Introduction à l'API Google Maps

TheCodingMachine

Par The Coding Machine - Bovino

Date de publication : 25 octobre 2011

L'API Google Maps fournit une interface intuitive et très réactive construite en utilisant les technologies AJAX. C'est une API ouverte permettant la personnalisation de la carte y compris la possibilité d'ajouter au sein de l'application des données spécifiques à la carte (personnalisation des contrôles, gestion des événements, création des marqueurs avec infobulle...). Encore mieux, Google donne accès à ce service gratuitement !
Dans cet article nous allons examiner quelques-unes des fonctionnalités de base fournies par l'API Google Maps.

Qu'est-ce qu'une API (Application Programming Interface) ?....3
Installation de l'API....3
Paramètres de l'URL....3
Google Maps avec SSL....4
Mise en place de la carte....4
Chargement asynchrone de l'API Google Maps....5
Gestion des événements Google Maps....5
Les contrôles Google Maps....6
Les marqueurs Google Maps....7
Utiliser la géolocalisation....8
Utilisation de la librairie Geometry pour le calcul de distance....9
Remerciements....9

Qu'est-ce qu'une API (Application Programming Interface) ?

Une **API** est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres.

D'un point de vue technique c'est un ensemble de fonctions, procédures ou classes mises à disposition par une bibliothèque logicielle, un système d'exploitation ou un service.

Toute application Web peut autoriser ou non des développeurs tiers à utiliser une partie de ses fonctionnalités et ce de plusieurs façons :

- en téléchargeant une bibliothèque de fonctions pour l'inclure directement sur son site ;
- en récupérant une donnée précise en construisant une URL comme avec Google charts : https://chart.googleapis.com/chart?cht=chart_type&chd=chart_data&chs=cha... ;
- comme Google Maps, en incluant directement la bibliothèque en ligne.

Installation de l'API

On va inclure la bibliothèque Google Maps directement dans le code JavaScript de la page. Cette URL permet d'accéder à l'ensemble des fonctionnalités de l'API.

Exemple d'ajout d'une balise script pour inclure les fonctions de Google Maps dans l'application cible :

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js"></script>
```

Paramètres de l'URL

Plusieurs déclinaisons de l'API sont disponibles en ajoutant des paramètres à l'URL de base ; en voici quelques exemples :

- **sensor** (true ou false) : utilisation ou non de la géolocalisation ;
- **language** : langue des textes à afficher sur la carte ;
- **region** : le pays.

On peut ajouter d'autres paramètres à l'URL pour inclure des librairies additionnelles :

- **geometry** : inclut des fonctions nécessaires pour le calcul scalaire de valeurs géométriques sur la surface de la Terre ;
- **adsense** : permet à votre application Maps d'inclure des annonces contextuelles, vous permettant de partager les revenus publicitaires pour les annonces diffusées aux utilisateurs ;
- **panoramio** :permet d'ajouter la fonctionnalité Panoramio : quand on clique sur l'icône d'une photo avec Panoramio, on a par défaut une pop-up qui s'ouvre avec des informations et la photo en plus grand.

On peut ajouter plusieurs librairies Google Maps dans l'URL, il suffit de les séparer par des virgules.

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?libraries=adsense,geometry&sensor=true"></
script>
```

On se retrouve donc avec une déclaration finale de ce type :

```
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=true"></script>
<style type="text/css">
html { height: 100% }
body { height: 100%; margin: 0px; padding: 0px }
#map_canvas { height: 100% }
</style>
</head>
```

On spécifie une taille pour les balises html et body (pour les navigateurs plus anciens il faut préciser la taille des éléments parents sinon ceux-ci supposent que la taille est de 0x0px). Ici la balise meta spécifie le mode plein écran et que la carte n'est pas redimensionnable par l'utilisateur.

