Introduction

Lorsque vous décidez d'écrire une application logicielle qui s'adresse au plus vaste marché de clients potentiels qui soit, et qui peut rapporter beaucoup, il est sensé de concevoir une application Windows 8.

Microsoft Windows est le système d'exploitation le plus utilisé dans le monde avec plus de 90 % du marché. En juin 2012, plus de 600 millions de licences Windows 7 avaient été vendues. La taille du marché Windows fait ressembler les autres marchés d'applications à des nains (y compris les marchés iPhone et Android).

Tous les utilisateurs d'une version antérieure de Windows ne vont pas basculer vers Windows 8, mais on peut sans se tromper prédire qu'une énorme proportion le fera. Le P-DG de Microsoft, Steve Ballmer (dont l'avis n'est sans doute pas le plus impartial), prévoit que plus de 500 millions de gens utiliseront Windows 8 d'ici à la fin de 2013.

Personnellement, je veux des WC en or massif, un jet privé et une décapotable de sport Tesla Roadster (orange). Ces objectifs restent assez modestes et je sais que beaucoup de mes lecteurs ont les mêmes. La solution la plus sûre pour les atteindre, vous ou moi (moi, je l'espère), est de concevoir des applications Windows 8.

Lorsque vous avez conçu une telle application, vous pouvez la mettre en vente directement dans Windows 8 qui comporte une boutique en ligne nommée Windows Store. Les applications peuvent y être proposées à un prix compris entre zéro et 999,99 USD. Les catégories sont très variées, de la suite de productivité (comme les agendas et les gestionnaires de contacts) aux jeux vidéo (pensez à *Angry Birds* et à *Cut the Rope*).

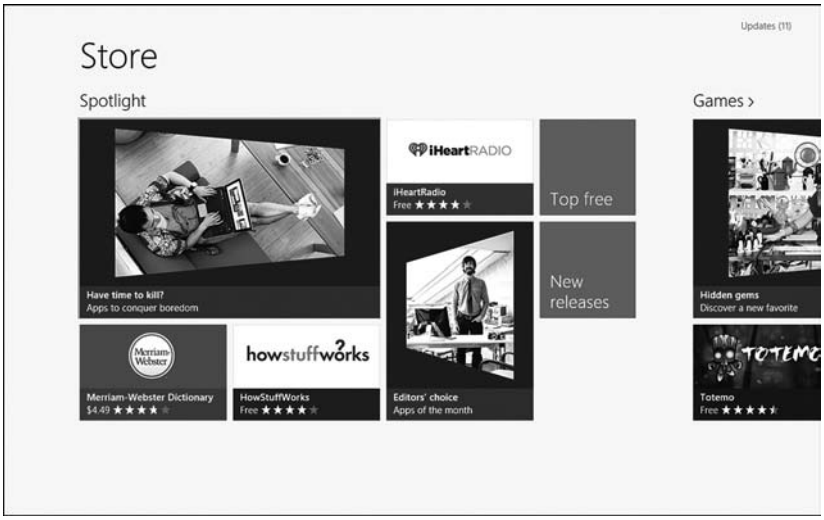

Figure 0.1

Ce livre est consacré à la conception d'applications Windows Store avec les langages JavaScript et HTML5 destinées à être vendues dans Windows Store.

Pourquoi JavaScript et HTML5 ? D'autres technologies sont utilisables pour créer une application pour le Windows Store , et notamment le C#, le XAML et le C++, mais nous nous concentrons dans ce livre sur le couple JavaScript-HTML5.

Le principal avantage de ce couple JavaScript-HTML5 est de vous permettre de réutiliser votre savoir-faire en conception de sites web pour créer des applications Windows. Si vous savez déjà programmer en JavaScript avec le HTML et le CSS, vous constaterez qu'il est facile d'écrire des applications Windows Store.

Ce livre contient tout ce qu'il faut savoir pour construire une application Windows Store. Vous y verrez comment utiliser la librairie WinJS (*Windows Library for JavaScript*) pour créer des applications JavaScript en exploitant par exemple des contrôles d'interface tels que `Rating`, `Menu` et `ListView`.

Vous apprendrez aussi à exploiter l'architecture Windows Runtime qui permet d'accéder aux fonctions de Windows 8 et à réaliser des actions normalement impossibles pour une application web pure, comme capturer de la vidéo et du son.

À la fin de ce livre, vous saurez comment créer une application Windows Store de productivité ou de jeu vidéo. Par exemple, les chapitres 7 et 8 présentent le contrôle `ListView` qui permet de créer facilement une liste de tâches à faire et le chapitre 12 montre comment concevoir un petit jeu d'arcade, *Brain Eaters*.

Lisez ce livre, concevez votre application Windows Store, vendez-en des milliers et allez acheter votre jet privé !

