

Développement Windows 8

**Créer des applications
pour le Windows Store**

API Windows Runtime (WinRT) • C#/C++/XAML
.NET • JavaScript/HTML5

Louis-Guillaume Morand
Luc Vo Van
Alain Zanchetta

Préface de Bernard Ourghanlian



EYROLLES

© Groupe Eyrolles, 2013, ISBN : 978-2-212-13643-2

Avant-propos

L'interface de Windows 8, au premier abord déroutante, cache une révolution : celle des interfaces de programmation (API). Si les technologies de développement Windows classiques restent disponibles dans l'environnement du bureau, de nombreuses API apparaissent. La plus importante, WinRT, permet de développer pour le nouveau Windows Store.

Pourquoi ce livre ?

Le nouveau visage de Windows, c'est WinRT, une nouvelle API très riche conçue pour le développement d'applications vendues via le nouveau Windows Store. WinRT touche à tous les aspects du développement applicatif, de la présentation graphique au dialogue avec les matériels, notamment les nombreux capteurs conçus pour Windows 8 (accéléromètre, NFC...), en passant par les fondamentaux tel que les accès réseau et la lecture ou écriture de fichiers.

Afin de garantir sécurité et qualité, les applications écrites en respectant les nouveaux standards sont publiées via le Windows Store, la plate-forme en ligne de distribution Microsoft ; elles doivent répondre à un cahier des charges technique très strict vérifié par Microsoft lors du processus de publication.

Ce livre vous guide à travers les principaux concepts de programmation de cette nouvelle génération d'applications, et décrit leur mise en œuvre à travers de nombreux exemples de code. Il distille également des bonnes pratiques, allant de certains principes fondamentaux de programmation, au respect des normes conditionnant la publication dans le Windows Store, en passant par des recommandations en vue de favoriser l'adoption des applications par leurs utilisateurs – but ultime du développement applicatif orienté vers le grand public.

À qui s'adresse ce livre ?

Écrit par des développeurs, ce livre s'adresse aux programmeurs qui connaissent déjà certaines technologies de développement Windows ou web (.NET, C++, HTML5, JavaScript...) mais qui peuvent se sentir démunis devant la nouvelle interface de programmation. Pourtant, quelle que soit l'ampleur de l'évolution des API, Microsoft a toujours privilégié la capitalisation des connaissances, plutôt que de forcer le développeur à abandonner tout ce qu'il sait et tout ce qu'il a écrit. Ainsi les applications du Windows Store peuvent-elles être programmées avec les langages de .NET tels C# et Visual Basic .NET, comme en JavaScript et HTML5, voire en C++.

Ce livre s'inscrit dans cette approche. Il peut être lu par tout programmeur maîtrisant l'une de ces technologies, grâce au socle commun que constitue WinRT.

Les exemples de code sont ainsi donnés tant en C# ou C++ qu'en JavaScript. Outre qu'elle permet de s'adresser à un public élargi, cette triple approche permet au lecteur, soucieux de s'informer pour faire les bons choix, de faire la part des choses entre le concept lui-même et les éventuelles difficultés de syntaxe liées à tel ou tel langage. D'ailleurs, pourquoi n'en profiterait-il pas pour découvrir d'autres technologies ou langages lui permettant de mieux réaliser ses idées ?

Structure de cet ouvrage


Chaque chapitre traite un pan de la programmation d'applications pour le Windows Store :


- Le **premier chapitre** présente les principes fondamentaux des applications Windows Store, ce qui les rassemble et ce qui les différencie des applications Windows classiques.
- Le **chapitre 2** aborde l'ergonomie des applications – les utilisateurs étant de plus en plus exigeants. Une application moderne se doit d'offrir un excellente « expérience utilisateur », d'autant plus que l'ergonomie fait partie des critères de validation avant publication.
- Le **chapitre 3** décrit comment modéliser les données utilisées par une application, comment les afficher et les mettre à jour de manière automatique et comment les rendre persistantes.
- Le **chapitre 4** aborde une technique de développement omniprésente depuis la sortie de Windows 8 : la programmation dite asynchrone, qui garantit à l'utilisateur une interface toujours réactive – mais au prix d'un changement de paradigme pour le développeur.
- Le **chapitre 5** détaille le cycle de vie des applications qui a vu son modèle modifié pour que le système puisse proposer une expérience utilisateur optimale basée sur la fluidité et l'accessibilité sur les unités mobiles (tablettes et smartphones, principalement).


- Le **chapitre 6** décrit comment les applications s'intègrent avec l'écran d'accueil et ses vignettes dynamiques, en abandonnant la métaphore du bureau si longtemps présente sous Windows.
- Le **chapitre 7** présente les nouveaux moyens standardisés de communication et d'échange entre applications comme le partage de données ou la recherche d'information.
- Le **chapitre 8** explique les techniques d'animation qui accompagnent l'utilisateur dans sa découverte de l'environnement d'une application.
- Le **chapitre 9** décrit un aspect essentiel de la nouvelle interface de Windows : la gestion du tactile.
- Le **chapitre 10** présente les moyens d'interagir avec le matériel et notamment avec les nombreux capteurs qui équipent les tablettes, les smartphones et autres matériels modernes.
- Le **chapitre 11** décrit la procédure de soumission d'une application au Windows Store et les bonnes pratiques associées pour aider le développeur dans cette nouvelle phase assez technique.
- Le **chapitre 12** décrit les moyens d'accéder au réseau Internet, omniprésent dans notre vie et souvent crucial pour le fonctionnement des applications modernes.
- Le **chapitre 13** propose au développeur expérimenté un voyage au centre des mécanismes faisait vivre les applications du Windows Store pour compléter ses connaissances et mieux en maîtriser les rouages sous-jacents.


L'histoire d'une aventure de 13 passionnés !

Nous ne pouvons pas vous laisser démarrer la lecture de cet ouvrage sans vous expliquer comment ce projet a démarré. Ce livre fait partie d'un projet de rédaction de quatre ouvrages sur la thématique du poste de travail, essentiellement sur Windows 8. Ce projet a réellement démarré au début de l'année 2012. Les 13 auteurs de ces ouvrages sont des passionnés et experts reconnus dans leurs domaines respectifs.

 *Sécurité et mobilité du poste de travail Windows 8*, Arnaud Jumelet, Stanislas Quastana et Pascal Sauliere, Eyrolles 2013

 *Développement Windows 8*, Louis-Guillaume Morand, Luc Vo Van et Alain Zanchetta, Eyrolles 2013

 *Virtualisation du poste de travail Windows 7 et 8*, William Bories, Abderrahmane Laachir, Philippe Lafeil, David Thieblemont et François-Xavier Vitrant, Eyrolles 2013

 *Déploiement et migration Windows 8*, William Bories, Olivia Mirial et Stéphane Papp, Eyrolles 2013

D'autres passionnés chez Microsoft, pour certains des experts reconnus, ont apporté une précieuse contribution à ces ouvrages par une relecture profonde et pragmatique, au cours de laquelle ils ont pu mettre les auteurs à l'épreuve. Enfin, ceux qui n'avaient pas d'expertise technique particulière n'ont pas hésité à relire des dizaines de pages pour améliorer la qualité de ces ouvrages. Bref, c'est un projet atypique auquel nous avons tous pris beaucoup de plaisir à participer !

En espérant que vous partagerez ce plaisir, je vous souhaite une excellente lecture !

William Bories,
Coordinateur du projet.

Remerciements

Nous tenons en premier lieu à remercier les éditions Eyrolles qui nous ont soutenus pour ce projet ainsi que notre G.O William dont l'enthousiasme est sans égal. Merci également à Olivier Dahan pour sa contribution. Un grand merci aussi à Bernard Ourghalian pour cette belle préface et tout l'éclairage qu'il nous apporte en permanence sur les nouvelles technologies et les choix effectués par Microsoft.

Merci aussi à tous nos relecteurs sans lesquels cet ouvrage n'aurait pas la même qualité : Myriam Delesalle avant tout, mais aussi Christophe Nasarre-Soulier, Patrice Manac'h et bien sûr Michel Molongo, Audrey Petit, Nicolas Antoine, Sébastien Bovo, David Coppet, Isabelle Leboucher et André Zambalas, tous volontaires pour relire sans relâche notre prose pendant leur temps libre afin de donner à ce livre la qualité qu'il possède aujourd'hui.

Remerciements de Louis-Guillaume Morand

Ce livre, mon troisième chez Eyrolles, fut pour moi le plus plaisant à écrire du fait de l'avoir réalisé avec Alain et Luc, deux grosses pointures de qui j'ai toujours beaucoup appris. C'est donc eux que je tiens à remercier en premier. Puis vient Sophie ma moitié qui a gentiment accepté de me voir passer mes soirées et nuits sur l'ordinateur, à écrire ; enfin, je remercie tous les copains dont la présence permet de s'aérer la tête et de profiter de la vie.

Remerciements de Luc Vo Van

Je suis, depuis très jeune, un fervent lecteur de livres liés à l'informatique. Être auteur a toujours été l'un de mes rêves et c'est donc avec enthousiasme que j'ai rejoint Louis-Guillaume et Alain dans la rédaction de cet ouvrage. Je tenais à remercier tout particulièrement ma femme Yseult et ma fille Clémentine pour leur patience ; Clémentine ayant su, de concert avec la rédaction de cet ouvrage, subtiliser chaque instant de sommeil potentiel.

Remerciements d'Alain Zanchetta

Je tiens à remercier ma femme et mes enfants pour leur patience et leur soutien : en plus de ce livre qui m'a occupé un certain nombre de soirées, je leur ai fait subir un déménagement de 8 000 km et de 360 cartons... 2012 aura décidément été une année bien remplie. Je remercie bien évidemment aussi mes deux compagnons de route pour cette expérience très enrichissante.

Préface

8... Un chiffre qui porte chance dans la culture chinoise, car il est homophone du mot « prospérité » en cantonais. Ce n'est pas un hasard si la cérémonie d'ouverture des Jeux olympiques de Pékin a eu lieu le 8 août 2008 à 20 heures et 8 minutes (08/08/08, 08:08) précises...

Ce chiffre 8 portera-t-il chance à Windows et à Microsoft ? Il est trop tôt pour le dire, et pourtant Windows 8 est probablement la version la plus importante depuis Windows 1.0. Qu'il me soit permis de tenter de le démontrer brièvement par une histoire (très abrégée) de ce système d'exploitation.

