

Manuel programmeur

Le code se répartit sur 2 fichiers : "index.php" et "recherche.php". Ces deux codes sont très similaires car "recherche.php" est à la base un copier/coller d'"index.php".

Nous présenterons donc d'abord ce que les deux pages ont en commun puis leurs particularités.

Parties communes

Outre le CSS, cette partie concerne l'interrogation de la base de données, l'extraction des données, l'initialisation de la carte et le MarkerClusterer.

Chacune des pages contient le formulaire qui permet de recueillir la recherche de l'utilisateur :

```

74      <!-- formulaire -->
75      <fieldset><center>
76          <legend>Recherche dans la BDD</legend><br/>
77          <form method="get" action="recherche.php">
78              <label>Numéro : <input type="text" name="numero" value="" /></label>
79              <label>Voirie : <input type="text" name="voirie" value="" /></label>
80              <label>Code postal : <input type="text" name="code_postal" value="" /></label>
81              <label>Ville : <input type="text" name="ville" value="" /></label>
82              <label>Pays : <input type="text" name="pays" value="" /></label>
83              <input type="submit" value="Envoyer" />
84          </form>
85      </center></fieldset>

```

La connexion à la base de données s'effectue grâce à l'extension pgsql :

```

86      <?php
87      // phpinfo(); // informations sur le PHP charge
88      //echo "<div><p><center>Chargement de l'extension 'pgsql' : <strong>";
89      echo extension_loaded('pgsql') ? 'Yes' : 'No'; // test by adding this in your index.php
90      echo "<strong><center><p></div><br/>";
91      $menu_choix = extension_loaded('pgsql');
92      // SWITCH
93      switch ($menu_choix){ // menu en fonction du chargement de l'extension
94          case true: // si l'extension pgsql a été chargée
95              $link = pg_connect("host=localhost port=5432 dbname=adresse_geocodage user=postgres password="); // host=localhost port=5432
96              //echo "<div><p><center>Erreur de connexion : <strong>"; // print pg_last_error($link) . "<strong><center><p></div><br/>"; // afficher l'
97              // erreur de connexion
98          }else{
99              //print "<div><p><center>Connexion réussie au port : <strong>"; // print pg_host($link) . "<strong><center><p></div><br/>"; // affiche le succès
100              // de la connexion
101              // EXTRACTION de la BDD des adresses
102              $result_adresse = pg_query($link, "SELECT * FROM adresse"); // requête SQL
103              if (!$result_adresse) { // si la requête n'a pas aboutie
104                  //echo "<div><p><center><strong>Erreur de requête SQL (adresse) !<strong><center><p></div><br/>"; // affichage du warning
105              }
106              else{
107                  //echo "<div><p><center><strong>Nombre de lignes du résultat de la requête (SELECT * FROM adresse) : ";
108                  echo pg_num_rows($result_adresse); // nombre de lignes du résultat de la requête
109                  echo "<strong><center><p></div><br/>";
110                  $nb_ligne_adr = pg_num_rows($result_adresse); // stocke le nombre de lignes du résultat de la requête
111                  $i = 1;
112                  //echo "<table class='tbl_bdd'>";
113                  echo "<tr><th>Adresse n°</th><th>Numéro</th><th>Voie</th><th>Code postal</th><th>Ville</th><th>Pays</th></tr>"; // entête/
114                  while($row = pg_fetch_row($result_adresse)){ // on accède aux valeurs de la ligne par leur indice dans le tableau