Prérequis du livre

Vous pouvez considérer que vous possédez les compétences pour bien exploiter ce livre si vous savez construire un site web avec du HTML, du JavaScript et des styles CSS.

Pour produire des applications Windows Store, vous devez satisfaire aux deux contraintes logicielles suivantes :

1. Tout d'abord, puisque vous allez concevoir des applications Windows Store, vous devez disposer d'une machine sous Windows 8. Sans le système Windows 8, vous ne pourrez pas mettre en pratique les exemples de ce livre.
2. Pour essayer les exemples, il vous faut l'atelier de développement Microsoft Visual Studio 2012. Vous pouvez vous servir de la version gratuite Visual Studio 2012 Express pour Windows 8. Elle est téléchargeable sur le site **microsoft.com**.

NOTE

Dans la version imprimée du livre, les lignes de code trop longues sont poursuivies en ligne suivante, opération de formatage qui est rendue visible par un caractère spécial (➥) qu'il ne faut bien sûr pas saisir.

Code source

Le code source de tous les projets et solutions du livre peut être téléchargé dans le référentiel GitHub à l'adresse suivante :

https://github.com/StephenWalther/Windows8AppsUnleashed

Utilisez le lien **Downloads** pour récupérer le fichier archive au format ZIP.

N.d.T.

L'archive des exemples en français est disponible dans la page de la version française. Rendez-vous sur le site de **pearson.fr** et cherchez le code livre **2589**.

1

Conception d'une application Windows Store

Sommaire

- Structure d'une application Windows Store
- Création d'une première application Windows Store
- Composants d'une application Windows Store
- Conception d'applications avec Visual Studio
- Débogage d'une application Windows Store
- Publication dans Windows Store

Je me propose dans ce chapitre de présenter les principes de conception d'une application Windows Store en commençant par voir en quoi elle se distingue d'une application Windows classique. Vous verrez ce qui caractérise une application destinée à être diffusée par le magasin Windows Store.

Libéré de toute appréhension, ce qui est, je l'espère, votre cas aussi, je vous guide au long de la création d'une première application Windows Store en utilisant Microsoft Visual Studio 2012 pour la concevoir, l'exécuter et la déboguer.

Nous pourrons ensuite analyser les constituants d'une telle application : le code HTML5, le code JavaScript, la librairie Windows Library for JavaScript et le moteur d'exécution Windows Runtime.

Nous terminerons par une partie financière puisque je vais vous montrer comment publier votre application dans le Windows Store dans l'espoir d'engranger des tas d'or.

Structure d'une application Windows Store

Je ne peux oublier mon premier contact avec un iPhone. Quand vous faites défiler une liste, l'image rebondit en arrivant en bout de liste ! Quand vous jetez un message à la corbeille, une animation fait aspirer l'objet par la corbeille ! C'est comme si l'iPhone contenait un véritable petit univers qui se plie aux lois de la physique.

Pour une raison que je n'ai pas totalement comprise, cette illusion d'un monde qui existe dans mon iPhone me rend heureux. Mes interactions avec un iPhone sont agréables.

Venons-en à Windows. Hormis peut-être l'effet de rebond des cartes à jouer dans le jeu Windows Solitaire, je n'ai pas souvenir d'avoir vu quoi que ce soit dans Windows qui m'ait donné ce sentiment d'enjouement. Je ne sais plus quand j'ai ressenti de la joie pour la dernière fois avec Windows.

Microsoft a enfin compris l'importance de la qualité de l'expérience utilisateur en concevant les applications pour le Windows Store qui sont régies par un jeu de principes de conception appelé *Microsoft design style principles*. En vous conformant à ces principes, vous allez créer des applications Windows Store plus agréables à utiliser et plus réactives.

Les principes de conception Microsoft Design Style

Les principes d'interface utilisateur Microsoft Design Style ont été conçus par Microsoft dans la perspective de l'utilisation des applications sur les Windows Phone, sur Xbox Live et le dorénavant abandonné baladeur Zune. Vous pouvez voir ces principes en action sur les sites de Microsoft tels que Microsoft SkyDrive et le portail Windows Azure Portal. Passons-les en revue :

Montrez votre amour de la belle ouvrage.

- Consacrez tout le temps et l'énergie requis pour soigner ces petits détails qui seront vus par beaucoup de personnes.
- Assurez-vous que l'expérience utilisateur donnera le sentiment d'être soignée à chaque étape.

Faites-en plus avec moins.

- Évitez les distractions, mais ne limitez pas la joie de la découverte. Si vous donnez aux utilisateurs la possibilité de se plonger dans ce qu'ils aiment, ils exploreront vraisemblablement le reste.
- Créez une expérience claire et fonctionnelle en affichant uniquement les éléments appropriés à l'écran pour que les utilisateurs puissent s'immerger dans le contenu.

Soyez rapide et fluide.

