-plan 355
3.2.2 Envoyer un fichier sur un serveur en arrière-plan 357
3.3 Les états d'exécution en arrière-plan spécifiques 359
3.3.1 Audio 359
3.3.2 L'envoi de photos en arrière-plan 360

10 _____ C# sous Windows Phone

Développez et publiez vos applications sur le Store

3.3.3	Application de suivi de la localisation.	360
4.	Partager du code avec une autre plateforme	360
4.1	Partager des assemblies avec la PCL.	361
4.2	Partager des fichiers sources entre projets.	362
4.3	Quelques techniques pour le partage de code.	364
4.3.1	L'utilisation d'interfaces et d'implémentations par plateforme	365
4.3.2	L'utilisation de classes partielles	365

Chapitre 9

Aller plus loin avec l'interface utilisateur

1.	Une interface plus agréable avec les contrôles signatures et le Windows Phone Toolkit	367
1.1	Les contrôles signatures	367
1.1.1	Le contrôle Pivot	367
1.1.2	Le contrôle Panorama	371
1.1.3	La barre d'application.	374
1.2	Les contrôles du Windows Phone Toolkit.	378
1.2.1	Le contrôle ToggleSwitch	378
1.2.2	Les menus contextuels.	379
1.2.3	Le ListPicker	382
1.2.4	Le champ de saisie avec autocomplétion	384
2.	Intégrer le contrôle de cartographie	387
2.1	Afficher une carte avec la position courante.	387
2.1.1	Récupérer la position courante de l'utilisateur.	388
2.1.2	Intégrer et manipuler le contrôle de cartographie	391
2.1.3	Utilisation avancée du contrôle de cartographie	394
2.2	Afficher des données sur une carte	397
2.2.1	Ajouter des points sur une carte	397
2.2.2	Ajouter des tracés et des formes géométriques sur une carte	398
2.2.3	Ajouter des objets sur une carte	400
2.3	Guider l'utilisateur en déplacement	402
2.3.1	Suivre la position de l'utilisateur en continu	402
2.3.2	Calculer un itinéraire	405
2.3.3	Rediriger l'utilisateur vers l'application de navigation.	408

- 3. Utiliser la voix comme moyen d'interaction 409
 - 3.1 Lancer l'application grâce à la voix 409
 - 3.1.1 Utiliser les commandes vocales pour lancer l'application 409
 - 3.1.2 Personnaliser les commandes vocales lors de l'exécution 414
 - 3.2 Utiliser la reconnaissance vocale 414
 - 3.2.1 Reconnaître du texte libre. 414
 - 3.2.2 Reconnaître une grammaire préétablie 416
 - 3.3 Parler à l'utilisateur 417
 - 3.3.1 Dictier un texte simple. 417
 - 3.3.2 Personnaliser l'expression de la voix 418
- 4. Adapter l'interface pour différentes langues et cultures 419
 - 4.1 La gestion de l'internationalisation dans Windows Phone 419
 - 4.1.1 Pourquoi internationaliser l'application ? 419
 - 4.1.2 Le processus d'internationalisation. 421
 - 4.2 Traduire l'application 423
 - 4.2.1 Traduire les éléments de l'interface 423
 - 4.2.2 Traduire les éléments présents dans votre code 424
 - 4.2.3 Ajouter des langues à l'application 424
 - 4.3 Adapter l'application aux différences régionales. 426
 - 4.3.1 Adapter les dates et heures 426
 - 4.3.2 Adapter l'affichage des devises 427

Chapitre 10

Intégrer les applications à Windows Phone

- 1. Utiliser l'écran d'accueil et les notifications 429
 - 1.1 Créer des tuiles 430
 - 1.1.1 Personnaliser la tuile principale 431
 - 1.1.2 Créer des tuiles secondaires. 434
 - 1.2 Utiliser les notifications locales 436
 - 1.2.1 Envoyer un toast 436
 - 1.2.2 Gérer le centre de notifications. 439
 - 1.2.3 Mettre à jour les tuiles régulièrement 441
 - 1.3 Utiliser les notifications push depuis un serveur 442
 - 1.3.1 Récupérer un canal de communication pour un téléphone 443
 - 1.3.2 Envoyer une notification. 444