La première version de Windows a été rendue disponible le 20 novembre 1985. À cette époque, le véritable système d'exploitation était MS-DOS, l'interface graphique étant matérialisée sous la forme d'un *shell* graphique s'exécutant au-dessus du véritable système d'exploitation. Une telle approche avait du sens : elle permettait d'améliorer l'interface homme-machine sans remettre en cause l'essentiel, à savoir la compatibilité avec les applications MS-DOS. Puis est venu Windows NT (appelé Windows NT 3.1, qui est devenu disponible le 27 juillet 1993) dont l'objectif initial était de remplacer OS/2 version 2, et dont l'interface de programmation initiale – d'aucuns l'ont oublié aujourd'hui – était l'API du Presentation Manager d'OS/2.

En raison du succès initial de Windows 3.0, puis de Windows 3.1, Microsoft changea son fusil d'épaule et fit de Windows NT le futur remplaçant des versions Windows 3.1, Windows 95, Windows 98 et Windows Millenium Edition. Pour cela fut créée une nouvelle API, Win32, qui serait compatible avec les noms de fonctions, la sémantique et l'utilisation des types de données de l'API Windows 16 bits existante. En ce sens, Win32 peut être vu comme une extension 32 bits de Win16, visant à faciliter le portage des applications Windows 16 bits existantes vers Windows NT. Cette API fut également rendue disponible sur les différentes versions de systèmes d'exploitation fondées sur MS-DOS – à commencer par Windows 95. Tout cela finit

par converger en une seule version, qui devait être initialement Windows 2000, mais qui fut finalement Windows XP qui vit le jour officiellement le 25 octobre 2001. Sont alors sortis successivement Windows Vista et Windows 7, ces versions continuant de s'appuyer largement sur l'API Win32 (Win64 étant la variante de cette API implémentée pour les plateformes 64 bits).

Ce qui est intéressant, quand on considère cette histoire très brièvement résumée, c'est que pendant ces quelque 27 années, jamais Microsoft n'a changé à la fois l'interface homme-machine, l'environnement d'exécution et les API utilisées par les développeurs. Avec Windows 8, c'est ce qui se passe pour la première fois !

En fait, si vous m'autorisez à filer cette métaphore, ce sont les planètes Microsoft qui sont en train de s'aligner... Tout a commencé avec l'interface homme-machine (IHM) et Windows Phone 7 qui a introduit pour la première fois la nouvelle IHM, anciennement appelée METRO et désormais appelée « Microsoft Design Language » ; puis la Xbox lui a emboîté le pas et enfin Windows avec Windows 8 qui partagent la même interface homme-machine. Pour le système d'exploitation, c'est l'inverse : Windows Phone 8 a emprunté à Windows 8 ses technologies de base telles que son noyau, son système de fichiers, sa pile réseau, sa sécurité, son support multimédia et son navigateur. Il ne manque désormais plus que le noyau de la Xbox pour que les planètes Microsoft soient complètement alignées mais seul l'avenir pourra nous en dire davantage sur ce dernier sujet... En l'état, sur Windows 8 et Windows Phone 8, bien que les applications ne soient pas identiques, il est relativement trivial de construire des expériences cohérentes : mêmes outils (Visual Studio et Blend), partage d'un ensemble significatif de code entre les projets Windows 8 et Windows Phone 8 : C#/XAML, C++/DirectX, contrôles Web HTML 5... D'où, pour les développeurs, un regain d'intérêt certain pour la plateforme Windows.

Mais revenons-en à Windows 8... Comme on vient de le dire, Windows 8 dispose de la même interface homme-machine que celle que l'on trouve depuis Windows Phone 7 ; celle qui s'instancie dans ce que l'on appelle désormais les « Windows Store Apps ». Ces applications s'exécutent au-dessus d'un nouveau moteur d'exécution appelé Windows Runtime ou WinRT¹ qui offre aux développeurs toute une série d'API permettant le support des communications et des données, du graphique et du multimédia, des terminaux, du stockage et de l'impression, plus un composant d'infrastructure de taille relativement modeste.

Appesantissons-nous un court instant sur WinRT. Pendant toutes ces années, les applications présentes sur le bureau Windows ont été développées en utilisant, plus ou moins directement, les API Win32 (ou Win64), que ce soit pour afficher des gra-

1. Ne pas confondre WinRT, l'environnement d'exécution des applications Windows Store avec WinRT, l'abréviation de Windows RT, version de Windows 8 qui s'exécute sur les processeurs ARM.

phiques, dialoguer avec l'utilisateur, communiquer à travers le réseau, etc. Bien entendu, Windows 8 ne renie pas son passé, ne serait-ce que pour assurer une compatibilité transparente des applications Windows 7. WinRT, quant à lui, est constitué d'une collection de nouvelles API permettant la création d'un nouvel univers d'applications immersives s'exécutant en mode plein écran (sans plus de décoration de fenêtres) et appelées Windows Store Apps. Bien entendu, les applications Windows dites « de bureau » sont toujours disponibles et restent toujours pertinentes dans de nombreux cas. C'est notamment le cas des applications Office telles qu'Excel, Word ou PowerPoint. Ces applications peuvent d'ailleurs tirer parti de l'API WinRT, par exemple pour communiquer avec des capteurs.

Les Windows Store Apps ont été conçues pour communiquer avec WinRT à travers une couche dite de « projection » indépendante des langages qui permet à des applications d'être écrites à la fois dans des langages typés de manière statique tels que VB, C++ ou C# mais aussi dans des langages dynamiques et faiblement typés tel que JavaScript.

Comme on le verra dans la suite de cet ouvrage, WinRT repose sur une évolution extrêmement significative de COM à travers l'ajout d'une seconde interface à l'insaisissable `IUnknown`, appelée `IInspectable`. Mais ce nouveau COM facilite le partage des données entre différents langages ; il permet même l'utilisation de métadonnées, comme en environnement .NET. Ainsi, les métadonnées WinMD constituent l'un des éléments fondateurs du support multi-langage de WinRT ; c'est un élément commun à tous les langages qu'utilise d'ailleurs l'infrastructure de WinRT pour générer les projections dont nous parlions plus haut.

Windows 8 apporte donc, de manière évidente, tout un ensemble de nouveautés : nouvelle interface homme-machine, nouvel environnement d'exécution, nouvelles API. Un tel renouveau n'a jamais eu de précédent dans l'histoire de Microsoft. Windows 7, à titre de comparaison, est construit au-dessus du même environnement d'exécution « Explorer » que toutes les versions de Windows avant lui depuis Windows 2000 et offre au développeur les mêmes API Win32 (avec quelques nouvelles possibilités, évidemment). Bien entendu, WinRT reste, en interne, un client de Win32, comme n'importe quelle autre application ; mais ceci est rendu complètement transparent pour le développeur, quel que soit son langage.

En ce sens, Windows 8 constitue certainement un pari important pour Microsoft. Un véritable changement de paradigme... Mais pourquoi un tel pari ? Tout simplement parce que, s'il est vrai que Windows a donné vie aux ordinateurs personnels qui sont aujourd'hui utilisés par plus d'un milliard de personnes sur terre, Windows est resté un système d'exploitation assez proche des professionnels de l'informatique et des entreprises. Ainsi, l'utilisation de la plupart des fonctions du système nécessitaient – jusqu'à Windows 8 – un certain niveau de familiarité avec des concepts tels que les fichiers, les répertoires, les permissions, les partages, etc.

Pourtant la démarche d'Apple centrée sur le consommateur à travers des produits tels que l'iPhone et l'iPad a démontré au monde qu'une autre approche était possible, à travers une interaction intuitive avec l'ordinateur, sans nécessiter le moins du monde la connaissance préalable du concept de fichier, de répertoire ou de procédure d'installation d'application. Windows 8 constitue à n'en pas douter la réponse de Microsoft à cette nouvelle approche, reprenant à son compte la « consommation de l'informatique », ce phénomène sociologique irrésistible qui procède d'une démocratisation sans cesse plus grande du monde numérique qui nous entoure.

Windows 8 est donc une aventure. Aventure intellectuelle tout d'abord, car il a commencé comme dans l'un de ces cercles occultes où, tout d'abord, seuls quelques initiés sont admis à partager la connaissance. Puis, cette connaissance s'est diffusée à un cénacle un peu plus large où seuls les femmes et les hommes dont la compétence était reconnue par leurs pairs étaient admis. Enfin, le temps est venu de la dissémination beaucoup plus large de ce savoir, par laquelle on cherchait à convaincre et à « évangéliser ». Aventure industrielle aussi dans laquelle Microsoft a investi des milliards et jeté toutes ses forces pour donner vie à ce qui n'était, au moins au début, qu'un rêve dans l'esprit de quelques ingénieurs.

Aujourd'hui, cette aventure a pris forme, avec un système d'exploitation, Windows 8, disponible sur de nombreux environnements matériels très divers : PC de bureau traditionnel, portable, tablette, *ultrabook*, hybride, renforcé, etc.

Nul ne sait quelle sera la fin de cette aventure. De nombreuses pages restent encore à écrire. Mais, comme dans le merveilleux roman de Michael Ende, *L'Histoire sans fin*, le lecteur a sa part de l'histoire à écrire...

« Ceux qui comprennent ne comprennent pas qu'on ne comprenne pas » nous disait Paul Valéry. Ce n'est pas le cas de ceux qui ont choisi d'écrire cet ouvrage. Ils ont chaussé leurs bottes de pédagogues et mis tout leur talent pour mettre à la portée du plus grand nombre des concepts parfois opaques.

Car Windows 8 (et Windows Phone 8) est ouvert à tous les styles de développements et à tous les langages, quels que soient leurs types : procéduraux, orientés objets, fonctionnels, de script, typés fortement ou non, de manière dynamique ou statique... et cet ouvrage se devait de respecter toute cette diversité en étant accessible à tous.

Développeurs, empressez-vous de tourner la première page de cet ouvrage, vous êtes ici chez vous !