- Faites en sorte que les utilisateurs puissent interagir directement avec le contenu et réagissez rapidement à leurs demandes avec énergie.
- Rendez l'expérience utilisateur vivante, créez un sens de continuité et construisez des enchaînements limpides pour raconter une histoire.

Soyez authentiquement numérique.

- Tirez pleinement parti de la nature numérique du projet. Supprimez les limites physiques pour créer des expériences plus efficaces et plus simples que dans la réalité.
- N'oubliez pas que votre application n'est en fait qu'une animation de pixels sur un écran. Utilisez des couleurs et des images innovantes, touchantes et remarquables qui vont au-delà des limites du réel.

Gagnez en équipe.

- Tirez parti de l'écosystème et veillez à ce que votre application fonctionne de concert avec d'autres applications, périphériques et le système pour fournir aux utilisateurs une solution complète à leur problème.
- Appuyez-vous sur le modèle d'interface utilisateur pour limiter la redondance. Profitez de ce que les gens savent déjà pour leur conférer un sentiment de familiarité, de contrôle et de confiance.

Note

Ces principes de style de conception Microsoft s'appelaient au départ "principes de conception Metro". La présente liste s'inspire du texte Microsoft situé à l'adresse **http://msdn.microsoft.com/en-us/library/windows/apps/hh464920**. Voyez aussi la page **http://msdn.microsoft.com/en-us/library/windows/apps/hh465424.aspx**.

Lorsque j'ai pris connaissance de ces principes, je les ai trouvés bien trop abstraits et évasifs. Tout à fait le genre de conseils que délivrent certains ergonomes éloignés de la réalité quotidienne.

Mais j'ai ensuite vu comment ces principes pouvaient être appliqués dans de vraies applications Windows Store, et j'ai commencé à réviser mon jugement à leur égard.

Prenez la deuxième règle demandant d'en faire plus avec moins. Une caractéristique très distinctive d'une application Windows Store est l'absence de tout habillage : une telle application sous Windows n'a pas de fenêtre-cadre. Le contenu est affiché en plein écran.

Cette absence d'habillage invite l'utilisateur à rester concentré sur le contenu de l'application. Windows 8 est ainsi livré avec deux variantes de son navigateur

Internet Explorer : une variante de bureau et la variante dédiée à Windows 8, conforme aux principes de conception Microsoft.

Je préfère la variante Windows 8 d'Internet Explorer à la variante de bureau avec habillage. Dans la variante Windows 8, vous voyez la page visitée, ce qui est après tout l'objectif de cette application.

Voyons le principe de fluidité. La raison pour laquelle j'adore manipuler mon iPhone est l'impression de vie que donne l'interface, et cette illusion est produite par un judicieux emploi d'animations dans l'interface. Sur un iPhone, les objets rebondissent et vibrent.

Vous êtes invité à tirer parti des animations lorsque vous concevez une application Windows Store. Si vous utilisez par exemple le composant de contrôle d'interface standard de liste nommé `ListView` (nous le verrons en détail dans ce livre), des animations sont ajoutées aux opérations d'ajout et de suppression d'éléments dans la liste. Lorsque vous ajoutez un élément, il est inséré progressivement et lorsque vous le supprimez, les éléments précédent et suivant viennent recouvrir l'espace qu'il occupait.

Note

Le terme technique qui désigne les moyens ajoutés dans une application informatique pour imiter ou rappeler la réalité physique est le *skeuomorphisme*.

Caractéristiques universelles des applications Windows Store

Une application Windows Store est par définition conforme aux principes de conception Microsoft. Elle est prévue dès le départ pour fonctionner sous Windows 8 et sous Windows RT.

Toutes les applications Windows Store partagent un jeu de fonctions que je vous propose de découvrir avec la variante Windows 8 d'Internet Explorer.

Gestion du clavier, de la souris, du toucher et du stylet

Une application Windows Store est caractérisée par de grandes icônes appelées *tuiles* et beaucoup d'espace libre. Cet aspect aéré de l'interface fait que les applications Windows Store sont faciles à utiliser sur écran tactile même avec de gros doigts.

Les applications Windows Store doivent pouvoir être pilotées aussi bien sur écran tactile que sur écran passif avec un clavier et une souris.

Du point de vue du développeur, le fonctionnement de Windows 8 est tel que vous n'avez pas à vous faire beaucoup de souci pour gérer les particularités des interactions tactiles. Tant que vous utilisez les contrôles d'interface standard WinJS, vous disposez automatiquement d'une gestion hybride par clavier-souris et tactile.

La barre d'application et la barre de navigation

La Figure 1.1 donne un aperçu de la version Windows 8 du navigateur Internet Explorer montrant un article d'un grand quotidien. Vous remarquez que le contenu occupe tout l'espace : pas de barre d'outils, de boutons ni de barre d'état.

Figure 1.1
Internet Explorer de Windows 8.