Google Maps avec SSL

L'utilisation du **SSL** avec Google Maps permet d'éviter les avertissements de sécurité dans la plupart des navigateurs.
Ce genre d'utilisation est prévu pour les sites collectant des données confidentielles sur des utilisateurs (telles que la localisation personnelle ou professionnelle d'un utilisateur dans des requêtes).

Exemple d'implémentation :

```
<script type="text/javascript" src="https://maps-api-ssl.google.com/maps/api/js?v=3.4&sensor=true"></
script>
```

Mise en place de la carte

Nous allons afficher la carte une fois la page HTML chargée. Pour cela nous allons ajouter l'attribut onload à la balise body avec en paramètre la fonction d'initialisation de la carte :

```
<body onload="initialize()">
<script type="text/javascript">
   function initialize() {
       //...
   }
</script>
```

Ensuite il reste à instancier un nouvel objet google.maps.Map comme ceci :

```
function initialize() {
   map = new google.maps.Map(document.getElementById("map_canvas"), {
       zoom: 19,
       center: new google.maps.LatLng(48.8695490, 2.3513734),
       mapTypeId: google.maps.MapTypeId.ROADMAP
   });
}
```

La nouvelle carte sera contenue dans la div "*map_canvas*", centrée sur les coordonnées de latitude 48.8695490 et longitude 2.3513734 (qui représentent la position de The Coding Machine à Paris 2e) et avec un niveau de zoom très précis.

Le paramètre mapTypeId sert à définir le type de carte que l'on souhaite, il en existe quatre :

- **ROADMAP** : affiche le plan classique, sans image satellite ni relief ;

- **SATELLITE** : pour les photos satellite ;
- **HYBRID** : pour afficher les photos satellite avec le plan superposé (les routes, le nom des villes) ;
- **TERRAIN** : affiche les différences de reliefs (montagnes, rivières, etc.).

Chargement asynchrone de l'API Google Maps

Généralement, l'API est chargée au démarrage de l'application. Cela peut occasionner un ralentissement du temps d'affichage de la page. Il est possible de charger l'API Google Maps en mode asynchrone, c'est-à-dire en différé ; on utilise pour cela le paramètre callback dans l'URL qui appellera la fonction d'initialisation de la carte.

```
function initialisation(){
   var maLatlng = new google.maps.LatLng(48.8695490, 2.3513734);
   var mesOptions = {
       zoom: 8,
       center: maLatlng,
       mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);
}
// Création de la balise script pour inclure les fonctions de Google Maps dans notre application
function loadScript() {
   var script = document.createElement("script");
   script.type = "text/javascript";
   script.src = "http://maps.google.com/maps/api/js?sensor=false&callback=initialisation";
   document.body.appendChild(script);
}
// Au chargement de la page on appelle la fonction loadScript qui appelle la fonction initialisation() grâce au p
window.onload = loadScript;
```

Gestion des événements Google Maps

Il est possible de capturer un événement sur un marqueur ou sur la carte grâce à la méthode addListener(). Il existe différents types d'événements simples :

- 'click' ;
- 'dblclick' ;
- 'mouseup' ;
- 'mousedown' ;
- 'mouseover' ;
- 'mouseout' ;
- zoom_changed.

Ces événements ressemblent à des événements DOM standard, mais ils font en réalité partie intégrante de l'API Google Maps.

```
var carte;
function initialisation(){
    // On rentre les coordonnées(latitude, longitude) de notre choix dans une variable
    var maLatlng = new google.maps.LatLng(-25.363882,131.044922);

    // On établit les options de notre choix :

  // la profondeur du zoom, les coordonnées sur lesquelles la carte sera centrée, le type de vue (satellite, plan
    var mesOptions = {
       zoom: 4,
       center: maLatlng,
       mapTypeId: google.maps.MapTypeId.ROADMAP
    }

    // On crée notre carte en lui passant toutes nos options en paramètre
```

```
    carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);


  // On place un listener sur la carte qui contrôle une action qui sera déclenchée lors de l'événement 'zoom_chan
    // Quand le zoom sera modifié la carte sera recentrée sur les coordonnées de The Coding Machine
    google.maps.event.addListener(carte,'zoom_changed', function() {
       setTimeout(allerChezTCM,3000);
    });

    // On crée un marqueur que l'on positionne grâce au paramètre "position"
    var marker = new google.maps.Marker ({
       position: maLatlng,
       map: carte,
       title: "Hello world :) !"
    });


  // On place un listener sur le marqueur qui contrôle une action qui sera déclenchée lors de l'évènement 'click'
    // Quand on clique sur le marqueur, le zoom de la carte passera à 8
    google.maps.event.addListener(marker, 'click', function(){
       carte.setZoom(8);
    });
}
function allerChezTCM() {
   var tcm = new google.maps.Latlng(48.8695490, 2.3513734);
   map.setCenter(tcm);
}
```

