Ceci est la version HTML du fichier <ftp://ftp.developpez.be/valtech/articles/dotnet/googlemaps/GoogleMaps-et-WinformDotnet.pdf>. Lorsque Google explore le Web, il crée automatiquement une version HTML des documents récupérés.

Astuce : Pour trouver rapidement votre terme de recherche sur cette page, appuyez sur **Ctrl+F** ou sur **⌘+F** (Mac), puis utilisez la barre de recherche.

|  |
| --- |
| **Page 1** |

**Jean-Louis VIDAL**

**Valtech Training**

**Intégrer Google Maps et Winform .NET**

**octobre 07**

**Résumé**

**1**

Google Maps rend un service visuel très intéressant pour l'utilisateur.

Comment utiliser et piloter les APIs Google Maps dans nos applications ?

Comment communiquer de .NET à Google Maps et inversement ?

Nous nous proposons de créer une application permettant de sélectionner une ville et d'afficher la

carte correspondante (via le géocodage).

L'utilisateur pourra préparer sa randonnée en traçant directement son parcours (les Way-points

métrés dynamiquement) sur la carte et l'exporter en format XML de façon à le télécharger sur son

GPS portable.

Lors de la randonnée, le GPS n'aura plus qu'à guider le randonneur...

Le présent document est rédigé sous la forme d’un tutoriel devant permettre à chacun de refaire la

présentation

|  |
| --- |
| **Page 2** |

**2**

**Table des matières**

**1. BUT DE LA PRÉSENTATION \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5**

**2. LA FENÊTRE CADRE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5**

2.1 Créer un projet WinForm.........................................................................................................5

2.2 Ajouter un contrôle WebBrowser ............................................................................................5

**3. AFFICHAGE D'UNE CARTE PAR DÉFAUT \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6**

3.1 Coder le javascript ..................................................................................................................6

3.2 Coder le HTML........................................................................................................................7

**4. ACTIVER LES CONTRÔLES PROPOSÉS PAR GOOGLEMAPS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8**

4.1 Le contrôle de la position par les flèches du clavier................................................................8

4.2 L’affichage du contrôle de zoom .............................................................................................8

4.3 L’affichage du contrôle de type de carte (plan, satellite, mixte) ..............................................8

4.4 L’affichage du contrôle d'aperçu général ................................................................................8

4.5 L’affichage de l'échelle............................................................................................................8

**5. PERSONNALISATION DE LA CARTE : AJOUT D’UN MARQUEUR ET RÉPONSE AUX ÉVÉNEMENTS**

**‘CLICK’ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9**

5.1 Afficher une icone sur le point central de la carte ...................................................................9

5.2 Créer un événement click sur le marqueur .............................................................................9

5.3 Afficher les coordonnées du clic ...........................................................................................10

**6. TRACER UN ITINÉRAIRE SUR LA CARTE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11**

6.1 Déclarer les variables route et points....................................................................................11

6.2 Créer une route … ................................................................................................................11

6.3 … et l'afficher ........................................................................................................................ 11

**7. CALCULER LES DISTANCES \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12**

7.1 Déclarer les variables............................................................................................................12

7.2 Calculer : ............................................................................................................................... 12

**8. DONNER À L'UTILISATEUR DOTNET UN MEILLEUR CONTRÔLE SUR LA CARTE\_\_\_\_\_\_\_\_\_\_\_\_\_14**

8.1 Établir la communication Dotnet => javascript ......................................................................14

8.1.1

Dans la page HTML, créer une fonction JavaScript "affiche"...................................................... 14

8.1.2

Coder l'appel à partir de Dotnet: .................................................................................................. 14

8.2 Établir la communication javascript => Dotnet ......................................................................15

8.2.1

La WinForm doit être marquée [ComVisible]............................................................................... 15

8.2.2

Le navigateur doit être initialisé pour envoyer ses appels au Formulaire : ................................. 15

8.2.3

Une méthode publique sera exposée .......................................................................................... 16

8.2.4

Coder le HTML............................................................................................................................. 16

**9. AJOUT DE LA FONCTION DE GÉOCODAGE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_16**

9.1 Ajouter dans la déclaration <script> de la page HTML la clef récupérée de GoogleMaps ...17

|  |
| --- |
| **Page 3** |

**3**

9.2 Déclarer la variable du géocodeur ........................................................................................17