Dans une application Windows Store, vous regroupez toutes vos commandes dans la barre d'application qui ne s'affiche que sur demande : en balayant l'écran depuis le bas ou le haut ou par clic droit.

La barre d'application d'Internet Explorer contient la barre d'adresse et plusieurs boutons, notamment celui de retour à la page précédente (voir Figure 1.2).

Figure 1.2
La barre d'application et la barre de navigation.

Création d'une première application Windows Store

N'ayez aucune crainte, car je vais vous guider pas à pas dans cette section dans la construction d'une application Windows Store complète. Nous n'allons pas nous plier à la tradition consistant à concevoir comme premier programme l'affichage d'un message de bienvenue du style *Hello World*. Je vous propose quelque chose plus en rapport avec le type d'application que l'on s'attend à trouver dans cet environnement.

Je vais en effet expliquer comment créer une application pour prendre des photos avec la webcam de l'ordinateur ou de la tablette. En choisissant la commande **Prendre une photo** dans la barre d'application, vous allez capturer puis afficher la photo (voir Figure 1.6).

Note

Le code source complet de ce projet se trouve dans le sous-dossier App1 du dossier du Chapitre 1 dans l'archive.

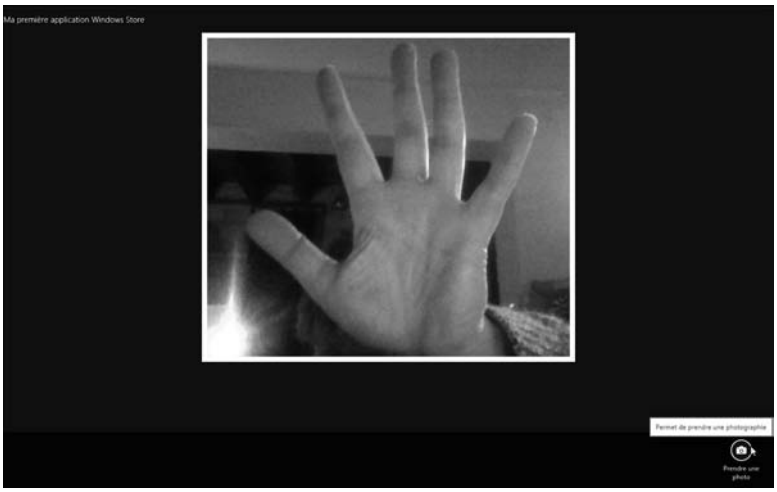

Figure 1.6
Aspect de votre première application Windows Store.

Lancement du projet Visual Studio

Nous commençons par lancer la création d'un projet Microsoft Visual Studio. Je me sers de l'atelier Visual Studio pour tous les exemples de ce livre, et même de sa version gratuite Visual Studio Express pour Windows 8 lorsque c'est possible. Dans deux projets, j'ai eu recours à la version payante complète. Vous pouvez télécharger l'atelier sur le site **microsoft.com**.

Note

Dans les deux chapitres dans lesquels j'ai recours à un service web, j'ai utilisé la version payante Visual Studio Professional 2012 parce que je devais créer un projet ASP.NET en plus du projet d'application Windows Store dans la même solution.

Note

Vous pouvez aussi créer des applications Windows Store avec Microsoft Blend, mais si vous voulez publier dans le magasin Windows Store, je vous conseille d'adopter Microsoft Visual Studio.

Rappelons que pour pouvoir créer des application Windows Store, vous devez utiliser une machine fonctionnant sous Windows 8. Si vous n'en avez pas, vous pouvez éventuellement lancer une machine virtuelle *via* Windows 8 avec un outil tel que VMware Player.

Démarrez l'atelier Visual Studio et ouvrez le menu **Fichier** puis choisissez **Nouveau projet**. Dans la boîte de nouveau projet, sélectionnez à gauche la catégorie **JavaScript** puis le modèle de projet **Application vide**. En bas, saisissez le nom de projet `App1` et validez par le bouton OK (voir Figure 1.7).

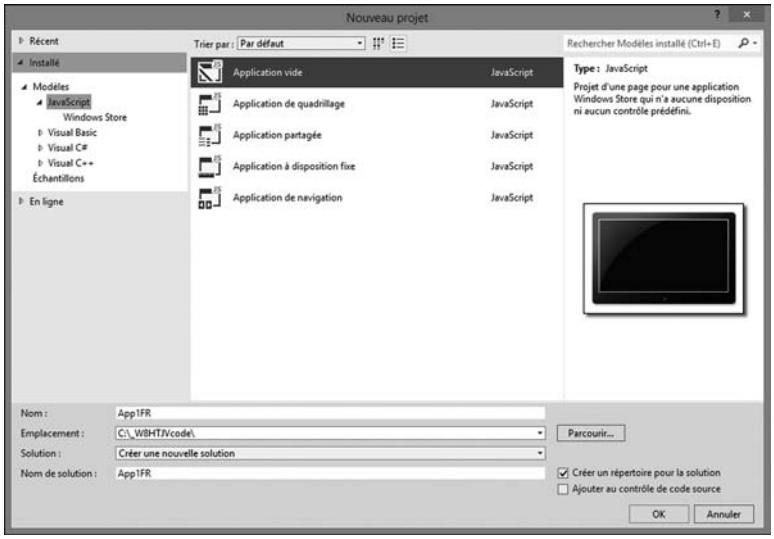

Figure 1.7
Lancement d'un nouveau projet dans Visual Studio.