Bernard Ourghanlian,
CTO de Microsoft France

Table des matières

Avant-propos	1
Pourquoi ce livre ?	1
À qui s'adresse ce livre ?	2
Structure de cet ouvrage	2
Remerciements	4
<i>Remerciements de Louis-Guillaume Morand</i>	4
<i>Remerciements de Luc Vo Van</i>	4
<i>Remerciements d'Alain Zanchetta</i>	4
CHAPITRE 1	
Applications Windows Store	5
Qu'est-ce qu'une application Windows Store ?	6
Une expérience utilisateur plus fluide	6
Un cycle de vie plus contrôlé	6
Distribution et déploiement	8
<i>Distribution via le Windows Store</i>	8
<i>Déploiement interne en entreprise</i>	8
<i>Déverrouillage pour les développeurs</i>	9
Sécurité	9
Performances et autonomie	10
Windows 8 et Windows RT	10
Windows 8 sur processeurs x86 et x64	10
Windows RT pour les processeurs ARM	11
Une plate-forme de développement	11
Des API accessibles avec le Windows Runtime (WinRT)	12
HTML5 et JavaScript	13
.NET et XAML (eXtensible Application Markup Language)	15
C++ et XAML	16
Outillage du développeur	18
Le SDK Windows 8	18
Visual Studio 2012	19

Blend pour Visual Studio	19
MSDN, la référence documentaire	19
Composants tiers	20
Une première application	20
XAML	20
JavaScript et HTML	22

CHAPITRE 2

Éléments d'ergonomie **25**

Une philosophie de design	26
Scénarios d'utilisation	30
États d'affichage	31
<i>Plein écran</i>	31
<i>Ancré</i>	32
<i>Remplissage</i>	32
Concepts ergonomiques de navigation	33
<i>Navigation hiérarchique</i>	34
<i>Navigation linéaire</i>	34
<i>Navigation hybride</i>	35
Langage d'interaction tactile	35
<i>Mouvements</i>	35
<i>Manipulations</i>	36
Structure des écrans types	37
<i>Pages</i>	37
<i>Panoramiques</i>	39
Éléments d'interface	40
<i>Vignettes dynamiques</i>	40
<i>Notifications</i>	42
<i>Icônes et contrats</i>	44
<i>Barres d'applications</i>	44
<i>Zoom sémantique</i>	46
Langage visuel	47
<i>Absence de contours</i>	47
<i>Polices</i>	47
<i>Alignement sur une grille</i>	48
<i>Authentiquement numérique</i>	48
<i>Branding et identité visuelle</i>	48
<i>Style visuel et ergonomie</i>	49

CHAPITRE 3

Modèle, liaisons et accès aux données..... 51

Principes généraux	51
Le pattern MVVM (Model-View-ViewModel)	54
Liaison de données en XAML avec .NET	55
Liaison simple	55
Liaison de listes	61
Liaison de données en C++	64
Liaison de données en HTML5 et JavaScript	68
Accès aux données	73
Lecture et écriture dans un fichier	74
<i>Principes de base</i>	74
<i>Utilisation de fichiers textes</i>	76
<i>Accès simplifié : la classe FileIO</i>	78
<i>Utilisation d'un fichier XML</i>	78
Accès à des données distantes	81
<i>Service WCF simple</i>	81
<i>Utilisation d'une bibliothèque d'interface</i>	82
<i>Utilisation de WCF Data Services</i>	86

CHAPITRE 4

Programmation asynchrone..... 89

Asynchronisme et interface utilisateur réactive	89
IAsyncOperation	93
Async et await	96
Tâches	99
Pool de threads	101
Appels asynchrones en JavaScript	102
Appels asynchrones en C++	103
Bonnes pratiques	104

CHAPITRE 5

Cycle de vie des applications 105

Le principe de cycle de vie	106
Le principe de suspension	109
Responsabilités d'une application	109
Sauvegarde et restauration d'état	109
<i>Sauvegarde de l'état applicatif en C#</i>	111
<i>Sauvegarde de l'état applicatif en JavaScript</i>	114
<i>Sauvegarde de l'état applicatif en C++</i>	115

Reprise des données	116
<i>La reprise en C#</i>	116
<i>La reprise en JavaScript</i>	117
<i>La reprise en C++</i>	117
L'activation	117
<i>L'activation en C#</i>	118
<i>L'activation en JavaScript</i>	119
<i>L'activation en C++</i>	120
Conclusion	121

CHAPITRE 6

Les vignettes et notifications..... 123

Vignettes dynamiques	123
Organisation d'une vignette	125
<i>Mise à jour d'une vignette</i>	128
<i>Mise à jour du badge</i>	129
<i>Le respect des bonnes pratiques</i>	130
Vignettes secondaires	130
Création d'une vignette secondaire	131
Transmission de contexte	132
Bonnes pratiques	133
Utilisation du Windows Push Notification Service	133
Conclusion	137

CHAPITRE 7

Tirer profit de la puissance des charmes..... 139

La notion de contrat	139
Le charme de partage	140
Source de partage	142
<i>Partager autre chose que du texte</i>	144
Déclenchement du partage	146
Cible de partage	147
Le charme de recherche	150
La suggestion de résultat	153
<i>Les templates de contrat de recherche</i>	155
Le charme de paramétrage	156
Personnalisation de l'écran Paramètres	157
Affichage d'un écran personnalisé	158
Les associations applicatives	160
L'association de fichier	160

<i>Chargement du fichier</i>	162
L'association de protocole	164
<i>Utilisation du protocole</i>	165
Les API de lancement	166
<i>Ouverture de l'application, par défaut ou non, d'un fichier</i>	167
<i>Ouverture de l'application par défaut d'un protocole</i>	169
Conclusion	170

CHAPITRE 8

Techniques d'animation..... 171

Rôle et importance des animations	172
Guider l'utilisateur	172
Renforcer la présentation	172
Animations types	173
Transitions de thème (XAML)	173
Transition de contenu	175
<i>Implémentation XAML</i>	176
<i>Implémentation JavaScript</i>	176
Listes	178
<i>Implémentation XAML</i>	178
<i>Implémentation JavaScript</i>	178
Animations libres	180
Animations indépendantes et dépendantes	181
Animations indépendantes	181
<i>XAML</i>	181
<i>HTML/CSS</i>	181
Animations dépendantes	182
<i>XAML</i>	182
<i>HTML/CSS</i>	182
Storyboard (XAML)	182
<i>Définition</i>	183
<i>Utilisation</i>	183
États visuels (XAML)	184
<i>Définition des états visuels</i>	185
<i>Appliquer un état visuel</i>	187
<i>Édition des états visuels avec Blend</i>	187
Transitions CSS	190
<i>Événement DOM transitionend</i>	191
<i>Utilisation conjointe avec l'extension ms-view-state</i>	191

CHAPITRE 9

Gestion de l'interface tactile 193

Interactions utilisateur sous Windows	193
Considérations pour les interfaces tactiles	194
Les pointeurs	195
Détection des capacités tactiles	197
Mouvements et manipulations	198
GestureRecognizer	198
Utilisation du GestureRecognizer (JavaScript)	198
Événements de manipulation (XAML)	200
Conclusion	203

CHAPITRE 10

Interfaces avec les capteurs et périphériques 205

La géolocalisation	205
Accès ponctuel	207
Accès continu	209
Bonnes pratiques de la géolocalisation	210
Les capteurs	211
Utilisation de la technologie NFC	213
Envoyer ou recevoir des informations via NFC	215
Utiliser NFC sans NFC	216
La gestion des périphériques	217
Lister les périphériques disponibles	217
Interagir avec les périphériques	218
Conclusion	222

CHAPITRE 11

Le Windows Store 223

Le Windows Store	223
Une découverte facilitée	224
Publier son application sur le Store	226
Licence développeur	226
Procédure d'enregistrement et de validation	227
<i>Manifeste et capacités</i>	227
Soumission de l'application	229
Les principales règles à respecter	232
Augmenter la visibilité de votre application	233
Noter l'application	233
Faire télécharger l'application depuis un site web	234
Gestion des applications publiées	235

CHAPITRE 12

Accéder à Internet	237
Données applicatives	237
Données itinérantes d'une application	238
Sécurité	240
Accès à des services Internet personnalisés	241
<i>Utilisation de sockets</i>	242
<i>Service HTTP « brut »</i>	246
<i>Service REST et sérialisation JSON</i>	249
<i>Service WCF</i>	251
Accéder à des services standards - SkyDrive	251
Développer plus vite grâce aux services mobiles Windows Azure	254
<i>Création d'un service mobile</i>	255
<i>Accès aux données stockées sur Azure</i>	256

CHAPITRE 13

Windows Runtime avancé	259
Processus WinRT	259
Composants WinRT	261
Évolution des composants logiciels	261
Anatomie d'un composant WinRT	264
<i>Classe et interfaces</i>	264
<i>Modèle de threading</i>	266
<i>Métadonnées</i>	267
<i>Activation et fabriques</i>	270
<i>Système de types WinRT et projections</i>	273

Conclusion	279
-------------------------	------------

Index	281
--------------------	------------

Applications Windows Store

Ces dernières années ont vu naître de nombreuses innovations technologiques. Accessibles au plus grand nombre, ces dernières ont entraîné l'émergence de nouveaux usages. Il est désormais possible de se servir de matériel léger, autonome et connecté pour créer librement l'information et y accéder, où que l'on soit. De nouvelles façons d'interagir avec les applications sont maintenant largement répandues : interface tactile, géolocalisation, connectivité permanente et services en ligne ouvrent la voie à de nouveaux scénarios d'utilisation. Windows 8 a été ré-imaginé en intégrant ces évolutions dans son cœur, donnant naissance à un nouveau type d'applications : les applications Windows Store.

À l'heure de la rédaction de cet ouvrage, Windows 8 représente la toute dernière génération de Windows. Plus qu'une simple évolution du système d'exploitation de Microsoft, Windows a été ré-imaginé afin de fournir aux utilisateurs de meilleures performances, plus de flexibilité et une productivité améliorée.

Les applications jouent un rôle majeur dans cette démarche : alors que le système d'exploitation fournit des services transverses et la gestion des composants bas-niveau de la machine, c'est avec les applications que l'utilisateur réalise les tâches qui sont pour lui les plus importantes. La qualité des applications et la synergie entre ces dernières forment un tout avec le système d'exploitation et en définissent ainsi la richesse et la puissance.

Qu'est-ce qu'une application Windows Store ?

Bien plus qu'une évolution de la plate-forme de développement, les applications Windows Store sont un nouveau type d'applications Windows à part entière. Cette section décrit quelques-unes des différences les distinguant des applications « classiques », dites Bureau, utilisées jusqu'alors.

Une expérience utilisateur plus fluide

Windows a changé du point de vue de « l'expérience utilisateur » (on parle le plus souvent d'UX pour *User eXperience*), avec son interface utilisateur baptisée *Modern UI* et ses applications dédiées, les applications Windows Store. Les interfaces utilisateur de ce nouveau type d'applications se caractérisent par leur fluidité, par la mise en avant des contenus utiles, différents modes d'affichage, une grande lisibilité due à l'utilisation très limitée du chrome (contours), ou encore une exploitation prononcée de la typographie. Pour être cohérentes avec le système, les applications Windows Store se doivent d'être, elles aussi, rapides, fluides et modernes.

Le chapitre 2, consacré à l'ergonomie, présente de manière non technique les principaux aspects liés à l'expérience utilisateur des applications Windows Store.

