

1 Rappel sur les possibilités d'intégration

Microsoft propose à travers son offre Bing Maps, de multiples possibilités d'intégration. Tout d'abord, l'exposition de l'ensemble de ses services et contenus à travers les services Web dédiés permet de pouvoir communiquer avec ces derniers depuis n'importe quelle application, plate-forme ou langage. Le chapitre consacré à l'utilisation dans un cadre de mobilité est un exemple de possibilité d'intégration sur une plate-forme totalement différente.

L'utilisation des contrôles fournis permet d'accélérer la réalisation des interfaces interactives et ici l'utilisation des technologies habituelles à savoir HTML, JavaScript ou plus récemment le contrôle Silverlight, permet d'intégrer ces éléments dans la plupart des applications Web quel que soit le socle de base utilisé pour le développement.

À travers ce chapitre, nous allons utiliser des plates-formes différentes de celles des exemples précédents afin de mettre en évidence cette interopérabilité.

2 Mise en œuvre dans des applications

2.1 Utilisation simple au sein de WPF

Description

À travers cet exemple, nous allons étudier la première méthode nous permettant d'intégrer Bing Maps et le contrôle cartographique au sein d'une application WPF.

Cette technique exploite l'utilisation du contrôle WebBrowser composant COM disponible à travers le système Windows depuis bien des versions.

Au sein de ce contrôle, l'idée est de charger une page HTML qui contiendra le contrôle et avec lequel il nous sera possible d'interagir grâce à la méthode `InvokeScript()` du contrôle WebBrowser. Cette méthode permet en clair de déclencher l'exécution de méthode JavaScript depuis du code managé de l'application.

Aussi, pour simplifier cette intégration, il est à noter qu'il existe un projet CodePlex permettant d'obtenir les méthodes de bases pour l'interaction avec le composant Bing Maps. Ce projet est disponible à l'adresse suivante : <http://vewpf.codeplex.com/>

Ce projet proposé sur CodePlex par Burcu Dogan, est à l'origine un Proof Of Concept mis à disposition par Marc Schweigert. Il ne contient pas toutes les fonctionnalités, et il est recommandé de conserver les sources afin de garantir la correction de bug lié à l'interopérabilité entre le code managé et le code JavaScript utilisé au sein du contrôle Webbrowser.

Le blog de Burcu Dogan est accessible à l'adresse suivante : <http://blog.burcudogan.com/>

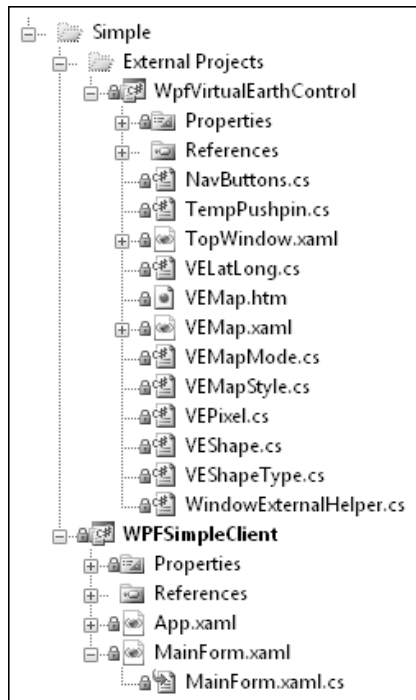
Le blog de Marc Schweigert est relayé via : <http://blogs.msdn.com/publicsector/>



Attention : le projet mis à disposition sur CodePlex nécessite l'utilisation d'une culture anglo-saxonne. Il peut être alors utile de modifier la langue utilisée au sein de l'application grâce à la ligne de code suivante : `Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("en-US");`
Il est également possible de corriger (comme ici dans l'exemple fourni) le code JavaScript directement.

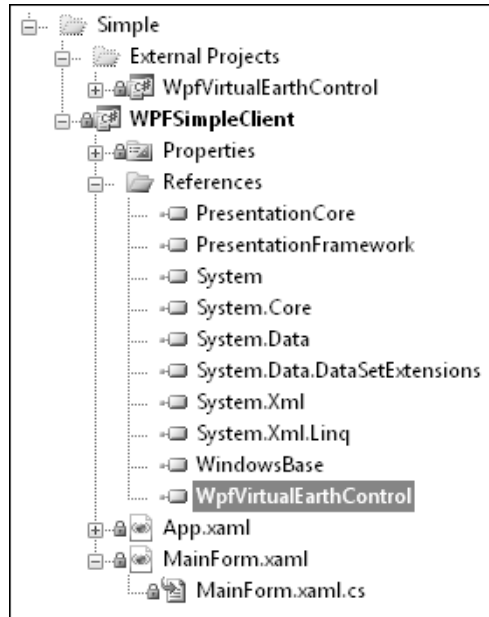
Développement

Pour réaliser ce projet d'exemple d'intégration au sein d'une application WPF, nous utilisons l'architecture suivante :



Ainsi on remarque l'import du projet issu de CodePlex, directement ajouté dans la solution afin de garantir une nouvelle fois, l'accès au code source de ce composant.

Le projet est ensuite ajouté comme référence au sein du projet client de type WPF :



Au sein de la fenêtre principale, on ajoute le préfixe que l'on associe à l'assembly du projet Codeplex :

```
.....  
xmlns:ve="clr-namespace:WpfVirtualEarthControl;assembly=WpfVirtualEarthControl"
```

Le code déclaratif peut alors être le suivant, notez que comme à l'habitude, on retrouve une complétion utile ici :

```
.....  
<Window x:Class="WPFSimpleClient.MainForm"  
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  xmlns:ve="clr-namespace:WpfVirtualEarthControl;assembly=  
WpfVirtualEarthControl"  
  Title="Window1" Height="400" Width="500">  
  <Grid>  
    <ve:VEMap x:Name="map"></ve:VEMap>  
  </Grid>  
</Window>
```

L'intégration à ce niveau est terminée, on obtient une carte directement interactive au sein de l'application :



Pour aller plus loin, limitations et considérations

Au sein de ce projet d'exemple, l'idée était de montrer comment exploiter le composant WebBrowser au sein d'une application WPF pour afficher la carte hébergée dans une page HTML.