Cette simple opération a permis de créer l'ossature de votre projet. Observez le contenu du volet droit de l'Explorateur de solutions (voir Figure 1.8). Trois fichiers ont été créés d'office pour constituer le point de départ de l'application Windows Store : dans la racine relative, le fichier default.html, puis un fichier JavaScript default.js (dans le sous-dossier js) et un fichier de styles default.css (dans le sous-dossier css).

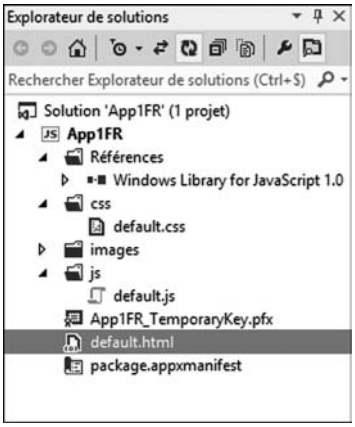

Figure 1.8
Les fichiers initiaux d'une application Windows Store.

Déclaration des capacités de l'application

Avant de passer à la rédaction des instructions de code source, il nous reste un paramétrage à réaliser. Notre projet doit permettre d'utiliser la webcam pour prendre des photos, ce qui peut constituer une intrusion dans la vie privée. Imaginez que l'application se lance en coulisses et vous surveille puis envoie les images à un paparazzi qui rôde sur Internet (ou à Steve Ballmer qui vous surveille de son bureau chez Microsoft).

Dès que votre programme peut faire une action délicate, vous devez déclarer cette dernière pour que l'utilisateur puisse se voir proposer d'autoriser ou non l'opération. Pour déclarer les capacités, vous accédez au fichier manifeste de l'application. Pour y accéder, double-cliquez sur le nom package.appxmanifest dans le volet de l'Explorateur de solutions.

Accédez alors au deuxième onglet, **Capacités**, qui affiche les valeurs de toutes les capacités possibles pour une application. Si vous avez besoin d'enregistrer des sons, vous activerez la capacité **Microphone** ; si vous voulez permettre d'enregistrer des images dans la bibliothèque personnelle de l'utilisateur, vous activerez la capacité **Bibliothèque d'images**. Dans notre projet, nous avons besoin de la capacité **Webcam** pour pouvoir prendre des photos (voir Figure 1.9). Cochez sa case et refermez le fichier en autorisant l'enregistrement (raccourci **ALT + F4**).

Figure 1.9
Activation de la capacité Webcam.

Lors du premier démarrage de l'application par un utilisateur, il lui est présenté un écran dans lequel il doit décider s'il accepte ou pas l'accès à sa webcam (voir Figure 1.10). Cette question n'est posée qu'une fois.

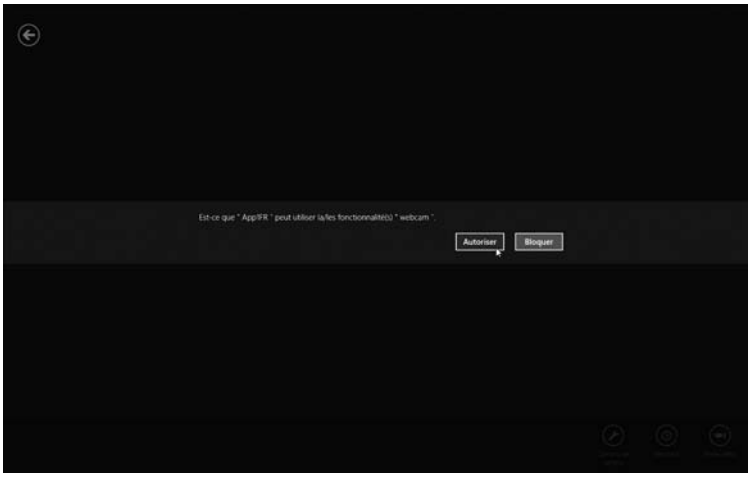

Figure 1.10
Demande d'autorisation d'utiliser la webcam.

Note

Une fois que l'utilisateur a donné son autorisation, il peut changer d'avis en supprimant une autorisation par le volet Autorisations du volet général des Paramètres.