Les contrôles Google Maps

Il est possible d'activer ou de désactiver un contrôle en changeant la valeur de sa propriété à true ou false :

```
{
   panControl: boolean,
   zoomControl: boolean,
   mapTypeControl: boolean,
   scaleControl: boolean,
   streetViewControl: boolean,
   overviewMapControl: boolean
}
```

On peut également modifier les options d'un contrôle (position et style)

```
zoomControl: true,
zoomControlOptions: {
   style: google.maps.ZoomControlStyle.LARGE,
   position: google.maps.ControlPosition.LEFT_CENTER
},
```

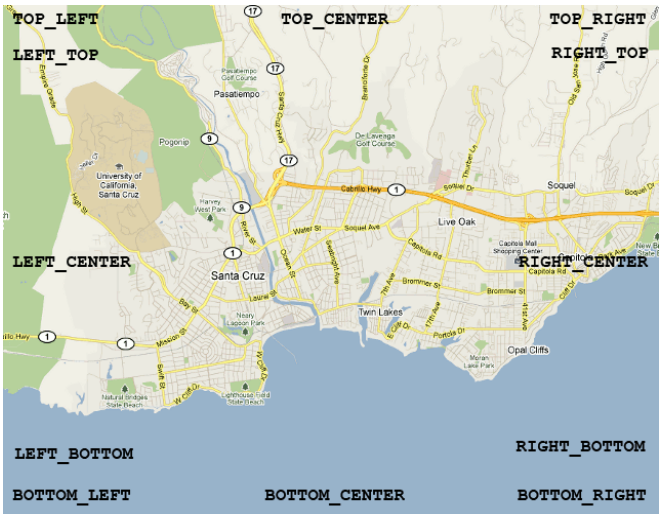
En voici quelques exemples :

- ZoomControlStyle.LARGE : zoom standard avec le slider ;
- ZoomControlStyle.SMALL : affiche un mini zoom avec juste les icônes + et - ;
- ZoomControlStyle.DEFAULT : choisit le bon format de zoom en fonction de taille de la carte du type d'appareil (mobile ou non) ;
- MapTypeControlStyle.HORIZONTAL_BAR : affiche une barre horizontale pour la sélection du type d'affichage de la carte ;
- MapTypeControlStyle.DROPDOWN_MENU : permet de choisir le type de carte (plan, satellite, terrain, hybride) via une liste déroulante ;
- MapTypeControlStyle.DEFAULT : comportement par défaut, dépend de la taille de l'écran.

Exemple de configuration d'un contrôle :

```
//On établit une liste déroulante pour le contrôle MapType et on spécifie que le Zoom control utilise un minizoom
function initialize() {
   var myOptions = {
        zoom: 4,
        center: new google.maps.LatLng(-33, 151),
        mapTypeControl: true,
        mapTypeControlOptions: {
            style: google.maps.MapTypeControlStyle.DROPDOWN_MENU
        },
        zoomControl: true,
        zoomControlOptions: {
            style: google.maps.ZoomControlStyle.SMALL
        },
        mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
}
```

Voici toutes les positions possibles d'un contrôle sur une carte :