9.3 Instancier le géocodeur.........................................................................................................17

9.4 Coder la fonction JavaScript suivante :.................................................................................17

9.5 Dans la partie Dotnet : ..........................................................................................................17

**10.AJOUTER LES BOUTONS ET TEXTBOX DE CONTRÔLE \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_18**

10.1 Création de la route sous Dotnet ..........................................................................................18

10.2 La route doit être créée dans le JavaScript...........................................................................19

10.3 Ajouter le code de sauvegarde en format GPX du parcours.................................................20

**11.IDÉES D’EXTENSIONS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22**

**12.CONCLUSION \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22**

|  |
| --- |
| **Page 4** |

**4**

**Liste des Figures**

*Figure 1 : La fenêtre cadre................................................................................................................................................... 5*

*Figure 2 : Le contrôle WebBrowser activé........................................................................................................................... 6*

*Figure 3 : Affichage d’une carte par défaut ......................................................................................................................... 7*

*Figure 4 : Les contrôles Google Maps activés*

*Figure 5 : La vue satellite ..................................................................... 9*

*Figure 6 : Un marqueur sensible au ‘clic’ ......................................................................................................................... 10*

*Figure 7 : Affichage des coordonnées en temps réel .......................................................................................................... 10*

*Figure 8 : Affichage de notre Route de randonnée............................................................................................................. 12*

*Figure 9 : La distance du dernier tronçon et le cumul sont affichées................................................................................. 13*

*Figure 10 : Appel de Dotnet (C#) vers le contrôle navigateur (JavaScript)....................................................................... 15*

*Figure 11 : Appel du contrôle navigateur (JavaScript) vers Dotnet (C#) .......................................................................... 16*

*Figure 12 : Utilisation de la fonction de Géocodage ......................................................................................................... 18*

*Figure 13 : Les commandes de début et fin de route .......................................................................................................... 18*

|  |
| --- |
| **Page 5** |

**5**

**1. But de la présentation**

Nous nous proposons de montrer comment il est possible de créer une application Windows Dotnet (C#)

s’appuyant sur les APIs Google Maps (JavaScript).

On montrera en particulier les techniques nécessaires pour effectuer une communication bi-directionnelle

entre les deux langages.

L’argument sera l’affichage de cartes et la création d’itinéraires personnalisés de façon à les charger sur les

matériels GPS mobiles.

**2. La fenêtre cadre**

Nous utilisons Visual Studio pour créer la structure de l’application.

La fenêtre principale accueillera un contrôle navigateur pour l’affichage des cartes et quelques contrôles pour

piloter les options d’affichage ainsi que le tracé du parcours de la randonnée.

**2.1 Créer un projet WinForm**

Le nommer GenereRando

Redimensionner le formulaire principal à 1024-768 (size) et le nommer MainForm

Lui donner les caractéristiques d'une boite de dialogue (Taille Fixe / FormBorderStyle, pas de maximisation)

et affichage maximisé (WindowState)

Le titre : Générateur de randonnées

**Figure 1 : La fenêtre cadre**

**2.2 Ajouter un contrôle WebBrowser**

Il est automatiquement ajusté à son conteneur, le formulaire principal.

Positionner son URL sur une page HTML de test

file://<répertoire du projet>/test.html

|  |
| --- |
| **Page 6** |

**6**

Voici un exemple de page HTML :

<html>

<head><title>**Page de test**</title></head>

<body><h1>**WebBrowser affiche HTML**</h1></body>

</html>

Tester l’application

**Figure 2 : Le contrôle WebBrowser activé**

**3. Affichage d'une carte par défaut**

L’utilisation basique des APIs Google Maps est très simple :

Il s’agit d’importer une feuille de style css et le code source du javascript

http://maps.google.com/maps?file=api&v=2

Nous devons prévoir une balise HTML pour recevoir la carte et instancier celle-ci.

Enfin, nous l’initialisons sur des coordonnées (latitude, longitude) ainsi qu’un niveau de zoom.