Création de la page HTML

Lorsque vous demandez la création d'un projet d'application Windows Store à partir d'un modèle, vous disposez dès le départ d'un fichier default.html minimal implanté dans la racine du projet. C'est cette page qui est affichée au démarrage de l'application. Il faut en enrichir le contenu pour qu'elle propose la commande de prise de photo qui est l'objectif de notre projet (voir Listing 1.1).

Listing 1.1 : La page default.html modifiée pour notre projet

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>App1</title>

    <!-- Références WinJS  -->
    <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
    <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
    <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

    <!-- Références App1  -->
    <link href="/css/default.css" rel="stylesheet" />
    <script src="/js/default.js"></script>
</head>
<body>

    <img id="imgPhoto" src="/images/placeholder.png" />

    <!-- Contrôle AppBar -->
    <div id="appBar1" data-win-control="WinJS.UI.AppBar">
        <button data-win-control="WinJS.UI.AppBarCommand"
            data-win-options="{
                id:'cmdPrendrePhoto',
                label:'Prendre une photo',
                icon:'camera',
                tooltip:'Permet de prendre une photographie'
            }">
        </button>
    </div>
</body>
</html>
```

Nous avons remplacé dans le corps BODY la mention `<p>Le contenu s'affiche ici</p>` par le bloc entier que vous pouvez récupérer par copier-coller depuis le dossier App1FR de l'archive des exemples.

Remarquez d'abord l'élément IMG associé à l'identifiant ID `imgPhoto`. C'est ici que nous afficherons la photographie.

Vient ensuite un élément DIV doté de l'attribut `data-win-control="WinJS.UI.AppBar"`. Il s'agit de votre première rencontre avec un composant de contrôle WinJS. Ce contrôle `AppBar` fait afficher une barre d'application qui héberge le bouton pour prendre la photo (voir Figure 1.11).

Figure 1.11
La commande de prise de photo dans la barre d'application.

Création de la feuille de styles CSS

Le deuxième fichier qui est produit automatiquement par le modèle de projet est la feuille de styles default.css dans le sous-dossier css. La version initiale contient quatre règles de gestion des états d'affichage de l'application Windows Store.

Note

Je présente les états d'affichage d'une application Windows Store dans le Chapitre 9.

Dans notre exemple, vous pouvez supprimer tout le contenu prégénéré du fichier default.css pour le remplacer par l'unique règle de style suivante. Elle contrôle l'aspect de la photographie qui va apparaître dans l'élément IMG :

```
#imgPhoto {
    display:block;
    margin: 15px auto;
    border: 10px solid white;
    max-width: 90%;
    max-height: 90%;
}
```

3

Observateurs, liaisons et templates

Sommaire

- Principe des observateurs
- Principe des liaisons de données déclaratives
- Principe des templates

Dans ce chapitre, vous allez voir comment afficher dans les pages de votre application Windows Store les données issues d'objets JavaScript, comme les données d'un produit ou d'une liste de produits.

Je présente tout d'abord les observateurs qui permettent de détecter que la valeur d'une propriété JavaScript a changé, et ce de façon automatique. Je montre ensuite comment profiter de l'objet `WinJS.Binding.List` afin de savoir quels sont les éléments d'un tableau qui ont évolué.

Je me tourne ensuite vers le concept de liaison de données déclarative. Vous verrez comment utiliser des objets JavaScript standard et des objets observateurs en relation avec cette liaison de données déclarative.

J'aborde enfin les techniques d'affichage d'un tableau d'objets au moyen d'un modèle WinJS appelé *template*. Un template permet de définir le format et d'afficher plusieurs objets JavaScript à la fois.

Principe des observateurs

Un *observateur* est un objet permettant de prévenir un ou plusieurs auditeurs que la valeur d'une propriété a changé.

N.d.T.

Les Anglais ont choisi le terme "observable" pour cette entité, alors qu'il s'agit d'une entité non pas passive mais active. Le terme "observateur" est donc plus approprié.

Les observateurs vous permettent de maintenir l'interface utilisateur synchrone avec les données de l'application. Vous pouvez ainsi demander la mise à jour automatique de l'interface dès qu'une propriété d'un produit change. Les observateurs constituent le fondement sur lequel s'appuie le mécanisme de liaison déclarative de la librairie WinJS.

Note

La librairie WinJS n'est pas la première librairie JavaScript à utiliser les observateurs. Les librairies Backbone, Knockout, Ember et la librairie Microsoft Ajax (dorénavant intégrée à Ajax Control Toolkit) en utilisent toutes.

Création d'un observateur

Supposons que j'ai créé un objet nommé `product` tel que celui-ci :

```
var product = {
    name: "Lait",
    description: "On peut le boire",
    price: 12.33
};
```

A priori, il n'a rien d'enthousiasmant avec ses trois propriétés intitulées `name`, `description` et `price`.

Supposons maintenant que je désire être prévenu automatiquement dès que l'une de ces propriétés change. Je peux créer un observateur à partir de l'objet `product` de la façon suivante :