Cette technique permet d'intégrer des existants de pages et ne nécessite globalement que quelques connaissances principalement orientées Web puisqu'ici le développement se résume au développement d'une page HTML et des scripts JavaScript associés.

Cependant, dans des cas où l'on souhaite obtenir une intégration plus avancée dans une application WPF, on peut souhaiter par exemple interagir avec la carte en sélectionnant des éléments au sein de l'application WPF, en choisissant une option dans un menu par exemple. Les invocations de script JavaScript peuvent poser quelques soucis et nécessitent une utilisation avec beaucoup de réflexion quant à l'impact que cela peut avoir surtout en termes de gestion d'erreurs.

De même, un des points négatifs de ce composant WebBrowser utilisé pour héberger la page HTML, réside dans le fait qu'il est rendu au dessus de tout contrôle WPF, peu importe comment on spécifie le Z-index. Ce composant issu des API Win32 ne prend pas en compte ces propriétés WPF et il n'est pas possible par exemple de gérer l'ordre d'affichage, la transparence, son orientation... Ce qui dans certains cas peut réellement poser problème.

Cette solution est donc à utiliser avec prudence mais elle permet principalement de se concentrer sur le développement d'une application Web et donc de favoriser la réutilisation des éléments développés avec les techniques et composants classiques.

2.2 Utilisation avancée au sein de WPF

Description

Compte tenu des limitations qui peuvent apparaître avec la technique précédemment décrite, il peut être utile de pouvoir intégrer la carte de manière plus complète au sein d'une application WPF.

Ici l'idée est de réaliser une application WPF qui sera utilisée sur la plate-forme Microsoft Surface, table tactile présentée en fin d'année 2008 au grand public qui connaît un succès grandissant et un intérêt certain pour ce type d'application utilisant la cartographie.

La technique utilisant le WebBrowser ne permet pas de récupérer les points de contact de la table Surface puisque cette capture des points de contact s'effectue au niveau de composants WPF. Le composant WebBrowser étant rendu par-dessus tout, les événements de type Contact ne sont pas capturés.

Avec la méthode ici utilisée, l'idée de base est simple. Elle consiste à utiliser le plug-in 3D de Bing Maps pour rendre la carte au sein d'une surface graphique qui accueille le rendu du plug-in. Avec cette méthode, on se retrouve donc limité à utiliser le plug-in 3D et donc la cartographie en 3D au sein de l'application. Par contre, il devient désormais possible de récupérer l'ensemble des événements associés aux éléments WPF et on peut bien entendu afficher des contrôles WPF par-dessus la carte permettant de proposer une interface enrichie.

Cette technique ici utilisée au sein d'une application WPF dédiée à une utilisation sur la plate-forme Surface, peut tout à fait être adaptée directement pour fonctionner sur une application WPF habituelle (sans contact).

Aussi, une nouvelle fois, pour simplifier l'intégration, l'équipe de la société Infostrat a partagé son projet d'intégration directement sur CodePlex au plus grand bonheur de tous. Il faut souligner que les efforts de développement à ce niveau-là sont directement partagés et la réalisation du code d'interopérabilité entre le plug-in et les événements WPF sont pour le moins peu simples. J'en profite pour réellement souligner l'esprit communautaire de la société Infostrat qui n'a pas hésité à répondre à l'invitation de partager leurs sources.

Ce projet est disponible à l'adresse suivante, il sera nécessaire pour le développement qui suit : <http://virtualearthwpf.codeplex.com/>

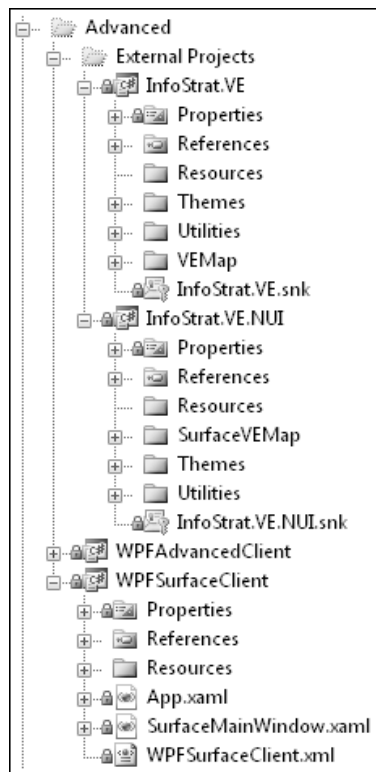
InfoStrat.VE



Ce composant nécessite l'installation du plug-in 3D de Microsoft Bing Maps. Dans un contexte de démonstration, n'oubliez pas d'installer ce plug-in avant de lancer l'application. Il est également possible d'extraire les assemblés utilisés sur une machine de développement en les récupérant dans le dossier GAC et ainsi de les fournir avec le livrable final sans obliger à installer le plug-in directement.

Développement

Afin de réaliser l'intégration décrite, voici l'architecture choisie :



On remarque l'intégration des deux projets récupérés sur le site CodePlex, chacun correspondant à l'utilisation au sein de WPF ou au sein d'une application WPF dédiée à être utilisée sur la table Surface.