**3.1 Coder le javascript**

Il est chargé de l'affichage d'une carte par défaut (La Défense)

<html>

<head>

<title>**Ma Page de Google Maps**</title>

<style type="text/css"> v\:\* **{**behavior**:**url(#default#VML)**; }**</style>

<script src="http://maps.google.com/maps?file=api&v=2"

type="text/javascript"></script>

<script type="text/javascript">

var **map =**null**;**

function **load() {**

|  |
| --- |
| **Page 7** |

**7**

try**{**

map = new **GMap2(document.getElementById(**"map"**));**

**}**

catch **(ex){alert(**"vous devez etre connecte a l'internet..."**);}**

if **(map !=**null**){**

//Centrer la carte sur "Paris La Défense" : zoom = 15

map.setCenter(new **GLatLng(48.8910, 2.2432) , 15);**

**}**

**}**

</script>

</head>

<body onload="load()" onunload="GUnload()">

</body>

**3.2 Coder le HTML**

Dans le corps de la page HTML, modifier le message de la page

<h1>**Dotnet utilise GoogleMaps !**</h1>

Ajouter une balise pour afficher la carte

<div id="map" style="width: 1000px; height: 600px"></div>

Tester

**Figure 3 : Affichage d’une carte par défaut**

Noter la possibilité de déplacement de la carte par glisser/déplacer du pointeur de souris.

|  |
| --- |
| **Page 8** |

**8**

**4. Activer les contrôles proposés par GoogleMaps**

L’API permet très simplement d’activer des possibilités permettant à l’utilisateur de contrôler l’affichage de sa

carte :

- déplacement de la carte par appui sur les touches directionnelles du clavier

- modification du niveau de zoom, via une échelle directement affichée sur la carte

- choix entre trois types de carte : plan, satellite ou mixte

- incrustation d’un aperçu général permettant d’accélérer la navigation

- affichage d’une cartouche représentant l’échelle de la carte

Ce sont des options de contrôle à renseigner en javascript

**4.1 Le contrôle de la position par les flèches du clavier**

Coder après l'instanciation de la 'map'

new **GKeyboardHandler(map);**//flèches clavier activées

**4.2 L’affichage du contrôle de zoom**

map.addControl(new **GLargeMapControl());**//Zoom

**4.3 L’affichage du contrôle de type de carte (plan, satellite, mixte)**

map.addControl(new **GMapTypeControl());**//Type de carte

map.setMapType(G\_NORMAL\_MAP);

**4.4 L’affichage du contrôle d'aperçu général**

map.addControl(new **GOverviewMapControl());**//Aperçu général

**4.5 L’affichage de l'échelle**

map.addControl(new **GScaleControl());**//Echelle

Tester

|  |
| --- |
| **Page 9** |

**9**

**Figure 4 : Les contrôles Google Maps activés**

**Figure 5 : La vue satellite**

**5. Personnalisation de la carte : Ajout d’un marqueur et**

**réponse aux événements ‘click’**

Il est possible de personnaliser la carte en affichant des données en surimpression.

Par exemple, nous allons marquer un emplacement par une icône.

Cet emplacement peut ensuite réagir au clic de l’utilisateur et afficher un élément HTML, ou exécuter toute

autre fonction javascript.

**5.1 Afficher une icone sur le point central de la carte**

//ajout du marqueur de depart

var **marker =**new **GMarker(**new **GLatLng(48.8910, 2.2432));**

map.addOverlay(marker);

Le marqueur est dessiné mais il est inactif lors d’un clic

**5.2 Créer un événement click sur le marqueur**

//Enregistrer l'événement 'clic' du marqueur

GEvent.addListener(marker, 'click', function**()**

**{marker.openInfoWindowHtml(**'<h2>Paris <font color="green">

La Défense</font></h2>'**);});}**

|  |
| --- |
| **Page 10** |

**10**

**Figure 6 : Un marqueur sensible au ‘clic’**

**5.3 Afficher les coordonnées du clic**

Nous désirons envoyer plus d’informations à l’utilisateur de notre carte.