Un cycle de vie plus contrôlé

Les applications Windows traditionnelles sont destinées à être exécutées sur le Bureau. Il s'agit de programmes généralement implémentés avec des technologies telles que MFC, Windows Forms ou encore WPF. Leur cycle de vie est relativement simple :

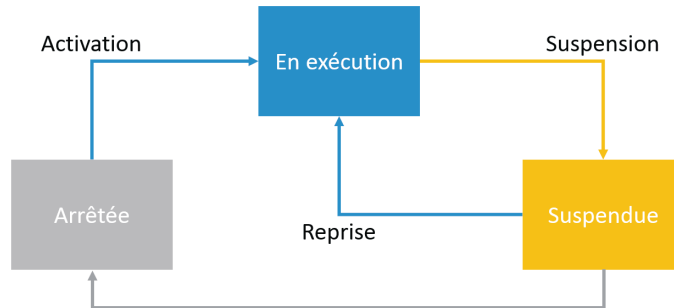
Figure 1-1
Cycle de vie d'une application
Windows classique



Les applications Bureau sont globalement libres de gérer leur propre cycle de vie et s'exécutent tant qu'elles ne sont pas explicitement fermées. Le système d'exploitation leur alloue les ressources nécessaires en fonction de leurs besoins et, sauf cas exceptionnels, n'a pas vocation à déterminer si une application doit être suspendue ou arrêtée.

Alors que le modèle lié aux applications Bureau a été conçu pour une utilisation en continu, les applications Windows Store ciblent des usages où l'interactivité et la mobilité ont un rôle prépondérant. Ces deux facteurs ont contribué à la conception d'un cycle de vie applicatif très différent de celui utilisé jusque-là, qui donne au système d'exploitation la possibilité de contrôler de manière plus fine le cycle de vie des applications (voir figure 1-2).

Figure 1-2
Cycle de vie d'une application Windows Store



Un nouvel état, « suspendu », a donc été ajouté au cycle de vie classique. Dans cet état, généralement atteint lorsque l'application n'est plus affichée à l'écran, l'application réside toujours en mémoire mais son code ne s'exécute plus. Le système d'exploitation peut également décider d'arrêter (fermer) une application si les ressources système viennent à manquer, et ce sans intervention de l'utilisateur. L'application est avertie par le système d'exploitation lorsque son état change et dispose alors d'un temps limité pour, par exemple, sauvegarder ses données applicatives.

Figure 1-3
Applications en mode remplissage (à gauche) et ancré (à droite)



Les applications affichées à l'écran, deux au maximum, se partagent la quasi-totalité des ressources système disponibles, tandis que les autres applications Windows Store non affichées sont suspendues, ne sollicitent pas de ressource processeur et peuvent être supprimées de la mémoire vive si le système d'exploitation en voit la nécessité. Ce modèle met en avant les principes de conception liés à la fluidité et à la mobilité : les ressources sont dédiées aux applications affichées à l'écran afin de leur assurer une interactivité et une fluidité maximales, tout en optimisant l'utilisation de la batterie. Plus encore que pour les applications Bureau, le système d'exploitation joue un rôle central dans la gestion du cycle de vie des applications Windows Store, car il impose aux applications de passer d'un état du cycle à un autre.

Les deux types d'applications proposées par Windows, Bureau et Windows Store, diffèrent donc grandement dans la manière dont sont gérés leurs cycles de vie respectifs. Ces différences ne sont pas que techniques : elles influent également sur l'utilisation qui pourra être faite des applications. Certains usages pourraient nécessiter une exécution en tâche de fond et auraient alors plus de sens comme applications Bureau, tandis que d'autres, nécessitant des spécificités réservées aux applications Windows Store (gestion des modes d'affichage, mode de distribution) et une meilleure autonomie de la batterie, pourraient être implémentés en tant qu'applications Windows Store.

Distribution et déploiement

Le mode de distribution des applications Windows Store diffère considérablement du mode de distribution classique : en accord avec l'évolution des usages, et mis à part le cas particulier du poste développeur, leur déploiement se fait de manière centralisée, que ce soit pour le grand public ou en entreprise. Il existe trois manières de déployer une application Windows Store, décrites ci-après.

Distribution via le Windows Store

Sans aucun doute la façon la plus répandue de déployer une application, la distribution via le Windows Store de Microsoft sert à publier et à monnayer un programme à grande échelle. Le chapitre 11 décrit en détail le processus de soumission d'une application, ainsi que les nombreuses possibilités offertes par ce mode de distribution.

Déploiement interne en entreprise

Le Windows Store public n'est pas le seul mode de publication. En effet, une entreprise peut déployer une application en interne au sein de son parc informatique, sans avoir à passer par le Windows Store public. De nombreuses applications métier n'ont en effet pas vocation à être distribuées publiquement. Autre aspect important : les déploiements internes sont intégralement sous le contrôle de l'entreprise. Ce n'est

pas le cas des applications distribuées sur le Windows Store public, qui doivent d'abord être validées par Microsoft avant d'être publiées.

Déverrouillage pour les développeurs

Le développeur d'applications a bien entendu besoin de déployer localement l'application qu'il développe. C'est le déverrouillage qui lui permet de le faire, moyennant l'ouverture d'un compte développeur Microsoft et le déploiement des certificats adéquats sur son poste. Le chapitre 11 décrit les modalités d'obtention d'un compte développeur Windows Store.

Sécurité

Au fil des années, la technologie prend une place grandissante dans notre quotidien. En conséquence, les problématiques liées à la sécurité et au respect de la vie privée sont devenues une priorité de premier ordre dans le développement logiciel. Ces considérations sont prises en compte dans Windows en donnant aux utilisateurs la possibilité de connaître et de gérer les capacités des logiciels dédiés à la nouvelle plate-forme. Le développeur de l'application se doit de déclarer les besoins de son application (accès aux contacts, géolocalisation, Internet et bien d'autres) lors de la publication du logiciel sur le Windows Store. Afin de valider la mise à disposition de l'application sur le Store, Microsoft vérifie que les fonctionnalités (API) utilisées sont bien celles déclarées par le développeur. Ces fonctionnalités sont ensuite présentées à l'utilisateur sur la page dédiée à l'application au sein du Windows Store. Lors du premier lancement de l'application, l'utilisateur devra donner explicitement son consentement sur l'utilisation de ces dernières. Les différentes fonctionnalités ont été identifiées en amont par Microsoft, sur des critères généralement liés aux aspects financiers (coûts de connexion réseau), à la protection de la vie privée et de la machine de l'utilisateur.

Il est demandé aux développeurs de prévoir et gérer dans leur code le cas où l'utilisateur refuse l'utilisation de certaines capacités. Une application peut par exemple continuer à fonctionner même si l'utilisateur refuse la géolocalisation.

La déclaration des besoins de l'application, quant à elle, se fait par le biais d'un fichier XML, dit `manifest`, que le développeur peut éditer manuellement ou par le biais d'une interface graphique dans Visual Studio.

Le chapitre 11 décrit en détail la déclaration de ces fonctionnalités.

Performances et autonomie

Les ambitions en termes d'expérience utilisateur et les contraintes de la mobilité doivent s'allier de manière élégante.

Comme mentionné dans la section traitant du cycle de vie applicatif, au maximum deux applications Windows Store peuvent s'exécuter simultanément. Cette contrainte permet au système d'exploitation à la fois de fournir une expérience immersive et de concentrer les ressources machines sur les applications que l'utilisateur est réellement en train d'utiliser. L'effet immédiat est une fluidité et un confort d'utilisation optimal, ne consommant les ressources que lorsque les applications présentes à l'écran en ont besoin.

Les applications Windows Store savent bien entendu exécuter des processus en tâches de fond. Cependant, ces dernières sont strictement supervisées par le système d'exploitation afin d'optimiser l'usage des ressources, prolongeant l'autonomie du système. Windows alloue un temps d'exécution et un quota de ressources, notamment CPU ou réseau, pour les exécuter : celles-ci peuvent à un moment donné être regroupées afin que le processeur reste en veille sur de plus longues périodes, et pour rassembler l'exécution des tâches quand celui-ci est réactivé.

Le rendu visuel et la fluidité sont des éléments essentiels de toute expérience utilisateur de qualité. Toutes les applications Windows Store, quelle que soit la technologie choisie pour leur implémentation, bénéficient de l'accélération graphique matérielle. Les processeurs graphiques spécialisés sont en charge de l'affichage, tandis que le processeur principal traite en parallèle des données non graphiques. La collaboration de ces deux éléments matériels est transparente pour le développeur et conduit à des applications performantes et généralement moins consommatrices en énergie.

Windows 8 et Windows RT

La sortie de la dernière version de Windows arrive avec une nouveauté majeure : une toute nouvelle déclinaison du système d'exploitation nommée Windows RT.

Windows 8 sur processeurs x86 et x64

Succédant au très populaire Windows 7, la version 8 du système cible les processeurs x86 et x64. Elle en assure la compatibilité ascendante, tant pour le parc logiciel que matériel, et met à disposition des fonctionnalités comme l'intégration dans des domaines d'entreprises ou la virtualisation. Windows 8 est disponible en trois éditions : standard, Professionnelle ou Entreprise.

Windows RT pour les processeurs ARM

Cette version de Windows est conçue pour fonctionner sur processeurs ARM, à ce jour plus économes en énergie et plus abordables que les processeurs basés sur les architectures x86 ou x64. Pour l'utilisateur classique, l'usage de Windows RT est très similaire à celui de Windows 8 : même interface graphique et même ergonomie. Une exception notable existe cependant entre les deux versions : les applications Bureau tierces ne peuvent s'exécuter sous Windows RT. En effet, ces dernières sont à ce jour toutes compilées pour les architectures x86/x64, les rendant incompatibles avec les processeurs ARM. Certaines applications Bureau très spécifiques et fournies par Microsoft existent cependant et sont livrées directement avec Windows RT, comme l'Explorateur de fichiers, Internet Explorer ou Office 2013. Il est à noter que, contrairement à Windows 8, Windows RT ne peut être acheté séparément et sera toujours livré préinstallé sur un appareil.

Les applications Windows Store sont les seules qui soient compatibles avec les deux versions du système d'exploitation et elles sont compilées par défaut pour les trois architectures x86, x64 et ARM. L'audience des applications Windows Store est donc très large car elle regroupe tous les utilisateurs de Windows 8 et Windows RT.

ATTENTION Ne pas confondre Windows RT et WinRT

Afin d'éviter toute confusion, il est important de noter que Windows RT est la version du système d'exploitation dédiée aux architectures ARM, tandis que l'abréviation WinRT représente le Windows Runtime, qui est la plate-forme technique sur laquelle reposent les applications Windows Store. Le Windows Runtime est détaillé au chapitre 13.

