Intégration d’interfaces de géolocalisation dans un site web avec google maps api et jquery

La géolocalisation est à la mode ! Elle est de plus en plus utilisée sur le web, dans divers contextes : e-commerce, sites corporate, site communautaires, annuaires. Quelle utilisation ? Dans quels buts et par quels moyens ?

Lors des étapes préliminaires de l’élaboration ou de l’évolution de sites web, beaucoup se voient intégrer des cartes dans leur site web afin de permettre à l’utilisateur de repérer et trouver facilement une (ou plusieurs) adresse(s). Qu’en est-il de l’implémentation pratique et de sa complexité ?

Objectifs recherchés et technologies disponibles :

Aujourd’hui, la praticité de l’affichage géolocalisé n’est plus à démontrer. Lorsque l’on souhaite se rendre quelque part, cartes, plans et guides, sont rendus caduques par les services de géolocalisation, que ce soit le web ou sur un appareil équipé de GPS. De manière proactive, les sites web proposent de plus en plus la géolocalisation intégrée dans leur site aussitôt que possible, évitant ainsi à l’utilisateur d’avoir à ouvrir un site ou un plan papier pour repérer une adresse.

Voici quelques exemples typiques de géolocalisation d’adresses dont on apprécie l’intégration sur un site :

- corporate : implantations, locaux, offices,
- magasins : boutiques, points de vente,
- communautés : bonnes adresses, localisation des membres.

Les fonctionnalités attendues de géolocalisation peuvent être plus ou moins élaborées :

1. statique : affichage d’adresses dans une image,
2. repérage : adresses dans une carte dont on peut contrôler le déplacement et le zoom,
3. exploration : carte qui peut agréger des adresses par niveaux de détail (par ex. pays, ville, quartier).

Les principaux fournisseurs pour ce type de service permettent au moins la cartographie, souvent gratuitement, avec des modes « plan » et « relief » et (Google Maps, Yahoo Maps, Live Search Maps, etc). Selon le pays, les base de données des fournisseurs peuvent enrichir la carte par diverses informations : POI (Points Of Interest), informations routières, etc. Mais ce sont alors souvent des services payants (Mappy, ViaMichelin, etc).

Selon les besoins, la problématique de contribution d’adresses peut-être prise en compte. Le contributeur doit alors être en mesure d’ajouter facilement une adresse, par informations de localisation (numéro, voie, ville) ou en plaçant lui-même le curseur à l’endroit précis où se trouve le lieu qu’il souhaite ajouter. Ceci requiert aux formulaires de contribution, privés ou publics, de présenter en plus des champs textes classiques une carte sur laquelle le contributeur disposera également de fonctions de géolocalisation.

Ces choix déterminent la complexité et la charge de développement lors de la mise en œuvre de la carte. Dans le cadre du projet qui nous sert d’exemple, les besoins sont les suivants :

- repérage de bonnes adresses autour d’une localité (avec zoom et déplacement),
- deux affichages : carte générale et localisation d’une adresse,
- possibilité pour les utilisateurs de saisir leurs adresses.

De la conception à l'implémentation

Conception

Nous cherchons à stocker les informations de géolocalisation pour chaque adresse, de manière aussi précise et interopérable que possible. Pour cela, l'adresse nous "aide" à trouver le point que nous voulons indiquer sur la carte. Mais pour des raisons évidentes, l'adresse (au format texte) n'est ni fiable ni durable dans le temps : il faut donc également sauver les coordonnées cartographiques du lieu à enregistrer.

Par ailleurs, en dehors des données objectives de localisation (latitude / longitude), il faut également laisser au contributeur la possibilité de choisir la portion de carte qui s'affiche, et avec quel niveau de zoom.

Ainsi, notre structure de données est :

- nom du lieu,
- adresse (éventuellement décomposée en numéro, voie, code postal, pays),
- latitude,
- longitude,
- zoom,
- [description, commentaires, etc - selon les besoins].

Implémentation à l'aide de Google Maps

Dans le cadre du projet, nous devons intégrer la carte sur trois pages distinctes :

- page de contribution d'une adresse,
- page d'affichage de l'ensemble des adresses,
- page d'affichage du détail d'une adresse.