Lors de chacun de ses ‘clics’, nous allons afficher les coordonnées exactes (latitude et longitude)

Coder le JavaScript

//Enregistrer l'événement 'clic' de la carte

**GEvent.addListener(map,**'click', function**(overlay,point) {**

var **latLngStr =**'latitude-longitude'**;**

if **(point){latLngStr =**'(' **+ point.x +**', ' **+ point.y +**') '**;}**

var **coordonnees = document.getElementById(**"coordonnees"**);**

**coordonnees.innerHTML = latLngStr;**

**})**

et réserver le tag HTML au bas de la carte

**coordonnées:**<span id="coordonnees"></span>

**Figure 7 : Affichage des coordonnées en temps réel**

|  |
| --- |
| **Page 11** |

**11**

**6. Tracer un itinéraire sur la carte**

L’interactivité de notre logiciel est minimale. Nous désirons à présent créer une route constituée de points

intermédiaires. Cette route devra s’afficher sur la carte, se déplacer avec elle et suivre les changements de

zoom imposés par l’utilisateur.

**6.1 Déclarer les variables route et points**

L’ajout des points intermédiaires se fera dans le gestionnaire du ‘clic’ de la carte.

Une collection des points doit être créée au niveau global du script pour mémoriser l’ensemble des points.

Coder, après la déclaration de map :

var **route =**null**;**

var **points = [];**

**6.2 Créer une route …**

Le code suivant est à ajouter dans le gestionnaire du clic de la carte

//ajouter les points à la route

if **(route) {map.removeOverlay(route);}**

**points.push(point);**

if **(points.length > 1)**

**{route =**new **GPolyline(points);map.addOverlay(route);}**

Si vous testez dès à présent, l’affichage désiré ne se produit pas…

Il faut rafraichir la carte, par exemple en sélectionnant un type de carte ou en effectuant un zoom !

**6.3 … et l'afficher**

Ceci est fait automatiquement sous Firefox

Avec IE, et donc avec le composant WebBrowser, plutôt que de forcer le rafraîchissement de la page on

codera la déclaration suivante en tête de page HTML :

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-

microsoft-com:vml">

Tester votre travail.

|  |
| --- |
| **Page 12** |

**12**

**Figure 8 : Affichage de notre Route de randonnée**

**7. Calculer les distances**

Il s’agit maintenant de fournir à l’utilisateur le moyen de connaître en temps réel la distance de sa randonnée

Nous devons mémoriser la distance totale et la position du dernier point ajouté

Créer des variables globales pour cela et ajouter au gestionnaire de clic le calcul de distance

Google Maps fournit une méthode de l’objet point nommée distance

distance = dernierPoint.distanceFrom(nouveauPoint);

**7.1 Déclarer les variables**

var **dernierPoint =**null**;**

var **totalDistance = 0;**

**7.2 Calculer :**

Remplacer l’ancien affichage des coordonnées

if **(point){latLngStr =**'(' **+ point.x +**', ' **+ point.y +**') '**;}**

|  |
| --- |
| **Page 13** |

**13**

par ...

var **distance = 0;**

if **(dernierPoint)**//calcul de la distance

**{**

**distance = dernierPoint.distanceFrom(point);**

**totalDistance = totalDistance + distance;**

**distance = Math.round(distance\*10)/10;**

**}**

**dernierPoint = point;**

if **(point){**

latLngStr = '(' **+ point.x +**', ' **+ point.y +**') ' **+ distance +**'m =>

total=' **+ Math.round(totalDistance\*10)/10 +**'m'**;}**

Tester le calcul et l’affichage de la distance

**Figure 9 : La distance du dernier tronçon et le cumul sont affichées**

|  |
| --- |
| **Page 14** |

**14**

**8. Donner à l'utilisateur Dotnet un meilleur contrôle sur la**

**carte**

Jusqu’à présent, les interactions avec l’utilisateur se limitent à la zone de responsabilité du contrôle

navigateur.

Seul le code JavaScript est concerné.

Pour ajouter d’autres fonctionnalités comme la récupération de la collection des points de la route en vue de

l’exporter en format d’échange (.GPX) sur un GPS mobile, nous devons organiser la communication entre le

JavaScript et l’environnement Dotnet (langage C#).