```

Vous noterez que le code prend en compte le chargement de l'extension pgsql par le biais d'un switch :

```

189      if(!pg_close($link)){ // cette fonction retourne TRUE en cas de succès ou FALSE si une erreur survient
190          //print "<div><p><center>Impossible de fermer la connexion au port " . pg_host($link) . " : <strong>"; // print pg_last_error($link) . "<strong><
191          //center><p></div><br/>";
192      }else{
193          //print "<div><p><center>Déconnexion de la base de données réussie !<center><p></div><br/>";
194      }
195      break;
196      case false: // si l'extension pgsql n'a pas été chargée
197          // STOCKAGE des adresses
198          $row_adr[1][1] = 1;
199          $row_adr[1][2] = 15;
200          $row_adr[1][3] = "Boulevard Copernic";
201          $row_adr[1][4] = "77420";
202          $row_adr[1][5] = "CHAMPS-SUR-MARNE";
203          $row_adr[1][6] = "FRANCE";
204          ///////////////

```

Ce qui veut dire que les données sont chargées de 2 manières. Soit vous avez réussi à charger l'extension et vous interrogez la base de données pour récupérer les adresses et les géocodages, soit vous n'avez pas chargé l'extension et vous rentrez les mêmes données mais en brut.

C'est moche mais ça marche !

La plupart des mises en commentaires ne sont que des echos de tableaux, reflétant le contenu extrait de la base de données, et de leurs paramètres.

Que ce soit le cas 1 (true) ou le cas 2 (deux), les informations extraites de la base de données sont stockées dans des variables portant le même nom.

Et pour finir, l'accès à la page "recherche.php" s'effectue lors de l'envoi du formulaire :

```
77 <form method="get" action="recherche.php">
```

Voilà pour la partie PHP. Passons maintenant au JavaScript.

Outre la fonction afficher() qui est appelée lorsqu'on clique sur le "<h2>Comparateur de web services de géocodage - la consultation</h2>", on commence par la fonction initialize() qui permet l'affichage de la carte et est appelée à chaque chargement du "<body>". On reviendra dessus car elle est propre à chaque page.

Pour ce qui est du MarkerClusterer, il faut appelé le script tout comme celui de l'API Google :

```
36 <!-- JAVASCRIPT -->
37 <script type="text/javascript" src="js-marker-clusterer-gh-pages/src/markerclusterer.js"></script> <!-- script MARKERCLUSTER -->
38 </script>
354 <!-- JAVASCRIPT -->
355 <script src="https://maps.googleapis.com/maps/api/js?v=3.exp"> // chargement de l'API Google
356 </script>
```

Cela nécessite de bien vérifier les chemins d'accès et sa bonne installation.

Le MarkerCluster fonctionne comme suit :

1. Création d'une variable globale et d'un tableau de markers :

```
368 var markerCluster; // variable du MARKERCLUSTERER
369 var markers = []; // tableau des markers pour le MARKERCLUSTERER
```

2. Dans la fonction addMarker(location,map,bulle), création d'un marker et ajout de celui-ci dans le tableau de markers :

```
376 var marker = new google.maps.Marker({ // nouveau marker
377   position: location,
378   label: "",
379   map: map
380 });
381 markers.push(marker); // ajoute le marker au tableau des markers pour le MARKERCLUSTERER
```

3. On finit la fonction initialize() par la création du MarkerClusterer qui prend en entrée la carte google et la liste des markers :

```
65 markerCluster = new MarkerClusterer(map, markers); // creation du MARKERCLUSTERER
```

L'emplacement de chaque variable et de chaque fonction dans le JavaScript est très important pour le bon fonctionnement du code !