Une plate-forme de développement

Windows est accompagné historiquement d'une plate-forme de développement riche et variée. Le développeur d'applications dispose d'un grand choix d'outils de qualité, quel que soit l'archétype de ses programmes : client riche, web, mobile ou service. Les applications Windows Store ne dérogent pas à la règle et peuvent être implémentées par des développeurs de tous horizons en se basant sur des technologies éprouvées telles que Visual Studio, .NET, C++, XAML ou encore HTML5 et JavaScript.

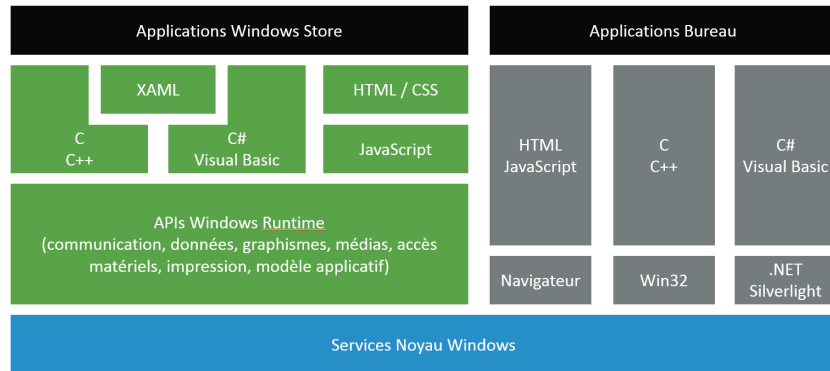
Tout comme le système d'exploitation sur lequel elle s'appuie, la plate-forme de développement soutenant les applications Windows Store a été complètement repensée, en capitalisant sur le savoir-faire de Microsoft.

Des API accessibles avec le Windows Runtime (WinRT)

Le Windows Runtime est la plate-forme sur laquelle s'appuient les applications Windows Store. Il s'agit pour Microsoft de fournir aux développeurs des API conçues pour faciliter le développement d'applications natives performantes, mobiles, contextuelles, fluides et sécurisées. Un soin tout particulier a également été apporté à l'ouverture de ces API : il est possible de les utiliser depuis de nombreuses technologies telles que le code natif C++, le code managé .NET ou encore JavaScript.

Dans les coulisses, le Windows Runtime et ses API sont implémentés en C++ en se basant sur l'évolution d'un composant fondamental et historique de Windows : le Component Object Model ou COM (dont la première version date de 1993 !). Sa remarquable flexibilité rend possible et naturelle l'utilisation des nouvelles API depuis des technologies aussi variées que JavaScript, C# ou C++, et ce grâce à un mécanisme de projection transparent. Le développeur peut également écrire ses propres composants Windows Runtime de bas niveau, réutilisables depuis n'importe laquelle des technologies citées précédemment. Pour l'utilisateur, la technologie d'implémentation n'a finalement pas d'importance : à l'utilisation, il s'agira dans tous les cas d'une application Windows Store.

Figure 1-4
Plate-forme applicative



Voici un exemple d'appel Windows Runtime (mise à jour d'une vignette secondaire) dans trois langages :

JavaScript

```
var tileTextAttributes = tileXml.getElementsByTagName("text");
tileTextAttributes[0].appendChild(tileXml.createTextNode("Mon texte !"));
var updater =
  notifications.TileUpdateManager.createTileUpdaterForSecondaryTile(appbarTileId);
```


C++

```
XmlNodeList^ tileTextAttributes = tileXml->GetElementsByTagName("text");  
tileTextAttributes->Item(0)->InnerText = "Mon texte !";  
TileUpdater^ secondaryTileUpdater =  
TileUpdateManager::CreateTileUpdaterForSecondaryTile(appbarTileId);
```

C#

```
XmlNodeList tileTextAttributes = tileXml.GetElementsByTagName("text");  
tileTextAttributes[0].InnerText = "Mon texte !";  
TileUpdater secondaryTileUpdater =  
TileUpdateManager.CreateTileUpdaterForSecondaryTile(appbarTileId);
```

Le chapitre 13 détaille le fonctionnement interne du Windows Runtime, et notamment le mécanisme de projection.

HTML5 et JavaScript

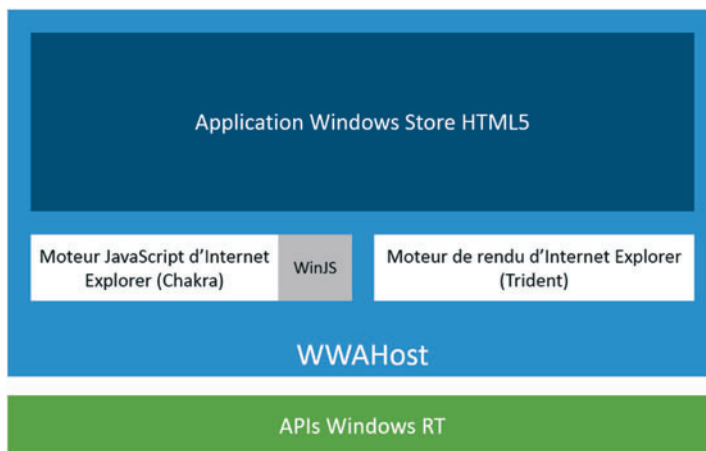
La possibilité de recourir à des technologies traditionnellement réservées au Web pour le développement d'applications natives est un des éléments remarquables de la nouvelle plate-forme Windows. Les applications Windows Store implémentées en HTML et JavaScript s'exécutent au sein d'un processus nommé [WWAHost](#) qui fournit aux applications hébergées un environnement basé sur les fondations d'Internet Explorer 10, assurant au développeur une compatibilité complète avec les standards de ce navigateur pour les technologies suivantes :

- HTML5 ;
- CSS3 ;
- SVG ;
- JavaScript/ECMAScript.

Le moteur de rendu d'Internet Explorer est directement utilisé pour afficher le contenu HTML5/CSS3/SVG ; ainsi, les applications bénéficient pleinement de l'accélération matérielle par le biais de DirectX. En d'autres mots, le ou les processeur(s) graphique(s) est(sont) mis à contribution lors de l'exécution des applications HTML5, résultant en des performances d'affichage optimales.

Les composants déjà développés avec ces technologies web peuvent ainsi être réutilisés au sein d'une application Windows, souvent sans changement de code. Une disparité fondamentale existe cependant entre une application web et une application Windows locale : son contexte d'exécution.

Figure 1-5
Environnement d'exécution
WWAHost



Ce dernier, appelé contexte local, semble parfois déroutant dans un premier temps pour le développeur web : l'application Windows Store ne s'exécutant pas au sein d'un serveur web, le concept de session locale est différent de celui d'une session web et l'application dispose d'un cycle de vie sans équivalent sur le Web. Les principales différences entre les contextes web et locaux sont les suivantes :

- Le contexte local donne accès à certaines fonctionnalités spécifiques de Windows, et tout particulièrement aux API du Windows Runtime, par le biais de la bibliothèque Windows pour JavaScript (WinJS). Cette dernière autorise les applications HTML5 à communiquer directement avec le système d'exploitation, donnant ainsi potentiellement accès aux capacités physiques de la machine, aux contrats Windows, à la gestion des manipulations tactiles, au multithreading et à bien d'autres fonctionnalités. Ces ajouts ouvrent la voie à des scénarios jusqu'alors impossibles avec les standards que sont HTML, CSS ou JavaScript.
- Le contexte local ne supporte pas les extensions ou plug-ins (par exemple Flash ou Silverlight).
- Les accès aux ressources machine, notamment l'accès réseau, sont soumis aux déclarations des besoins de l'application, décrites dans le chapitre 11.
- Le développement d'applications HTML5 est une solution particulièrement séduisante pour les développeurs disposant d'une expérience dans le développement web, car elle allie des standards reconnus et maîtrisés aux capacités offertes par le système d'exploitation Windows via le Windows Runtime. Cette intégration de HTML5 et de JavaScript au Windows Runtime crée implicitement une exception technique qui peut surprendre : les applications de ce type, bien qu'utilisant les standards du Web sont intimement liées à Windows et ne peuvent pas s'exécuter au sein d'un navigateur web. Ce nouveau mode de développement s'apparente donc plus à une ouverture supplémentaire offerte par Microsoft dans

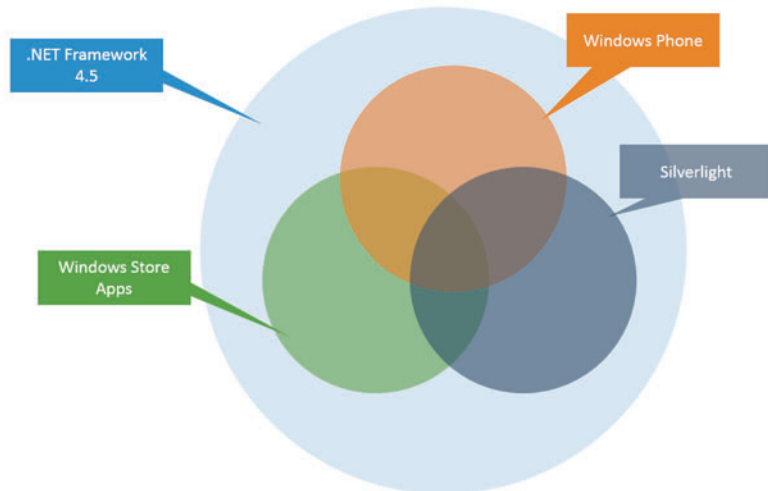
le cadre de la ré-exploitation d'un savoir-faire, plutôt que dans celui d'une réutilisation du code entre application native et application web.

.NET et XAML (eXtensible Application Markup Language)

Les développeurs .NET, en particulier ceux familiers avec WPF et Silverlight, peuvent réutiliser leurs compétences afin de développer des applications Windows Store. En effet, Windows 8 et Windows RT leur offrent la possibilité d'écrire leurs applications en se basant sur le framework .NET 4.5 (Visual C# et Visual Basic) couplé à un moteur de rendu XAML.

Depuis sa version 3.5, le framework .NET cible différentes plates-formes par le biais d'un mécanisme de profils. Ces derniers définissent des sous-ensembles du framework complet, afin de donner ou non l'accès à certaines API en fonction de la plate-forme cible. Par exemple, le profil .NET standard donne accès aux API disponibles sur les postes de travail Windows, tandis que le profil Windows Phone donne accès à celles s'exécutant sur les smartphones équipés du système d'exploitation de Microsoft.