- contrôler le mode affichage : Simples coordonnées ou création de parcours

- réinitialiser un parcours

- récupérer en dotnet la collection des points de passage (way points)

Pour les deux premières fonctionnalités, il faut pouvoir communiquer de Dotnet vers le JavaScript.

Pour la dernière, la communication inverse doit être établie.

**8.1 Établir la communication Dotnet => javascript**

Le contrôle navigateur possède une propriété Document qui permet l’appel de script via sa méthode

InvokeScript

Les arguments sont au nombre de deux :

- un tableau d’objets

- le nom de la fonction JavaScript

***8.1.1 Dans la page HTML, créer une fonction JavaScript "affiche"***

Cette fonction est une fonction de test qui sera appelée par le code C#. Elle sera chargée d’afficher le

message envoyé par C#.

On pourra la placer après la fonction ‘load’ :

function **afficheJS(msg){alert(**"Page JavaScript :\n" **+ msg);}**

***8.1.2 Coder l'appel à partir de Dotnet:***

Placer un bouton sur le formulaire : DotnetAppelleJS et coder son gestionnaire :

private void **btnDNappelleJS\_Click(**object **sender,**EventArgs **e)**

**{**

|  |
| --- |
| **Page 15** |

**15**

object**[] args = {**"Message de Dotnet" **};**

**webBrowser.Document.InvokeScript(**"afficheJS"**, args);**

**}**

Tester la bonne communication :

**Figure 10 : Appel de Dotnet (C#) vers le contrôle navigateur (JavaScript)**

**8.2 Établir la communication javascript => Dotnet**

La communication proposée passe par un appel de la technologie « COM » de Microsoft.

Pour que l’objet Dotnet récepteur de l’appel soit joignable, il doit exposer sa classe aux librairies COM, il doit

être déclaré « ComVisible ».

Désignons le formulaire lui-même comme classe exposée (MainForm)

Le WebBrowser doit transmettre les appels du script vers cet objet formulaire (this)

Une méthode publique doit être créée pour recevoir les appels provenant de JavaScript

L’appel proprement dit utilise le modèle DOM, en particulier window.external pour accéder à la méthode

publiée précédemment.

Les étapes sont décrites ci-dessous :

***8.2.1 La WinForm doit être marquée [ComVisible]***

**[System.Runtime.InteropServices.**ComVisible**(**true**)]**

***8.2.2 Le navigateur doit être initialisé pour envoyer ses appels au Formulaire :***

On demandera la génération d’un gestionnaire pour exécuter l’initialisation de l’objet navigateur (WebBrowser)

lors du chargement de la page

private void **MainForm\_Load(**object **sender,**EventArgs **e)**

**{**

|  |
| --- |
| **Page 16** |

**16**

**webBrowser.ObjectForScripting =**this**;**

**}**

***8.2.3 Une méthode publique sera exposée***

public void **AfficheDN(**string **msg)**

**{**

MessageBox.Show(msg, "TITRE\_APPLICATION"**);**

**}**

Ceci utilise une constante de type chaîne de caractères utilisée pour donner un titre à l’application.

private static const string **TITRE\_APPLICATION**

**=**"Générateur de Randonnées"**;**

***8.2.4 Coder le HTML***

Ajouter à la page HTML, avant la ‘map’, une fonction déclenchée par un bouton

<input type="button" value="JavaScript appelle Dotnet"

onclick="window.external.AfficheDN('message JavaScript');" />

**Figure 11 : Appel du contrôle navigateur (JavaScript) vers Dotnet (C#)**

**9. Ajout de la fonction de géocodage**

Pour l’instant, il faut connaître les coordonnées (latitude et longitude) du lieu que nous désirons atteindre et

représenter sur la carte.

Pour trouver plus facilement un lieu et donc afficher la carte de ses environs, nous allons ajouter la fonction de

géocodage.

Exemple d'application : la recherche d’une carte par les noms de ville et de pays.

Cette fonctionnalité proposée par l’API nécessite l’enregistrement auprès de Google Maps et l’obtention d’une

clef associée à un site web.

|  |
| --- |
| **Page 17** |

**17**

Remarque : nous n’utilisons pas de site web ici, une clef valide suffira.