12 _____ C# sous Windows Phone

Développez et publiez vos applications sur le Store

1.3.3	Gérer les notifications lors de l'exécution	448
2.	S'intégrer aux autres applications	449
2.1	Lancer les applications système	449
2.2	Associer l'application à un protocole	451
2.3	Associer l'application à un type de fichiers	453
3.	Utiliser l'environnement du téléphone	456
3.1	Utiliser l'emplacement géographique de l'utilisateur	456
3.1.1	Mettre en place une geofence	457
3.1.2	Gérer les événements geofence dans l'application	459
3.1.3	Gérer les événements geofence en arrière-plan	460
3.2	Utiliser les informations de positionnement dans l'espace	463
3.2.1	Connaître l'orientation avec le compas	464
3.2.2	Réagir aux mouvements du téléphone avec l'API Motion	465
3.3	Échanger des données par contact avec le NFC	470
3.3.1	Échanger de l'information avec un périphérique NFC	470
3.3.2	Communiquer entre deux périphériques à proximité	474

Chapitre 11

Conception et ergonomie des applications

1.	Introduction	479
2.	Le processus de conception d'une application mobile	479
2.1	L'idée	480
2.1.1	Réfléchir, imaginer et construire son idée	481
2.1.2	Consolider ses idées	485
2.2	L'architecture	486
2.2.1	Établir la structure de l'application	486
2.2.2	Peut-on commencer à développer dès maintenant ?	488
2.3	L'ergonomie et l'interaction	488
2.3.1	Réfléchir à ses écrans avec le sketching et le wireframing	488
2.3.2	Les autres types d'interactions	493
2.4	Le visuel	493

- 3. Les principes de design et d'ergonomie de Windows Phone 495
 - 3.1 L'origine de la "conception moderne Microsoft" 495
 - 3.2 Les principes de la conception moderne Microsoft 496
 - 3.2.1 Fierté de notre savoir-faire 496
 - 3.2.2 Faire plus avec moins 497
 - 3.2.3 Rapidité et fluidité 498
 - 3.2.4 Numérique authentique 499
 - 3.2.5 Gain collectif 499
- 4. Quelques notions de design pour le développeur 500
 - 4.1 La typographie 500
 - 4.2 Les couleurs 506
 - 4.3 La mise en page 511
- 5. Quelques notions d'ergonomie pour le développeur 512
 - 5.1 L'affordance 512
 - 5.2 La loi de Fitts et la disposition des contrôles 513
 - 5.3 Le nombre de Miller 514
- 6. Conclusion 515

Chapitre 12
Diffuser et faire évoluer l'application

- 1. Préparer et tester son application 517
 - 1.1 Tester la performance et la réactivité de son application. 517
 - 1.1.1 Utiliser les compteurs de performances 517
 - 1.1.2 Utiliser les outils de performances 519
 - 1.2 Tester le fonctionnement de son application 524
 - 1.2.1 Utiliser les tests unitaires 524
 - 1.2.2 S'assurer du fonctionnement de l'application
 dans toutes les situations 527
- 2. Soumettre son application sur le Windows Phone Store 528
 - 2.1 Créer un compte développeur 528
 - 2.2 Soumettre une application en accès bêta 530
 - 2.3 Soumettre une application publique 534
 - 2.4 Les autres types de soumission 535
 - 2.4.1 Soumettre une mise à jour 535
 - 2.4.2 Soumettre une application cachée 537