Figure 1-6
Profils .NET



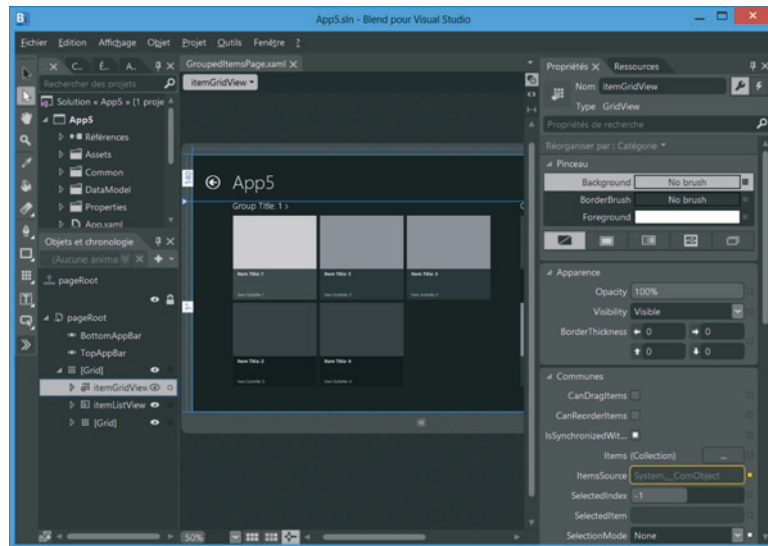
Le profil Windows Store donne accès à un sous-ensemble du framework .NET ainsi qu'à des API spécifiques au Windows Runtime. Le mécanisme de projection rend également certaines API du Runtime accessibles en .NET, facilitant leur utilisation par le développeur et son application.

XAML (eXtensible Application Markup Language) est un langage de description applicatif conçu par Microsoft et basé sur le standard XML. Également utilisé pour décrire des workflows dans Workflow Foundation, son usage le plus courant consiste

en la description d'interfaces graphiques. C'est dans ce cadre que le XAML joue un rôle important dans le développement d'applications Windows Store .NET ou C++ car il permet aux développeurs de décrire des interfaces graphiques complexes en bénéficiant des avantages suivants :

- Basé sur du XML, ce langage autorise les développeurs et les intégrateurs graphiques à utiliser des outils spécialisés comme Blend pour construire des interfaces graphiques de manière intuitive et visuelle.
- La séparation est propre entre la description des écrans (XAML) et le code (C#/VB/C++) qui les contrôle.
- À l'exécution, l'utilisation d'un moteur de rendu performant réécrit en code natif utilisant DirectX transforme le XAML en primitives graphiques vectorielles comprises par le GPU.

Figure 1-7
Blend pour Visual Studio permet d'éditer le XAML en WYSIWYG



.NET permet également l'implémentation de composants Windows Runtime bas niveau en C# ou Visual Basic. Ces composants, au même titre que ceux écrits en C++, sont réutilisables depuis les autres langages reconnus par le Windows Runtime.

C++ et XAML

Avec le Windows Runtime, le langage C++ revient sur le devant de la scène ; il s'agit d'ailleurs de la technologie utilisée pour implémenter le Windows Runtime lui-même. La couche de projection est donc quasi inexistante, d'où une interaction

encore plus transparente avec le Runtime. Le développement d'applications en C++ dispose d'outils spécifiques le rendant particulièrement intéressant dans certains cas :

- Techniquement, il s'agit de la plate-forme de développement la plus performante. Bien que les applications s'exécutent en grande majorité de manière parfaitement fluide quelle que soit la technologie de développement choisie, les applications en code natif C++ tirent au maximum parti des capacités de la machine.
- Certaines API ne sont disponibles que pour ce langage, notamment celles liées à DirectX. Elles sont de plus combinables avec le moteur de rendu XAML, alliant ainsi performance et productivité lors du développement d'interfaces graphiques complexes.
- Le C++ permet la réutilisation de code portable (propre à l'application ou appartenant à des bibliothèques Open Source) au cœur de l'application, la couche de présentation « supérieure » étant nécessairement liée à Windows.

Il est à noter que dans sa version 2012, Visual C++ implémente l'essentiel de la norme C++ 11, qui rend ce langage plus accessible, notamment au niveau de la gestion de la mémoire grâce aux notions de `shared_ptr<T>` ou `unique_ptr<T>`.

Le standard C++ implémenté dans Visual Studio 2012 est décrit dans la norme ISO/CEI 14882:2011.

Considérations pour les développeurs d'applications Bureau

Pour le développeur ayant déjà une expérience dans le développement d'applications clientes Windows (C++, WPF ou même Silverlight), les privilèges octroyés aux applications Windows Store peuvent sembler extrêmement réduits. Ces dernières s'exécutent en effet dans des contextes contrôlés restrictifs (dits *sandbox*), où les fonctionnalités susceptibles d'avoir un effet sur la sécurité et les ressources sont déclarées à l'avance dans le manifeste de l'application. Voici quelques exemples d'actions qu'une application Windows Store ne pourra pas effectuer :

- accéder sans consentement explicite de l'utilisateur à des dossiers ou fichiers du disque local ;
- lire ou écrire en base de registre ;
- interagir avec l'interface utilisateur hors du périmètre de la zone occupée par l'application elle-même ;
- avoir accès à des informations système locales, journal des événements, processus en cours d'exécution, etc.

Comme mentionné précédemment, le cycle de vie applicatif est lui aussi particulier, ce qui semble complexifier le développement. Il faut cependant garder en tête que ces restrictions garantissent un niveau élevé de sécurité et une confiance accrue de la part de l'utilisateur, tout en laissant le système d'exploitation exercer un contrôle optimal sur l'utilisation des ressources. Pour ces raisons, de nombreux scénarios auront davantage de sens s'ils sont implémentés en tant qu'applications Bureau, tandis que d'autres tireront un grand bénéfice à profiter d'une intégration avec le Windows Runtime.

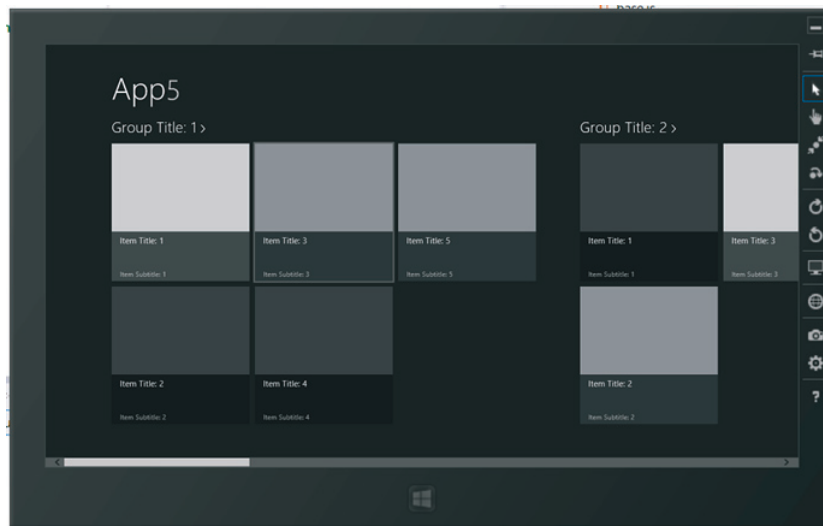
Outillage du développeur

Microsoft a toujours fourni des outils de développement en phase avec l'évolution de ses plates-formes. Les applications Windows Store ne dérogent pas à cette règle et la gamme Visual Studio a été largement enrichie afin de fournir une plate-forme de développement complète pour les applications Windows Store. Mieux encore, certaines versions de ces outils sont disponibles gratuitement, pour que le plus grand nombre, aussi bien professionnels que développeurs occasionnels, soit en mesure de développer des applications Windows Store.

Le SDK Windows 8

Élément fondamental, le SDK Windows 8 (Software Development Kit) fournit les outils nécessaires au développement d'applications Windows Store. En plus de nombreuses bibliothèques, le SDK propose également un simulateur Windows 8 pour tester les applications Windows Store dans différentes résolutions et configurations et pour simuler des événements tactiles sur des machines n'étant pas dotées de ces capacités matérielles.

Figure 1-8
Le simulateur Windows 8

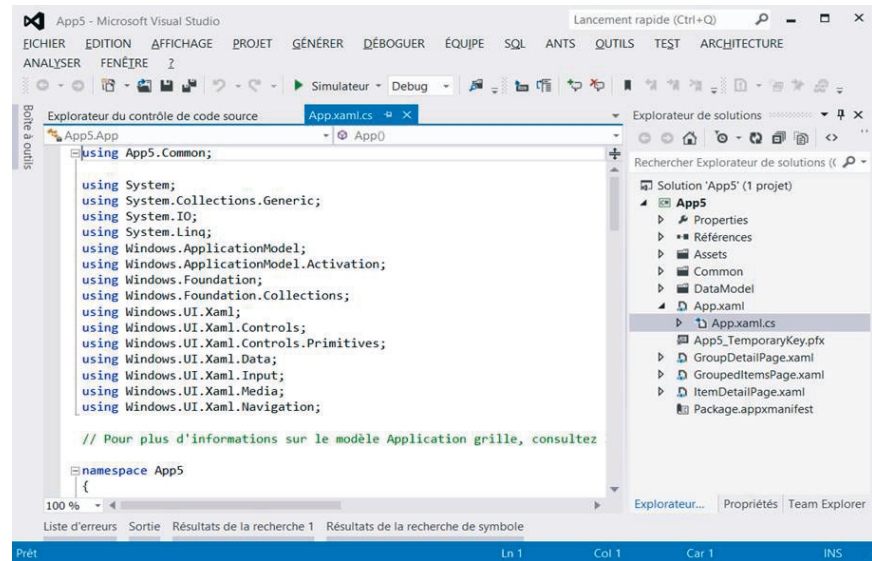


Le SDK étant intégré par défaut à Visual Studio, son installation indépendante est donc un geste rare généralement réservé à la mise en place de machines dédiées à l'intégration continue.

Visual Studio 2012

La nouvelle version de l'IDE emblématique de Microsoft permet évidemment de développer des applications Windows Store, mais propose également des centaines d'autres nouveautés, dont une nouvelle interface graphique fluide et épurée, des outils de recherche sophistiqués ou encore de tous nouveaux outils destinés à faciliter le développement d'interfaces graphiques XAML et HTML5.