```
var observableProduct = WinJS.Binding.as(product);
```

Cette instruction demande la création d'un nouvel objet JavaScript nommé `observableProduct` à partir de l'objet JavaScript existant qui se nomme `product`. Le nouvel objet hérite des trois propriétés `name`, `description` et `price`, mais à la différence de celles de l'observateur, ces trois propriétés savent déclencher des notifications lorsque la valeur d'une propriété change.

En fait, les propriétés du nouvel objet ont été transformées en propriétés d'accès, donc avec un couple de méthodes de lecture (get) et d'écriture (set). Voici à quoi ressemble la propriété `price` de cet observateur :

```
price: {
        get: function () { return this.getProperty("price"); }
        set: function (value) { this.setProperty("price", value); }
}
```

Dorénavant, lorsque vous demandez la valeur de la propriété `price`, cela provoque un appel à la méthode nommée `getProperty()`. De même lorsque vous voulez modifier la valeur de `price`, cela provoque un appel à la méthode `setProperty()`. Ces deux méthodes appartiennent à l'objet observateur.

Voici les différentes méthodes et propriétés que possède l'objet observateur :

- `addProperty(nom, valeur)` : ajoute une propriété à un observateur et en avertit tous les auditeurs.
- `backingData` : objet incarnant la valeur de chaque propriété.
- `bind(name, action)` : permet d'exécuter une fonction lorsqu'une propriété change.
- `getProperty(nom)` : renvoie la valeur d'une propriété à partir du nom de la propriété sous la forme d'une chaîne.
- `notify(nom, valeurNouv, valeurAncien)` : méthode privée qui exécute chacune des fonctions trouvées dans le tableau `_listeners`.
- `removeProperty(nom)` : supprime une propriété et en avertit tous les auditeurs.
- `setProperty(nom, val)` : modifie une propriété et en avertit tous les auditeurs.
- `unbind(nom, action)` : permet de forcer l'arrêt d'exécution d'une fonction lorsqu'une propriété change.
- `updateProperty(nom, val)` : modifie la valeur d'une propriété et en avertit tous les auditeurs.

Lorsque vous créez un observateur, vous obtenez en retour un nouvel objet qui possède les mêmes propriétés que l'objet de départ. Lorsque vous modifiez les propriétés d'un observateur, vous pouvez en profiter pour prévenir les auditeurs que la valeur d'une propriété a changé, et ce de façon automatique.

Supposons que vous modifiiez la valeur de la propriété `price` ainsi :

```
observableProduct.price = 2.99;
```

Voici la séquence d'événements déclenchée par cette modification :

- La méthode d'écriture de `price` appelle la méthode `setProperty("price", 2.99)`.
- La méthode `setProperty()` modifie la valeur de la propriété `backingData.price` puis appelle la méthode `notify()`.
- La méthode `notify()` exécute tour à tour toutes les fonctions trouvées dans la liste des auditeurs associées à la propriété `price`.

Vous pouvez faire exécuter automatiquement une ou plusieurs fonctions d'auditeurs dès qu'une propriété d'un observateur change.

Attention

Vous subirez une exception si vous appelez la méthode `WinJS.Binding.as()` sur un objet WinRT. En effet, les objets WinRT sont immuables, mais la méthode `WinJS.Binding.as()` tente d'ajouter une nouvelle méthode nommée `_getObservable()` à l'objet non modifiable. Autrement dit, vous pouvez créer des observateurs pour des objets JavaScript, mais pas pour des objets WinRT.

Création d'un auditeur d'observateur

Il ne suffit pas de créer un objet observateur pour être informé de ses évolutions. Il faut également lui associer un auditeur, en établissant un lien entre les deux au moyen de la méthode `bind()`, comme illustré dans le Listing 3.1.

Listing 3.1 : Liaison d'un propriété d'objet à un auditeur (observables\observables.html)

```
// Simple product object
var product = {
    name: "Lait",
    description: "C'est à boire",
    price: 12.33
};

// Création de l'observateur product
var observableProduct = WinJS.Binding.as(product);

// Déclenche une fonction lorsque le prix change
observableProduct.bind("price", function (newValue, oldValue) {
    console.log(newValue + " was " + oldValue);
});

// Change le prix
observableProduct.price = 2.99;
```

Dans cet exemple, la méthode `bind()` permet d'associer la propriété nommée `price` à une fonction qui est exécutée en renvoyant l'ancienne et la nouvelle valeur de la propriété sur la console JavaScript de Visual Studio (voir Figure 3.1).

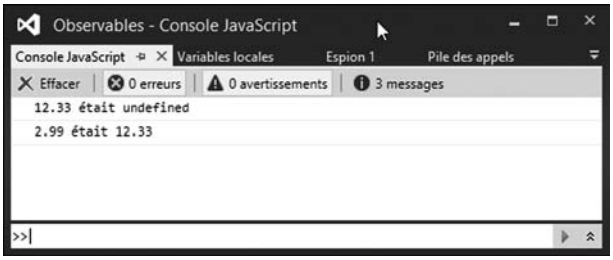

Figure 3.1
Exemple de message après liaison d'une propriété.

Revoyons comment est établie l'association entre la propriété `price` et la fonction :