14 _____ C# sous Windows Phone

Développez et publiez vos applications sur le Store

2.4.3	Déploiement d'applications d'entreprise.	538
3.	Monétiser son application.	538
3.1	Vendre son application.	539
3.1.1	Paramétrer la vente de son application.	539
3.1.2	Proposer une expérience d'essai.	541
3.2	Utiliser les achats additionnels.	542
3.2.1	Intégrer les achats additionnels dans son application	542
3.2.2	Tester les achats additionnels	545
3.3	Rémunérer son application via la publicité.	547
3.3.1	Les principes de la publicité sur mobile	547
3.3.2	Intégrer le SDK Microsoft Advertising.	549
4.	Surveiller et gérer son application publiée	553
4.1	Utiliser les outils du tableau de bord	553
4.1.1	Suivre les téléchargements et les commentaires des utilisateurs	553
4.1.2	Suivre les exceptions	555
4.2	Utiliser des outils tiers	556
4.2.1	Utiliser un outil de suivi des exceptions.	556
4.2.2	Contrôler son application à distance	557
	Index.	559

Chapitre 6

La création de l'interface utilisateur

1. Organiser et positionner les contrôles

1.1 Retour sur XAML

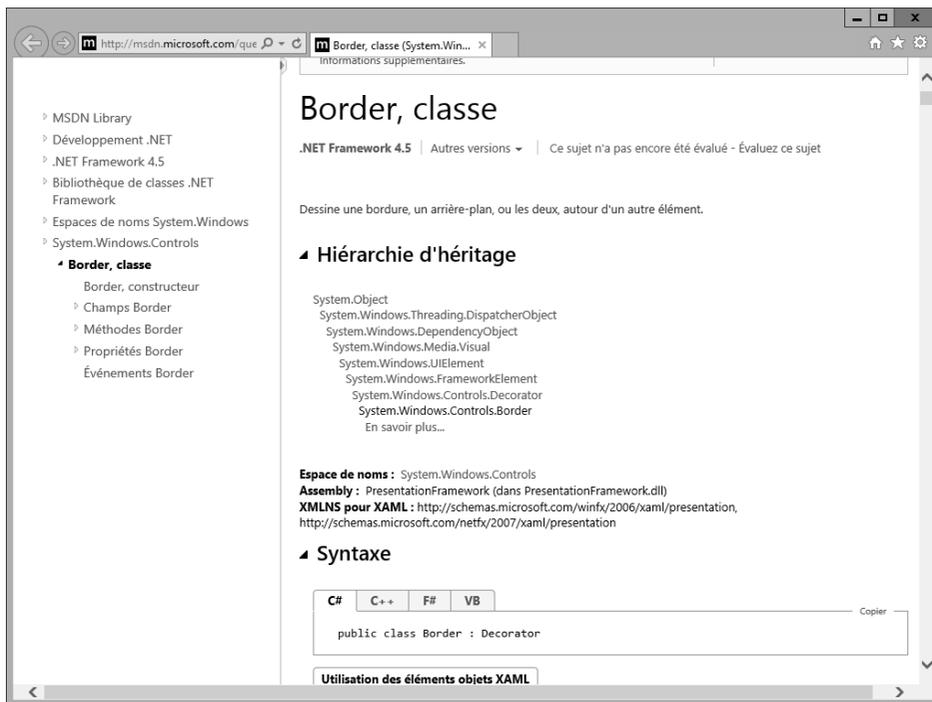
Nous avons déjà abordé l'écriture de code XAML dans le chapitre La plateforme Windows Phone. Ce chapitre va aborder les détails de la création de vos interfaces utilisateur avec ce langage.

La plateforme XAML disponible sous Windows Phone 8 (qui peut différer des autres plateformes disponibles, par exemple, pour les applications Windows Store ou WPF) offre un ensemble de contrôles afin de créer votre interface. Ces contrôles partagent un ensemble de propriétés ou de comportements. Nous allons tout d'abord nous pencher sur ces similitudes avant de voir ce que ces contrôles peuvent vous apporter.