**9.1 Ajouter dans la déclaration <script> de la page HTML la clef récupérée de GoogleMaps**

<script

src="http://maps.google.com/maps?file=api&v=2&key=CLEF\_PERSONNELLE"

type="text/javascript">

</script>

**9.2 Déclarer la variable du géocodeur**

Dans la zone globale du JavaScript

var **geocoder =**null**;**

**9.3 Instancier le géocodeur**

Dans la fonction ‘load’ :

geocoder = new **GClientGeocoder();**

**9.4 Coder la fonction JavaScript suivante :**

En fin de bloc script

function **showAddress(address) {**

if **(geocoder) {geocoder.getLatLng(address,**

function**(point) {**if **(!point) {alert(address +**" not found"**); }**

else **{ map.setCenter(point, 13);**var **marker =**new

**GMarker(point);map.addOverlay(marker);marker.openInfoWindowHtml(addre**

**ss);}}**

**);}}**

**9.5 Dans la partie Dotnet :**

Réserver un peu de place au dessus du contrôle WebBrowser pour les controles dotnet

Pour cela, on changera la propriété ‘Dock’ du contrôle WebBrowser en ‘Bottom’

Ajouter des champs d'entrée Ville et Pays ainsi qu’un un bouton Géocodage

Coder l'appel à la fonction JS de géocodage sur le gestionnaire du bouton :

object**[] args = { txtVille.Text +**", " **+ txtPays.Text };**

**webBrowser.Document.InvokeScript(**"showAddress"**, args);**

Tester

|  |
| --- |
| **Page 18** |

**18**

**Figure 12 : Utilisation de la fonction de Géocodage**

**10. Ajouter les boutons et textbox de contrôle**

La dernière partie que nous nous proposons de réaliser est l’implémentation des commandes suivantes :

- Démarrage d'un parcours, pour passer en mode création

- Fin du parcours, en vue de la récupération des données pour transformation XML

**Figure 13 : Les commandes de début et fin de route**

**10.1 Création de la route sous Dotnet**

Déclaration de la route sous Dotnet

private List<PointF> route = new List<PointF>();

Ajouter à la classe MainForm une méthode appelée par JavaScript à chaque nouveau point ajouté

|  |
| --- |
| **Page 19** |

**19**

public void **AjouterPoint(**float **lat,**float **lon)**

**{**

**route.Add(**new PointF**(lat, lon));**

**}**

Ajouter les boutons btnDebutRoute et btnStopRoute ainsi que leurs gestionnaires :

private void **btnDebutRoute\_Click(**object **sender,**EventArgs **e)**

**{**

**btnFinRoute.Enabled =**true**;**

**btnDebutRoute.Enabled =**false**;**

**route.Clear();**

**webBrowser.Document.InvokeScript(**"debutRoute"**);**

**}**

et

private void **btnFinRoute\_Click(**object **sender,**EventArgs **e)**

**{**

**btnFinRoute.Enabled =**false**;**

**btnDebutRoute.Enabled =**true**;**

**webBrowser.Document.InvokeScript(**"finRoute"**);**

**}**

**10.2 La route doit être créée dans le JavaScript**

Elle doit ensuite être communiquée à la partie Dotnet

Coder en JavaScript les fonctions de début et de fin de route

var **creeRoute =**false**;**

function **debutRoute() { points=[];**

totalDistance = 0;

dernierPoint = null**;**

creeRoute = true**;**

if **(route) {map.removeOverlay(route);}**

**}**

function **finRoute() { creeRoute =**false**;}**

Dans le JavaScript, placer dans le gestionnaire de clic :

**window.external.AjouterPoint(point.x, point.y);**

Ajouter les protections pour éviter le calcul de distance si la route n'est pas demandée

|  |
| --- |
| **Page 20** |

**20**

if **(creeRoute)**

**{**

...

else

**{**

if **(point){latLngStr =**'(' **+ point.x +**', ' **+ point.y +**') '**;}**

**}**

if **(creeRoute)**

**{**

points.push(point);

...

**}**

Tester le mode affichage simple de coordonnées ainsi que le mode création de parcours.