A partir du moment où nous nous sommes procuré une clé Google Maps (gratuite, jusque 15.000 requêtes par jour) - ici notée `[clé]` - le code pour charger l'API est une simple inclusion de la librairie mise à disposition par Google. La carte peut alors être chargée avec le code suivant :

```
<script type="text/javascript" src="http://maps.google.com/maps?
file=api&v=2&key=[clé]"></script>
<div id="map-canvas"></div>
<script type="text/javascript">
if (GBrowserIsCompatible()) {
        map = new Gmap2(document.getElementById('map-canvas'),
{size: new GSize(500, 440)});
}
</script>
```

Cet exemple est le code minimal pour initialiser la librairie, mais la carte ne s'affichera que lorsqu'on lui aura donné les coordonnées du point à afficher. Par ailleurs, la carte est ici insérée dans le div *#map-canvas*, qui est initialement caché (cf plus bas) : il vaut mieux en préciser les dimensions par l'objet *Gsize*, afin qu'elle se charge correctement.

Nous implémentons une première fonction Javascript, *mapMarkLatLng* qui place un marqueur sur une coordonnée géographique et centre la carte dessus :

```
function mapMarkLatLng (latLng, centerMap) {
        map.setCenter(latLng);
        mapMarker = new Gmarker(latLng);
        map.addOverlay(mapMarker);
}
```

Ensuite, nous préparons également une fonction *mapMarkAdress* qui détermine les coordonnées d'une adresse à l'aide du service *Geocoder* puis marque cette adresse par appel de la fonction précédente :

```
function mapMarkAdress (address) {
        mapGeocoder = new GClientGeocoder();
        mapGeocoder.getLatLng(address, function(latLng) {
                if (latLng) mapMarkLatLng(latLng);
        });
}
```

Alors, le code Javascript suivant nous permet d'afficher une adresse dans notre carte :

```
mapMarkAdress('10 boulevard des Capucines, Paris, France');
```

Après quelques améliorations et une adaptation du code selon les besoins, voici ce que peut donner l'intégration et l'habillage d'une carte dans un site web standard, avec quelques marqueurs issus de la base de données :

Implémentation de jQuery

JQuery est une librairie Javascript dont Clever Age a déjà parlé, qui est très répandue et combine à la fois d'excellentes performances et une grande richesse fonctionnelle. Ceci dit, le code présenté ici pourrait tout-à-fait être réalisé à l'aide d'autres toolkits (Dojo, Prototype, YUI...), sans plus de difficultés. Dans tous les cas, des adaptations au look & feel du site seront nécessaires (uniquement par ajout de classes CSS).

La fonctionnalité de boîte de dialogue est mise à notre disposition dans la package jQuery UI (qui nécessite l'emploi du plugin jQuery dimensions). Afin de rendre notre carte dans une boite de dialogue (cachée à l'initialisation), nous utilisons simplement le code HTML suivant :

```
<div id="map-canvas-dialog" style="display: none;">
        <div id="map-canvas"></div>
</div>
```

Et pour transformer ce nœud HTML en boite de dialogue, voici le code Javascript correspondant :

```
$(document).ready( function () {
        $('#map-canvas-dialog').dialog( {
                buttons: { 'Ok': function() { $
(this).dialog('close'); } },
                autoOpen: false,
                resizable: false,
                title: 'Titre de la carte',
                width: 500,
                height: 500,
                dialogClass: 'dialogCssClass',
        } );
        $('#map-canvas-dialog').show();
} );
```

Par la suite, il faudra simplement implémenter les fonctions d'ouverture *$('#map-canvas-dialog').dialog('open') ;* et de fermeture *$('#map-canvas-dialog').dialog('close') ;* de cette boite de dialogue selon les évènements de l'application, ainsi que d'éventuelles actions lors de la fermeture de la boite de dialogue (l'attribut *buttons* est dédié à cela).