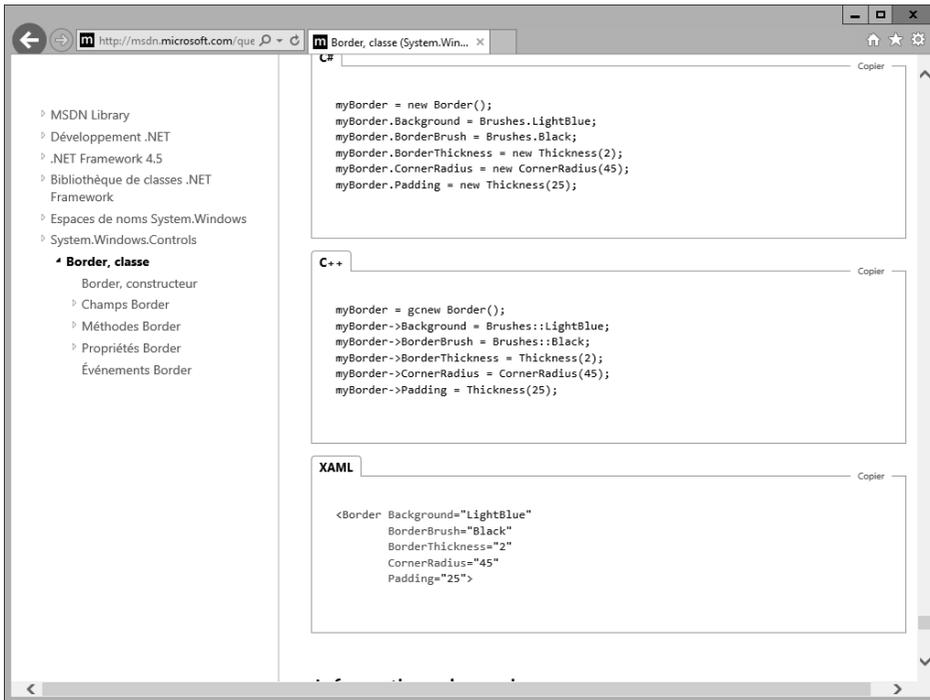
■ Remarque

Il existe des contrôles créés par des sociétés (telles que Telerik) ou disponibles sous licences libres. Ils peuvent vous faire gagner du temps ou enrichir rapidement vos applications. Nous verrons une partie de ces contrôles dans le chapitre Aller plus loin avec l'interface utilisateur.

Tous les contrôles disponibles sont accessibles directement via l'Intellisense dans un fichier XAML, ou depuis la barre d'outils. Si vous positionnez le curseur de l'éditeur au sein d'une balise XAML et que vous appuyez sur la touche [F1] de votre clavier, le navigateur s'ouvre directement sur la page d'aide du contrôle courant.



Cette page nous apporte plusieurs informations intéressantes. Tout d'abord, cela nous permet de nous remémorer que tous les objets graphiques ne sont que des classes. Ils sont donc accessibles également depuis du code C#, que cela soit pour créer des objets, ou pour modifier les propriétés d'objets définis dans du code XAML. La documentation en ligne apporte également des exemples de code pour utiliser ces contrôles.



Remarque

Il vous est possible de consulter la documentation sans connexion Internet. La procédure est expliquée sur <http://blog.maneu.net/accéder-a-la-documentation-visual-studio-windows-phone-net-hors-connexion/>.

Cette page de documentation nous apporte également une autre information intéressante : la hiérarchie d'héritage. Tous les contrôles graphiques héritent de la classe `DispatcherObject`. Ils sont donc intégrés au thread `Dispatcher` que nous avons abordé dans le chapitre La plateforme Windows Phone. D'autres classes parentes apportent des propriétés utiles pour un grand nombre de contrôles, telles que les propriétés `Width` et `Height` que nous aborderons plus loin dans ce chapitre.