```
observableProduct.bind("price", function (newValue, oldValue) {
    console.log(newValue + " was " + oldValue);
});
```

Notez bien que la fonction associée à la propriété est appelée deux fois, une fois pour la valeur initiale de la propriété, puis une autre lorsque la propriété a changé de valeur.

Note

Pour associer un auditeur à une propriété complexe, vous pouvez fournir un objet en tant que second paramètre de la méthode `WinJS.Binding.bind()`, comme ceci :

```
// Création d'objet à propriété complexe
var customer = {
    shippingAddress: {
        street: "312 rue du Bac"
    }
};

// Création de l'observateur
var observableCustomer = WinJS.Binding.as(customer);

// Liaison à la propriété complexe
WinJS.Binding.bind(observableCustomer, {
    shippingAddress: {
        street: function (newValue) {
            console.log("Adresse de livraison changée en " + newValue);
        }
    }
});


// Change la valeur de la propriété complexe
observableCustomer.shippingAddress.street = "100 avenue Solent";
```

Création d'une collection d'observateurs

Par défaut, lorsque vous créez un objet `WinJS.Binding.List` à partir d'un tableau JavaScript, la liste résultante devient un observateur, contrairement aux éléments de cette liste. La liste contient tout simplement les éléments du tableau qui correspondent à des objets JavaScript classiques.

Pour que chacun des éléments du tableau JavaScript devienne un observateur à son tour, il faut mentionner l'option de liaison lors de la création de la liste `WinJS.Binding.List`, de la manière suivante :

```
var products = [
    { name: "Lait", price: 2.99 },
    { name: "Oranges", price: 2.50 },
    { name: "Pommes", price: 1.99 }
];

// Création de la liste
var productsList = new WinJS.Binding.List(products, { binding: true });

// Se tient paré aux changements de prix
productsList.getAt(1).bind("price", function () {
    console.log("price a changé");
});
```

Dans cet exemple, `productsList` contient une liste d'observateurs. Chaque objet peut donc être associé à une fonction auditeur qui sera déclenchée en cas de modification d'une propriété de l'observateur. Dans notre exemple, nous faisons afficher un message sur la console dès que la propriété `price` change.

Liaisons de données déclaratives

Une liaison de données déclarative permet de relier les attributs d'un élément HTML aux propriétés d'un objet JavaScript. Vous profiterez de ce mécanisme dès que vous aurez besoin d'afficher des données dans une page HTML.

Commençons par un exemple simple. Le Listing 3.5 affiche les détails d'un produit (voir Figure 3.5).

Listing 3.5 : Exemple de liaison de données déclarative simple (dataBinding\ dataBinding.html)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Chapter03</title>
```

```
        <!-- Références WinJS -->
        <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
        <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
        <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

        <!-- Références du Chapitre 3 -->
        <link href="/css/default.css" rel="stylesheet" />
        <script src="/dataBinding/dataBinding.js"></script>
    </head>
    <body>

        <h1>Détails de produit</h1>

        <div>
            Nom du produit:
        <span data-win-bind="innerText:name"></span>
    </div>
    <div>
        Prix du produit:
        <span data-win-bind="innerText:price"></span>
    </div>
    <div>
        Visuel du porduit:
        <br />
        <img data-win-bind="src:photo;alt:name" />
    </div>
    </body>
    </html>
```

Figure 3.5
Affichage des détails d'un produit grâce à la liaison de données déclarative.

Note

Vous remarquez dans le Listing 3.24 que nous utilisons la méthode `render()` en lui fournissant un objet vide :

```
tmplProduct.winControl.render({}).then(function () {
   ...
});
```

Cette précaution permet de contourner un bogue de la librairie WinJS.

Résumé

Ce chapitre était consacré à l'affichage des contenus des objets JavaScript sur une page. Nous avons d'abord étudié les observateurs, et notamment leur capacité à vous prévenir automatiquement du changement de valeur d'une propriété d'un objet JavaScript. Nous avons vu comment utiliser l'objet `WinJS.Binding` pour détecter différents types de changements dans un tableau d'éléments.

Nous avons ensuite vu l'intérêt des liaisons de données déclaratives, avec des objets JavaScript standard et avec des objets observateurs. Ce mécanisme de liaison permet d'afficher les valeurs des propriétés JavaScript dans une page.

Je vous ai enfin montré comment utiliser les templates de WinJS pour contrôler le format et l'affichage d'un tableau d'objets. Nous avons vu comment créer un template de façon impérative et de façon déclarative.