index.php

Par rapport au fichier "recherche.php", la fonction initialize() est plus simple. Pour chaque adresse de la base de données, on cherche ses géocodages qui nous serviront à placer les markers :

```

42  var map; // variable pour la carte google
43  function initialize(){ // lors du chargement de la page
44      var latlng = new google.maps.LatLng(48.849335,2.607764); // coordonnees a charger
45      var mapOptions = {
46          zoom:14, // zoom a charger
47          center:latlng
48      }
49      map = new google.maps.Map(document.getElementById("map"),mapOptions); // carte Google
50      // affichage de toutes les coordonnees
51      for(var i=1;i<nb_ligne_geo;i++){ // parcours des geocodages
52          var loc = { lat: parseFloat(row_geo[i][3]), lng: parseFloat(row_geo[i][4]) }; // { lat: 48.843336, lng: 2.622286 }; // parseFloat() =>
53          // converti en float
54          console.log(loc); // affiche la localisation
55          for(var j=1;j<nb_ligne_adr;j++){ // cherche 1 adresse correspondante
56              //console.log(row_geo[i][1]);
57              //console.log(row_adr[j][1]);
58              if(row_geo[i][1]==row_adr[j][1]){ // s il s agit de 1 adresse
59                  var bulle = row_adr[j][2]+" "+row_adr[j][3]+" "+row_adr[j][4]+" "+row_adr[j][5]+" "+row_adr[j][6]; // numero voieie code_postal
60                  // commune pays
61                  console.log(bulle); // affiche 1 info bulle
62                  break; // si tout est bon
63              }
64          }
65          addMarker(loc,map,bulle); // ajout du marker
66      }
67      markerCluster = new MarkerClusterer(map, markers); // creation du MARKERCLUSTERER

```

C'est la fonction addMarker(loc,map,bulle) qui ajoute le marker et gère l'info bulle laquelle n'affichera que l'adresse du marker :

```

370  function addMarker(location,map,bulle){ // Adds a marker to the map.
371      // Add the marker at the clicked location, and add the next-available label
372      // from the array of alphabetical characters.
373      var infowindows = new google.maps.InfoWindow({ // nouvelle info bulle
374          content: bulle
375      });
376      var marker = new google.maps.Marker({ // nouveau marker
377          position: location,
378          label: "",
379          map: map
380      });
381      markers.push(marker); // ajoute le marker au tableau des markers pour le MARKERCLUSTERER
382      marker.addListener('click',function(){ // affiche 1 info bulle au clic
383          infowindows.open(map,marker); // ouvrir 1 info bulle
384      })
385  }

```

Certaines variables PHP extraites de la base de données sont converties en variable JS, on notera l'utilisation du JSON pour les tableaux :

```

358  // VARIABLE PHP
359  var nb_ligne_adr = <?php echo $nb_ligne_adr; ?>; // nombre de lignes dans la table adresse
360  console.log("Nombre de ligne du tableau des adresses : " + nb_ligne_adr);
361  var row_adr = <?php echo json_encode($row_adr); ?>; // tableau des adresses
362  console.log(row_adr);
363  var nb_ligne_geo = <?php echo $nb_ligne_geo; ?>; // nombre de lignes dans la table geocodage
364  console.log("Nombre de ligne du tableau des géocodages : " + nb_ligne_geo);
365  var row_geo = <?php echo json_encode($row_geo); ?>; // tableau des geocodages
366  console.log(row_geo); // row_geo[1][2]