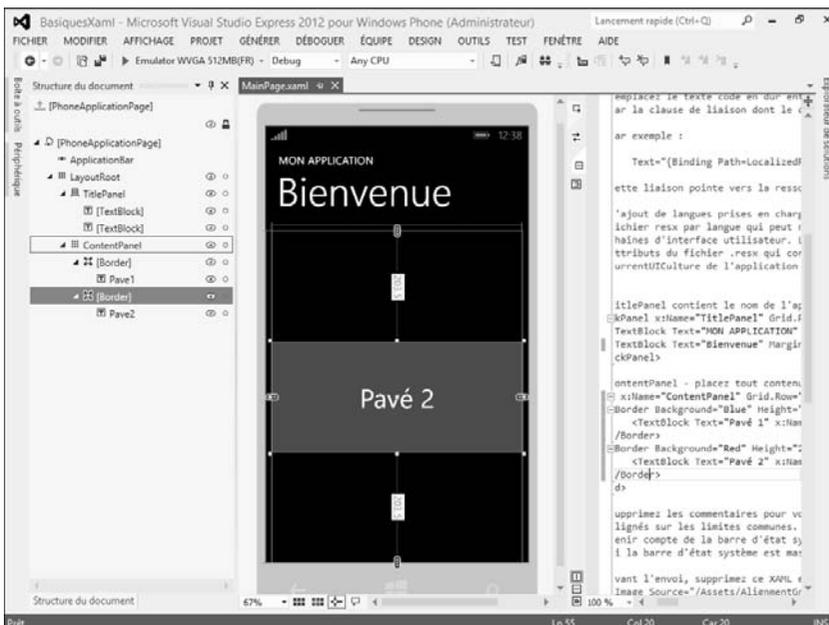
Avant de nous pencher sur les propriétés directement liées au positionnement, il faut comprendre comment fonctionne le moteur XAML pour calculer la mise en page des objets graphiques. Lorsqu'une propriété impactant le positionnement change de valeur, telle que `Height` ou `HorizontalAlignment`, le recalcul de l'écran n'est pas effectué directement. C'est le système qui décide quand est exécuté le prochain cycle de mise en page. Celui-ci est réalisé en deux phases: **Measure** et **Arrange**.

La phase **Measure** va déterminer la taille nécessaire à l'objet pour être dessiné. La méthode `Measure()` est ainsi appelée sur tous les contrôles qui vont en retour indiquer la taille d'affichage qu'ils demandent. À partir de ces résultats, la phase **Arrange** va finir le processus de mise en page en définissant ainsi la taille réelle qu'obtiendra chacun des contrôles. La manière dont la taille de chaque élément est déterminée dépend des objets conteneurs, que nous verrons en détail plus loin dans ce chapitre.

Remarque

Ce principe de fonctionnement a de nombreuses implications, notamment en termes de performances, ou dans les animations. Il est donc important de le conserver à l'esprit dès que l'on crée des animations ou que l'on modifie via le code des propriétés impactant la positionnement.

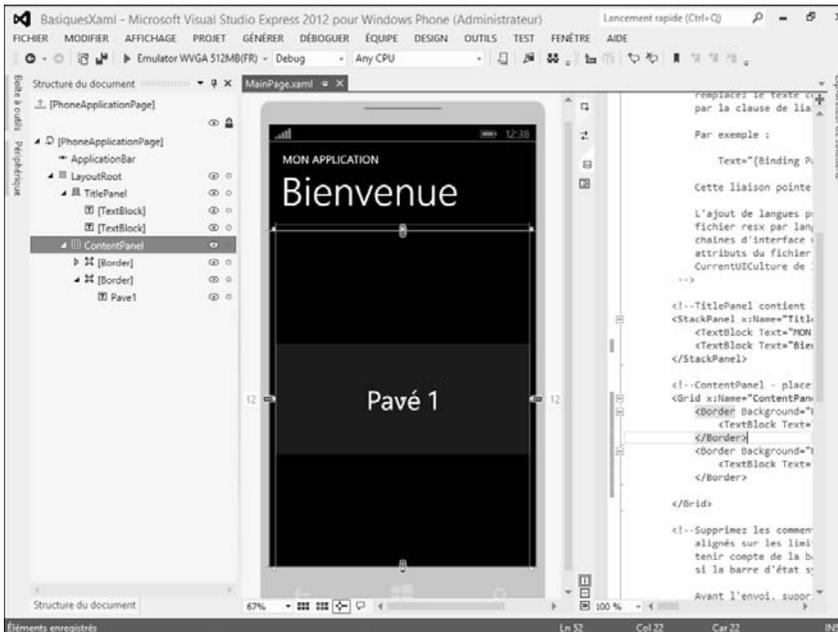
Pour calculer la mise en page d'une page entière, il suffit que le système appelle les méthodes `Measure()` et `Arrange()` sur le contrôle enfant de la page. Il va ainsi appeler ces deux mêmes méthodes sur l'ensemble de ses enfants, qui vont faire de même et ainsi de suite, afin de couvrir l'ensemble de la hiérarchie des contrôles. Cette **hiérarchie de contrôles XAML** n'a rien à voir avec la notion d'héritage telle qu'on l'a vu en C#, ou au début de ce chapitre. Il s'agit ici de l'ensemble de l'arborescence de contrôles qui a été définie en XAML (ou en C#). On peut visualiser cette hiérarchie dans l'éditeur XAML avec le panneau **Structure du document**.