Les marqueurs Google Maps

Voici comment créer un marqueur simple avec un contenu info bulle HTML :

```
function initialize() {
   var maLatlng = new google.maps.LatLng(48.8695490, 2.3513734);
   var mesOptions = {
        zoom: 4,
        center: myLatlng,
        mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);
   var image = 'beachflag.png';
   // Instanciation de notre marqueur
   var marker = new google.maps.Marker({
        position: maLatLng,
        map: carte,
        icon: image
   });
   var message = "Vous êtes ici !";
   var infowindow = new google.maps.InfoWindow({
        content: message,
        size: new google.maps.Size(50,50)
   });
   google.maps.event.addListener(marker, 'click', function() {
```

```
        infowindow.open(carte,marker);
    });
}
```

Les marqueurs sont personnalisables à l'aide de la propriété icon, il suffit de lui passer une image en paramètre.

En bonus, il est possible d'ajouter une animation aux marqueurs en appelant la méthode setAnimation() sur l'objet marker. Deux types sont possibles :

- **DROP** : indique que ce marqueur devrait tomber du haut de la carte jusqu'à son emplacement définitif quand il est affiché sur la carte la première fois. L'animation cesse une fois que le curseur est en place. Ce type d'animation est généralement spécifié lors de la création du marqueur ;
- **BOUNCE** : indique que le marqueur doit remuer. Un marqueur de ce type va continuer de bouger jusqu'à ce que sa propriété soit explicitement désactivée (= null).

```
var stockholm = new google.maps.LatLng(59.32522, 18.07002);
var tcm = new google.maps.LatLng(48.8695490, 2.3513734);
var marker;
var map;
function initialize() {
    var mapOptions = {
        zoom: 13,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        center: stockholm
    };
    map = new google.maps.Map(document.getElementById("map_canvas"), mapOptions);
    marker = new google.maps.Marker({
        map:map,
        draggable:true,
        animation: google.maps.Animation.DROP,
        position: tcm
    });
    google.maps.event.addListener(marker, 'click', toggleBounce);
}

function toggleBounce() {
    if (marker.getAnimation() != null) {
        marker.setAnimation(null);
    } else {
        marker.setAnimation(google.maps.Animation.BOUNCE);
    }
}
```

Utiliser la géolocalisation

```
// On teste tout d'abord si le navigateur prend en charge la géolocalisation HTML5
if (navigator.geolocation){
    var watchId = navigator.geolocation.watchPosition(
        successCallback,
        null,
        {enableHighAccuracy:true}
    );
}
else{
    alert("Votre navigateur n'est pas compatible avec la géolocalisation HTML 5");
}
function successCallback(position){
    map.panTo(new google.maps.Latlng(position.coords.latitude, position.coords.longitude));
    var marker = new google.maps.Marker({
        position: new google.maps.Latlng(position.coords.latitude, position.coords.longitude),
        map: map
    });
}
```

À chaque déplacement, un nouveau marqueur est placé ; dans le cas d'une perte de signal le marqueur se positionne au relais 3G le plus proche.

Pour contourner le problème on utilise la propriété accuracy :

```
if(position.coords.accuracy < 100) {
   //...
}
```

Utilisation de la librairie Geometry pour le calcul de distance

Tout ce que vous avez à faire est d'inclure la librairie *Geometry* lors de l'intégration de l'API à l'application, puis d'appeler la méthode computeDistanceBetween() et lui passer deux objets LatLng en paramètres :

```
var nyc = new google.maps.LatLng(40.715, -74.002);
var london = new google.maps.LatLng(51.506, -0.119);
var distance = google.maps.geometry.spherical.computeDistanceBetween(nyc, london);
```

Remerciements

Cet article a été publié avec l'aimable autorisation de **The Coding Machine**, l'article original peut être vu **sur le blog de The Coding Machine**.

Nous tenons à remercier **ClaudeLELOUP**, **kdmbella** et **jacques_jean** pour leur relecture attentive de cet article.