```

recherche.php

Pour la partie PHP ce qui change est le traitement de la recherche envoyée par la méthode GET :

```

376 $numero = $_GET['numero'];
377 $voirie = $_GET['voirie'];
378 $code_postal = $_GET['code_postal'];
379 $ville = $_GET['ville'];
380 $pays = $_GET['pays'];
381 $affichage = false; // permission pour afficher le resultat sur la carte
382 // affichage de la recherche
383 echo "<div><p><center>Votre recherche est : <strong>";
384 echo $numero;
385 echo " ";
386 echo $voirie;
387 echo " ";
388 echo $code_postal;
389 echo " ";
390 echo $ville;
391 echo " ";
392 echo $pays;
393 echo "<strong><center></p></div>";
394 // affichage de la reponse
395 for($i=1;$i<=$nb_ligne_adr;$i++){ // parcourt les adresses
396     if($row_adr[$i][6]==$pays){ // si le pays existe
397         if($row_adr[$i][5]==$ville){ // si la ville existe
398             if($row_adr[$i][4]==$code_postal){ // si le code postal existe
399                 if($row_adr[$i][3]==$voirie){ // si la voirie existe
400                     if($row_adr[$i][2]==$numero){ // si le numero existe
401                         echo "<div><p id='adr_find'><center>Adresse n°";
402                         echo $row_adr[$i][1];
403                         echo " trouvée : <strong>";
404                         echo $row_adr[$i][2];
405                         echo " ";
406                         echo $row_adr[$i][3];
407                         echo " ";
408                         echo $row_adr[$i][4];
409                         echo " ";
410                         echo $row_adr[$i][5];
411                         echo " ";
412                         echo $row_adr[$i][6];
413                         echo "<strong><center></p></div><br>";
414                         $number = $row_adr[$i][1]; // numero de l adresse trouvee
415                         $affichage = true; // permission pour afficher le resultat sur la carte
416                         break; // pour le bon chargement des script JS !
417                     }elseif($i==$nb_ligne_adr){ // sinon si on a fini de parcourir les adresses
418                         echo "<div><p><center><strong>Adresse introuvable dans la BDD !<strong><center></p></div>";
419                         break; // pour le bon chargement des script JS !
420                     }
421                 }elseif($i==$nb_ligne_adr){ // sinon si on a fini de parcourir les adresses
422                     echo "<div><p><center><strong>Adresse introuvable dans la BDD !<strong><center></p></div>";
423                     break; // pour le bon chargement des script JS !
424                 }
425             }elseif($i==$nb_ligne_adr){ // sinon si on a fini de parcourir les adresses
426                 echo "<div><p><center><strong>Adresse introuvable dans la BDD !<strong><center></p></div>";
427                 break; // pour le bon chargement des script JS !
428             }
429         }elseif($i==$nb_ligne_adr){ // sinon si on a fini de parcourir les adresses
430             echo "<div><p><center><strong>Adresse introuvable dans la BDD !<strong><center></p></div>";
431             break; // pour le bon chargement des script JS !
432         }
433     }elseif($i==$nb_ligne_adr){ // sinon si on a fini de parcourir les adresses
434         echo "<div><p><center><strong>Adresse introuvable dans la BDD !<strong><center></p></div>";
435         break; // pour le bon chargement des script JS !
436     }
437 }

```

C'est tout pour le PHP, passons maintenant au JS.

Si la recherche trouve une adresse, on récupère le numéro de celle-ci qui nous servira d'identifiant pour la suite :

```

443 // VARIABLE PHP
444 var affichage = <?php echo $affichage; ?> // permission pour afficher le resultat sur la carte
445 if(affichage){ // transfert/voirie JS
446     var number = <?php echo $number; ?> // converti le numero de l adresse
447     console.log(number); // affiche le numero de l adresse
448     /*var numero = <?php echo $numero; ?>
449     console.log(numero);
450     var voirie = <?php echo $voirie; ?>
451     console.log(voirie);
452     var code_postal = <?php echo $code_postal; ?>
453     console.log(code_postal);
454     var ville = <?php echo $ville; ?>
455     console.log(ville);
456     var pays = <?php echo $pays; ?>
457     console.log(pays);*/
458 }

```

Le transfert de variable de PHP à JS s'effectue avant la validation de la recherche et ne concerne que le tableau des géocodages :

```

365 <script type="text/javascript">
366 // VARIABLE PHP a placer ici pour le bon chargement
367 var nb_ligne_geo = <?php echo $nb_ligne_geo; ?>; // nombre de lignes dans la table geocodage
368 console.log("Nombre de ligne du tableau des géocodages : " + nb_ligne_geo);
369 var row_geo = <?php echo json_encode($row_geo); ?>; // tableau des geocodages
370 console.log(row_geo); // row_geo[1][2]
371 var number = null; // pour l'initialisation de la carte
372 console.log(number);
373 </script>