Figure 1–9
Visual Studio 2012



Blend pour Visual Studio

Expression Blend est un outil familier aux développeurs d'applications WPF et Silverlight. Outil par excellence de conception d'interfaces graphiques XAML, sa nouvelle déclinaison nommée Blend pour Visual Studio sert à concevoir des applications Windows Store aussi bien en XAML qu'en HTML5.

Expression Blend reste disponible pour les applications WPF et Silverlight, tandis que Blend pour Visual Studio est une édition distincte du logiciel, livrée en standard avec Visual Studio et ne ciblant que les applications Windows Store.

MSDN, la référence documentaire

MSDN (Microsoft Developer Network) est la référence pour tous les développeurs sur plate-forme Microsoft. Il regroupe des centaines de milliers de pages de documentation traitant des différentes technologies de développement Microsoft, dont

une large section dédiée aux applications Windows Store. Des chapitres entiers sont consacrés à l'expérience utilisateur et à l'ergonomie, et des milliers de pages décrivent en détail chaque API du Windows Runtime.

RESSOURCES MSDN pour les applications Windows Store

► <http://msdn.microsoft.com/fr-FR/windows/apps/>

Composants tiers

De nombreux éditeurs tiers se lancent dans la commercialisation de composants pour les applications Windows Store. Il s'agit généralement de fournir des bibliothèques et API facilitant l'intégration des contrôles utilisateur (graphiques, grilles, panoramas...) ou encore de traiter/afficher différents formats de fichiers spécifiques.

Bien que la qualité des composants disponibles sur le marché soit inégale, il est tout de même recommandé d'étudier l'intégration de tels composants dans les applications, car elle résulte parfois en une considérable réduction du temps de développement global d'un logiciel.

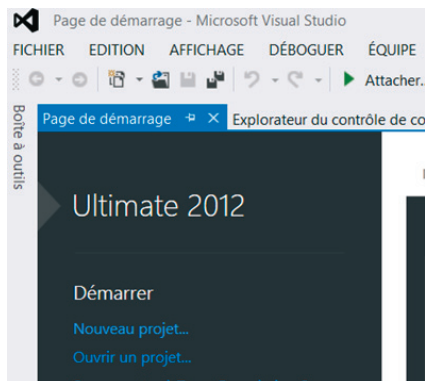
Une première application

Cette section décrit pas à pas la mise en œuvre des outils mentionnés afin de créer une première application Windows Store.

XAML

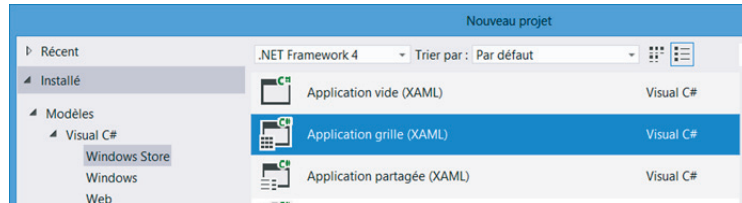
- 1 Démarrer Visual Studio 2012. Sur la partie gauche de l'écran, sélectionner *Nouveau projet...*

Figure 1-10



- 2 Dans les modèles préinstallés, déplier l'arborescence de gauche. Sous l'en-tête *Visual C#* ou *Visual Basic* en fonction du langage cible, sélectionner la section dédiée aux projets *Windows Store*. Sélectionner *Application grille (XAML)*.

Figure 1-11



- 3 Nommer le projet dans la zone basse de l'écran et cliquer sur *OK*. Visual Studio crée alors les fichiers nécessaires à la compilation d'un projet d'application Windows Store basé sur .NET et XAML.
- 4 Visual Studio et Blend sont des outils conçus pour être utilisés de concert. Depuis l'explorateur de solution, un clic droit sur un des fichiers XAML affiche un menu contextuel donnant la possibilité de l'ouvrir dans Blend. Blend pour Visual Studio démarre et le fichier XAML sélectionné s'affiche dans la surface de conception, donnant accès à une palette impressionnante d'outils qui permettent aux développeurs et aux intégrateurs de parfaire les aspects visuels des applications.

Figure 1-12

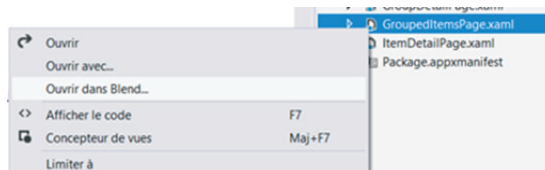
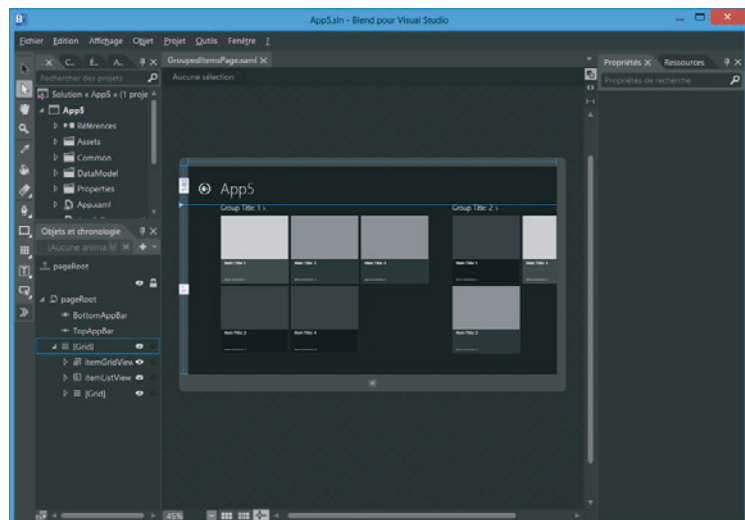


Figure 1-13



- 5 Revenir dans Visual Studio 2012. L'application telle que créée par l'assistant est prête à s'exécuter. Le bouton de lancement laisse choisir dans quel contexte l'application sera déboguée : dans l'environnement Windows de la machine locale, sur une machine distante, ou au sein de l'émulateur du SDK. Sélectionner l'un d'eux pour démarrer cette première application.

Figure 1-14

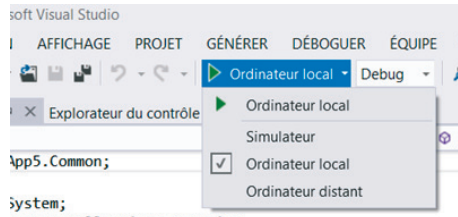
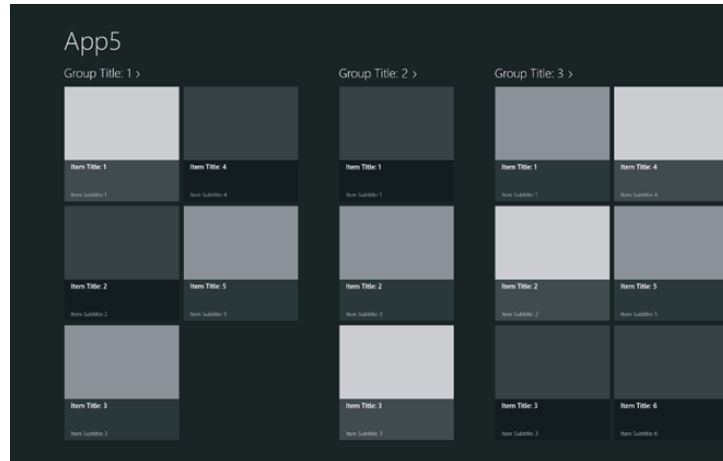


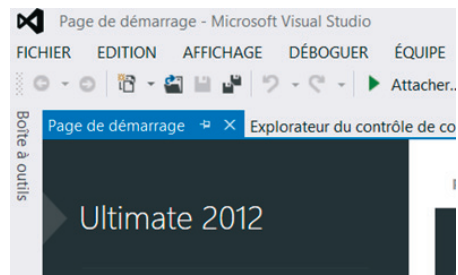
Figure 1-15



JavaScript et HTML

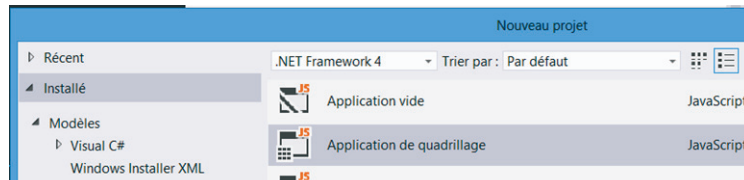
- 1 Démarrer Visual Studio 2012. Sur la partie gauche de l'écran, sélectionner *Nouveau projet...*

Figure 1-16



- Dans les modèles préinstallés, déplier l'arborescence de gauche. Sous l'en-tête *JavaScript*, sélectionner la section dédiée aux projets *Windows Store*. Sélectionner *Application de quadrillage*.

Figure 1-17



- Nommer le projet dans la zone basse de l'écran et cliquer sur *OK*. Visual Studio crée alors les fichiers nécessaires à la compilation d'un projet d'application Windows Store basé sur JavaScript et HTML.

Visual Studio et Blend sont des outils conçus pour être utilisés de concert. Depuis l'explorateur de solution, un clic droit sur un des fichiers HTML affiche un menu contextuel donnant la possibilité de l'ouvrir dans Blend. Blend pour Visual Studio démarre et le fichier HTML sélectionné s'affiche dans la surface de conception, donnant accès à une palette impressionnante d'outils qui permettent aux développeurs et aux intégrateurs de parfaire les aspects visuels des applications.

Figure 1-18

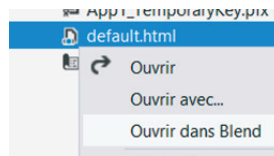
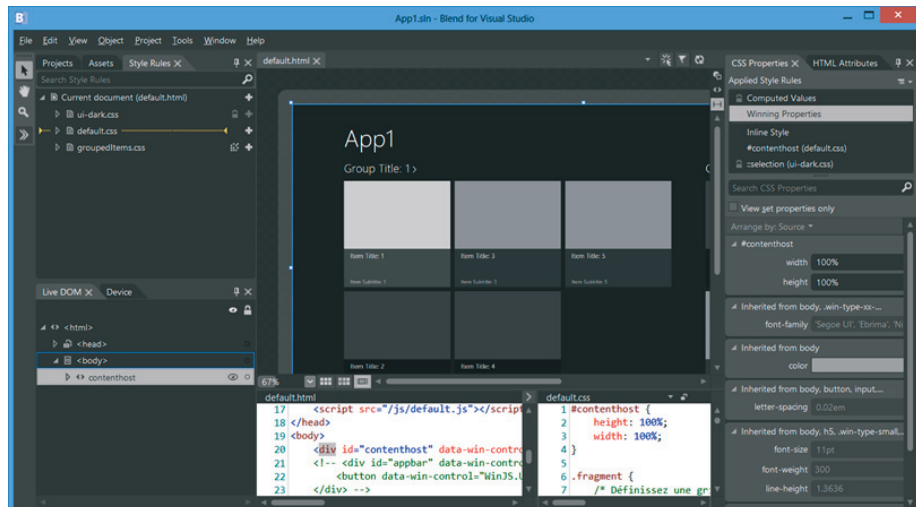


Figure 1-19



- 4 Revenir dans Visual Studio 2012. L'application telle que créée par l'assistant est prête à s'exécuter. Le bouton de lancement laisse choisir dans quel contexte l'application sera déboguée : dans l'environnement Windows de la machine locale, sur une machine distante, ou au sein de l'émulateur du SDK. Sélectionner l'un d'eux pour démarrer cette première application.