Cette structure, et l'ordre des contrôles dans cette hiérarchie, a un impact sur la mise en page. Par exemple, prenons le code utilisé dans la capture d'écran précédente. Nous pouvons voir qu'au sein de l'élément `Grid` nommé `ContentPanel`, nous avons défini deux contrôles `Border`. Un seul de ces éléments est visible, celui qui a été défini en dernier dans le code XAML.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <Border Background="Blue" Height="200">
        <TextBlock Text="Pavé 1" x:Name="Pave1"
HorizontalAlignment="Center" VerticalAlignment="Center"
FontSize="48"/>
    </Border>
    <Border Background="Red" Height="200">
        <TextBlock Text="Pavé 2" x:Name="Pave2"
HorizontalAlignment="Center" VerticalAlignment="Center"
FontSize="48"/>
    </Border>
</Grid>
```

Si nous inversons les deux blocs afin que le pavé 1 soit déclaré en dernier, c'est bien ce second pavé qui est visible. Ce comportement est dû à deux facteurs : le fonctionnement du contrôle `Grid`, et l'ordre de déclaration. Les contrôles déclarés en fin de fichier ont la priorité sur ceux déclarés précédemment.



1.2 Les propriétés liées au positionnement

1.2.1 Width et Height

De nombreuses propriétés peuvent affecter directement le positionnement. Dans cette liste, `Width` et `Height` sont certainement les plus simples. Elles permettent, respectivement, d'indiquer la largeur et la hauteur souhaitées. Afin de mieux comprendre comment ces deux propriétés fonctionnent, nous allons prendre les quatre exemples suivants, en appliquant ces propriétés sur les objets `Border`.

```
<StackPanel x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
  <Border BorderBrush="Aqua" BorderThickness="3">
    <TextBlock Text="Bloc 1" />
  </Border>
  <Border Width="100" BorderBrush="Red" BorderThickness="3">
    <TextBlock Text="Bloc 2" />
  </Border>
  <Border Width="Auto" BorderBrush="MediumPurple"
BorderThickness="3">
    <TextBlock Text="Bloc 3" />
  </Border>
  <Border Height="180" BorderBrush="Orange"
BorderThickness="3">
    <TextBlock Text="Bloc 4" />
  </Border>
</StackPanel>
```

Le premier bloc n'a aucune de ces propriétés définie explicitement. C'est donc l'objet parent – ici le `StackPanel` – qui va déterminer la hauteur et la largeur de ce bloc.

Dans le second bloc, nous avons explicitement défini une largeur, exprimée en pixels. Cette largeur étant inférieure à la largeur du `StackPanel` parent, elle est directement appliquée. Il est possible de définir une largeur ou une hauteur plus grande que la taille dont nous disposons, mais le résultat dépend alors du conteneur utilisé.

Le troisième bloc a le même positionnement que le premier bloc, alors qu'il définit la valeur `Auto` comme largeur. `Auto` est en fait une valeur spécifique en XAML (traduite en `Double.NaN` en code) qui indique au système qu'il doit évaluer la valeur en fonction d'autres propriétés et du conteneur.