Mise en pratique

Bien entendu, ce ne sont là que les briques essentielles nous permettant ensuite de construire notre interface de géolocalisation. Dans la pratique, ce code, jusqu'ici très concis, doit être personnalisé selon le site web (ou l'application) dans lequel s'intègrent nos cartes.

Dans le cas spécifique de notre implémentation, sont effectuées les opérations suivantes :

- chargement de la page avec les librairies et le nœud HTML de la carte (caché),
- initialisation de la carte et mise sous forme de boite de dialogue,
- centrage de la carte sur un point, selon la page dans laquelle est intégrée la carte :
    - nouvelle adresse : coordonnées de l'adresse saisie ou point arbitraire,
    - ensemble d'adresses : coordonnée décidé par le contributeur ou centre des adresses,
    - détail d'une adresse : coordonnée saisi par le contributeur,
- affichage de marqueurs, à nouveau selon la page :
    - nouvelle adresse : marqueur déplaçable,
    - ensemble d'adresses : un marqueur au centre et plusieurs marqueurs pour les adresses (autre couleur),
    - détail d'une adresse : un seul marqueur sur l'adresse concernée,
- mise en place des évènements qui provoqueront l'ouverture et la fermeture de la boite de dialogue.

Tout ceci met donc en jeu des interactions entre la page web, Google Maps API et les librairies jQuery, entièrement en Javascript, qui nous permettent d'intégrer et de manipuler des cartes, selon nos besoins et sans surcharger les interfaces et le site web.

La contribution d'une adresse

C'est là le petit plus, qui peut faire la différence lors de l'utilisation de votre application au quotidien. En effet, lors de la contribution d'adresses, de trop nombreuses applications ont tendance à enregistrer les données de l'adresse uniquement. Si cependant Google Maps est insuffisamment précis ou renvoie des coordonnées erronées pour une adresse donnée, le contributeur devrait avoir la possibilité de placer lui-même le marqueur sur la carte. De plus, il est utile de spécifier le niveau de zoom qui paraît le plus adapté, dépendant de l'adresse qu'il affiche : le contributeur garde le contrôle sur ce qui ses affiché.

Ainsi, nous donnons le contributeur pourra ouvrir la carte en popup, pour laquelle nous mettons en place les mécanismes suivants :

- lors de l'ouverture selon les champs textes :
    - s'ils ont été remplis, la carte est centrée sur les coordonnées renvoyées par le *Geocoder* de Google Maps,
    - sinon, la carte est arbitrairement centrée sur '*France*',
- à tout moment, le contributeur peut :
    - déplacer le marqueur, zoomer ou dézoomer,
    - entrer une adresse dans un champ libre à chercher (pour faciliter la recherche),
- lors de la validation, seront transmis pour être enregistrés :
    - les coordonnées,
    - le niveau de zoom,
    - les autres champs du formulaire.

Cette interface de contribution permet à l'utilisateur de véritablement contrôler ce qui s'affiche dans la carte, là aussi en évitant de surcharger l'interface de saisie, grâce au principe des boîtes de dialogue jQuery. L'atout de ce type d'interface réside dans le fait de pouvoir localiser, facilement et visuellement, une adresse sur la carte, tout en s'assurant que le niveau de zoom permettra, lors de l'affichage, de repérer aisément le lieu localisé.

Conclusion

Nous voyons qu'en utilisant les bons outils, le développement d'une interface de géolocalisation en javascript rencontre finalement peu de difficultés, et rivalise largement avec les intégrations d'images ou d'iframes. Et, contrairement à ce que l'on pourrait croire, la contribution d'adresses dans le frontend se met également en place sans difficultés et peut apporter beaucoup, notamment aux sites communautaires.

Mais ce type d'interfaces vient bousculer la navigation traditionnelle d'un site. Lorsque la base de données d'adresses est très fournie, lorsque l'on souhaite enrichir les affichages (informations routières, moyens de transports, lieux intéressants), d'autres problématiques se présentent : paginations inadaptées, agrégation des résultats, choix du type d'information à afficher... tout ceci à étudier dans les futurs billets du blog de Clever Age !

Auteurs : Damien Nouvel, Stéphane Prohaszka
Tags : XML, Javascript, jQuery, geolocalisation