Figure 1-20

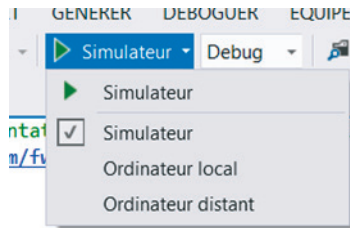
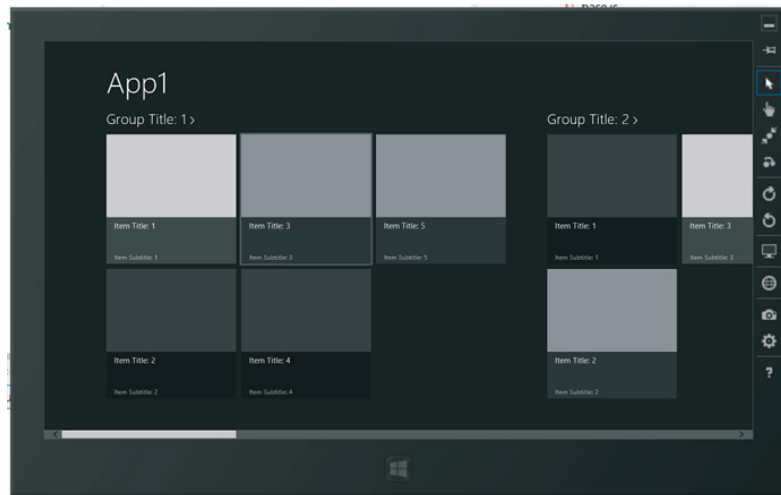


Figure 1-21



Comme l'ont montré les étapes précédentes, la création d'applications Windows Store est facilitée par les fonctionnalités et les outils de développement fournis par Visual Studio. En quelques clics seulement, il est ainsi possible d'obtenir une application Windows Store fonctionnelle !

Conclusion

Windows crée sa petite révolution, notamment avec son nouvel écosystème applicatif qui permet d'installer une galaxie d'applications et de les faire interagir entre elles pour apporter de nouvelles fonctionnalités et ainsi enrichir l'expérience utilisateur.

Pour le développeur, les opportunités sont énormes. La double interface de Windows va immédiatement assurer la présence de cet environnement pourtant nouveau sur un nombre impressionnant d'ordinateurs et, grâce au Windows Store, le développeur proposera ses réalisations à tous ces utilisateurs de manière très simple.

Dans ce livre, nous avons abordé la programmation d'applications pour le Windows Store en combinant différents objectifs.

Tout d'abord, l'API WinRT est très riche et il est important d'en connaître les principaux concepts et leur mise en œuvre. Nous avons vu comment construire une application en gérant ses différents états visuels, puis présenté la gestion de ses données (à la fois en termes d'affichage et de persistance) et comment intégrer une application dans un environnement plus large (interfaces avec les autres applications, avec les périphériques et enfin avec les principaux types de services Internet).

Comme un tel ouvrage ne peut être exhaustif, nous avons le plus souvent possible pris un peu de recul et expliqué la logique qui se cache derrière les concepts présentés, afin que le développeur soit en mesure d'appréhender aussi facilement les API qui n'ont pas été abordées.

L'avantage d'une refonte complète d'une plate-forme de programmation est qu'elle gagne ainsi en cohérence. Contrairement à la vénérable API Win32, dont chaque sous-ensemble est le reflet de l'époque à laquelle il a été ajouté, l'API WinRT est homogène et le lecteur de cet ouvrage ne se sentira jamais perdu lorsqu'il abordera un nouveau concept.

Enfin, nous avons insisté sur le respect des bonnes pratiques. Ceci s'applique à la fois à la vision externe des applications, mais aussi à leur implémentation. L'aspect externe est fondamental, car il est au contact de l'utilisateur qui abandonnera sans pitié toute application trop compliquée à utiliser ou peu réactive car, dans de nombreux cas, le Windows Store proposera d'autres applications aux fonctionnalités assez proches... Les évaluations laissées par des utilisateurs mécontents risquent d'hypothéquer fortement l'avenir d'une application. L'aspect interne ne doit pas être négligé pour autant, car le développeur y gagnera au final en termes de productivité, surtout lorsqu'il devra mettre son application à jour pour ajouter une fonction rendue indispensable par l'apparition de nouveaux services externes ou de nouvelles applications concurrentes. Rien n'est plus difficile que de se replonger dans un code mal ficelé après quelques mois d'éloignement.

Bon développement !

Alain, Louis-Guillaume et Luc

Index

.NET 15

A

- accès aux données 73
- accès aux données distantes 81
- activation d'une application suspendue 117
- ADO.NET 73, 81
- animation
 - dépendante 182
 - état visuel (XAML) 184, 185, 187
 - indépendante 181
 - libre 180
 - liste animée 178
 - rôle 172
 - storyboard (XAML) 182
 - transition CSS 190, 191
 - transition de contenu 175
 - transition de thème (XAML) 173
 - type 173
- API de lancement d'application 166
- application Bureau 6, 17, 30
- approche orientée objet 52
- association application
 - charger le fichier 162
- association applicative 160, 219
 - de fichier 160
 - de protocole 164
 - utiliser le protocole 165
- async 96
- asynchronisme 89

B

- badge
 - mise à jour 129
- bibliothèque d'interface 82
- Blend 19

C

- C#
 - sauvegarde de l'état 111
- C++ 16
 - liaison de données 64
 - sauvegarde de l'état 115
- capteur 211
- charme
 - paramètres 156, 157, 158
 - partage 140, 144, 146, 147
 - périphériques 216
 - recherche 150, 153, 155
- composant COM 12, 262
- composant tiers 20
- concept de navigation 33
 - hiérarchique 34
 - hybride 35
 - linéaire 34
- contexte local 14
- contrat 139, 219
 - paramètres 140
 - partage 140
 - PlayTo 140
 - recherche 140
 - sélection de fichier 140
- cycle de vie 6, 105, 106

D

- DataBinding 52
- DataTemplate 62
- délai de sauvegarde 107
- déploiement interne 8
- design language 27
- déverrouillage développeur 9
- données applicatives 237
 - itinérantes 238

E

- écran d'accueil 106
- écran type
 - page 37
 - panoramique 39
- élément d'interface 40
 - barre d'applications 44
 - icône 44
 - vignette dynamique 40
- ergonomie 25
- état d'affichage 31
 - ancré 32
 - plein écran 31
 - remplissage 32
- expérience utilisateur 6

G

- géolocalisation 210
 - accès continu aux coordonnées 209
 - accès ponctuel aux coordonnées 207

H

- HTML5 13

I

- IAsyncOperation 93
- interface avec les capteurs 205
 - géolocalisation 205
- interface tactile 193, 194
 - détecter les capacités machine 197
 - événement de manipulation (XAML) 200
 - GestureRecognizer 198
 - mouvement 198
 - pointeur 195
- interface utilisateur
 - réactive 89
- Internet 237
 - accéder aux données d'un service mobile 256
 - créer un service mobile 255
 - service HTTP 246
 - service REST 249
 - service standard SkyDrive 251
 - service WCF 251
 - services mobiles Windows Azure 254

J

- JavaScript 22
 - sauvegarde de l'état 114

L

- langage d'interaction tactile 35
 - manipulation 36
 - mouvement 35
- langage visuel 47
- lecture/écriture dans un fichier 74
 - fichier texte 76
 - fichier XML 78
 - FileIO 78
- liaison de données 52, 96
 - HTML5 68
 - liaison de listes 61
 - liaison simple 55
- LINQ 86

M

- Media Center 27
- méthode longue
 - durée 93
- modèle EDMX 86
- Modern UI 6, 26, 123
- MSDN 19
- ms-view-state 191
- MVVM 54

N

- NDEF (NFC Data Exchange Format) 213, 216
- NFC (Near Field Communication) 216
- NFC (Near Field Communication) 213
 - envoyer/recevoir des informations 215
- notification 42
 - vignette 43

O

- OData 86

P

- performance 10
- périphériques
 - événement de 220
 - gérer 217
 - interagir 218

- lister les périphériques disponibles 217
- philosophie de design 26, 27
- plate-forme de développement 11
- pool de threads 101

R

- reprise des données 116
- requête asynchrone
 - C++ 103
 - callback 87
 - JavaScript 102
- ressources machine 14
- REST 86

S

- sandbox 17
- sauvegarde d'état 109
- scénario d'utilisation 30
- SDK Windows 8 18
- sécurité 9, 150, 240
- sérialisation 242
 - JSON 79
 - XML 79
- service Internet personnalisé 241
- service WCF simple 81
- socket 242
- suspendu 7, 108, 109
- SuspensionManager 112

T

- tâche 99

V

- vignette dynamique 123
 - décrire en XML 128
 - mettre à jour 128
 - organisation 125
 - taille 128
 - transmettre le contexte 132
 - vignette secondaire 130

- Visual Studio 19

W

- WCF Data Services 86
- Windows Phone 27
- Windows Push Notification Service 133
- Windows Runtime 12, 92, 259
 - composant WinRT 261, 264
 - fabrique de composants WinRT 270
 - métadonnées 267
 - modèle de threading 266
 - types WinRT 273
- Windows Store 8, 223
 - augmenter la visibilité 233
 - découvrir une application 224
 - enregistrement et validation 227
 - gérer les applications publiées 235
 - licence développeur 226
 - localiser les captures d'écrans 231
 - manifeste d'une application 227
 - mode release 229
 - noter l'application 233
 - publier une application 226
 - règles à respecter 232
 - soumettre une application 229
 - télécharger 234, 235
- Windows Store vs Bureau 30
- Windows 8 10
- Windows RT 11
- WWAHost 13

X

- XAML 20
 - liaison de données 55

Z

- zoom sémantique 36, 46
- Zune 27