```

La fonction initialize() est un peu plus complexe. En effet dans un premier temps, on va déterminer l'envergure, l'emprise de la carte. Si on a une adresse on se focalise dessus sinon on prend l'emprise de base. On crée la carte puis pour chaque géocodage lié à l'adresse, on affiche un marker et son info-bulle. Le tout est envoyé à la fonction addMarker(loc,map,bulle) qui est la même que celle d'"index.php":

```









39 <!-- JAVASCRIPT -->
40 <script type="text/javascript" src="js-marker-clusterer-gh-pages/src/markerclusterer.js"></script> <!-- scrip MARKERCLUSTER -->
41 <script>
42 var markerCluster; // variable du MARKERCLUSTERER
43 var markers = []; // tableau des markers pour le MARKERCLUSTERER
44 function afficher() { // signature
45 alert("Projet réalisé par Rudolf MILLET dans le cadre du projet Web Mapping (ING2 2015/2016) !");
46 }
47 var map; // variable pour la carte google
48 function initialize() { // lors du chargement de la page
49 for(var i=1;i<=nb_ligne_geo;i++){ // parcours des geocodages
50 if(row_geo[i][1]==number){ // si il s agit du geocodage de l adresse
51 var latlng = new google.maps.LatLng(parseFloat(row_geo[i][3]),parseFloat(row_geo[i][4])); // extraction des coordonnees
52 var mapOptions = { // options de la carte
53 zoom:17,
54 center:latlng
55 }
56 break; // des qu on a un couple de coordonnees on arrete la boucle for
57 }else if(i==nb_ligne_geo){ // sinon on creer la carte de base
58 var latlng = new google.maps.LatLng(48.849335,2.687764);
59 var mapOptions = {
60 zoom:14,
61 center:latlng
62 }
63 }
64 }
65 map = new google.maps.Map(document.getElementById("map"),mapOptions); // carte Google
66 // affichage de toutes les coordonnees de l adresse
67 for(var i=1;i<=nb_ligne_geo;i++){ // parcours des geocodages
68 if(row_geo[i][1]==number){ // si c est le geocodage de l adresse choisie
69 var bulle = row_geo[i][2]; // afficher le service
70 console.log(bulle); // affiche l info bulle
71 var loc = { lat: parseFloat(row_geo[i][3]), lng: parseFloat(row_geo[i][4]) }; // extraction des coordonnees
72 console.log(loc); // affiche la localisation
73 addMarker(loc,map,bulle); // ajout du marker
74 }
75 }
76 markerCluster = new MarkerClusterer(map, markers); // creation du MARKERCLUSTERER
77 }
78 </script>

```

Ce qu'on peut améliorer

- Séparer le CSS et le JS dans des fichiers différents afin d'éclaircir le code PHP/HTML.
- Placer le code PHP qui interroge la BDD dans un fichier index appelé par les 2 pages.
- De même pour les fonctions JS identiques appelées par les 2 pages.
- Il est peut-être inutile de se connecter à la BDD à chargement de page, d'où l'idée de faire un fichier .php en index pouvant être appelé par les 2 pages.
- Optimiser les boucles for, la recherche dans les tableaux pour gagner du temps. Les break participent déjà mais ce n'est pas très propre.
- Optimiser le JS. Placer les déclarations de variables et de fonctions et leurs appels au bon endroit pour éviter les erreurs de chargement.

Bibliographie

	Documentation proposée dans le sujet : Pour le service de géocodage
	Explication du géocodage dans son ensemble : The Google Maps Geocoding API
	Documentation de la fonction pg_connect() : pg_connect
	Comment charger l'extension PHP "pg" : Fatal error: Call to undefined function pg_connect()
	Documentation de la fonction switch en PHP : switch
	Documentation sur l'utilisation des Marker Google : Personnaliser les marqueurs Google Maps
	Documentation sur le "MarkerClusterer" : Too Many Markers!
	Source des fichiers utilisés pour la fonction "MarkerClusterer" : googlemaps/js-marker-clusterer

Sommaire

Manuel programmeur.....	1
Parties communes.....	1
index.php.....	3
recherche.php.....	4
Ce qu'on peut améliorer.....	5
Bibliographie.....	6