**10.3 Ajouter le code de sauvegarde en format GPX du parcours**

Il faudra ajouter une référence à l’assembly Microsoft.Visual.Basic pour utiliser la classe Interaction

Les deux clauses using suivantes doivent être placées en tête de fichier

using **Microsoft.VisualBasic;**

using **System.Xml;**

Ajouter les lignes suivantes à la fin du gestionnaire du bouton FinRoute :

//récupérer le nom du fichier gpx

string **nom =**Interaction.InputBox("donnez un nom à votre

Route"**, TITRE\_APPLICATION,**"mon parcours"**, 10, 10);**

if **(nom.Length == 0)**

**{**

MessageBox.Show("Erreur"**, TITRE\_APPLICATION);**

return**;**

**}**

XmlTextWriter **writer =**new

XmlTextWriter**(**Application.StartupPath + "\\"+ nom+ ".gpx"**,**

Encoding.UTF8);

**writer.Formatting =**Formatting.Indented;

**writer.WriteStartElement(**"gpx"**,**

"http://www.topografix.com/GPX/1/0"**);**

**writer.WriteStartElement(**"trk"**);**

//writer.WriteElementString("name", nom);

|  |
| --- |
| **Page 21** |

**21**

**writer.WriteElementString(**"number"**,**"1"**);**

**writer.WriteStartElement(**"trkseg"**);**

//DateTime.Now.ToString();

foreach **(**PointF **pt**in **route)**

**{**

string **x = pt.X.ToString().Replace(**','**,**'.'**);**

string **y = pt.Y.ToString().Replace(**','**,**'.'**);**

**writer.WriteStartElement(**"trkpt"**);**

**writer.WriteAttributeString(**"lat"**, y);**

**writer.WriteAttributeString(**"lon"**, x);**

**writer.WriteElementString(**"ele"**,**"0.0"**);**

**writer.WriteElementString(**"time"**,**

DateTime.Now.ToString("yyyy-MM-ddThh:mm:ssZ"**));**

**writer.WriteEndElement();**//trkpt

**}**

**writer.WriteEndElement();**//trkseg

**writer.WriteEndElement();**//trk

**writer.WriteEndElement();**//gpx

**writer.Close();**

Voici un exemple de fichier résultat

<gpx xmlns="http://www.topografix.com/GPX/1/0">

<trk>

<number>1</number>

<trkseg>

<trkpt lat="48.89082" lon="2.243099">

<ele>0.0</ele>

<time>2007-10-16T04:24:51Z</time>

</trkpt>

<trkpt lat="48.89274" lon="2.237563">

<ele>0.0</ele>

<time>2007-10-16T04:24:51Z</time>

</trkpt>

<trkpt lat="48.89841" lon="2.246919">

<ele>0.0</ele>

<time>2007-10-16T04:24:51Z</time>

</trkpt>

<trkpt lat="48.89023" lon="2.256317">

<ele>0.0</ele>

<time>2007-10-16T04:24:51Z</time>

</trkpt>

<trkpt lat="48.88778" lon="2.252927">

<ele>0.0</ele>

|  |
| --- |
| **Page 22** |

**22**

<time>2007-10-16T04:24:51Z</time>

</trkpt>

<trkpt lat="48.89079" lon="2.243249">

<ele>0.0</ele>

<time>2007-10-16T04:24:51Z</time>

</trkpt>

</trkseg>

</trk>

</gpx>

**11. Idées d’extensions**

Ajouter des fonctions d’annulation (Défaire et refaire...)

Marquer les points intermédiaires avec des icones X

Rendre notre application « smart » en mettant en cache les cartes élémentaires. Elle sera ainsi utilisable en

mode déconnecté d’internet.

Permettre le chargement de parcours existants et leur affichage en vue de modification.

Donner la possibilité de modifier les points intermédiaires par glisser/déplacer

Récupérer les données d'altitude pour le calcul du dénivelé

Imposer une allure moyenne et calculer les temps de passages

Pour tout cela : http://www.google.com/apis/maps/documentation/

**12. Conclusion**

Il est aisé d’intégrer des fonctionnalités de cartographies dans une application Dotnet.

L’API Google a été créée dans le but d’être utilisée par des sites Web, mais il est simple de créer une

application WinForm profitant des mêmes fonctionnalités

La maîtrise des communications JavaScript /Dotnet aller et retour est une clef des personnalisations