Samuel BONNAN - Benjamin BARBE

Projet Web Services
DAGPI
2017/2018

https://github.com/Zouif/DAGPI-Projet---Web-Services

Application de gestion web de gestion de réunions pour les entreprises
Synchronisation avec une feuille de calcul Google et affichage sur une carte

Liste des APIs Google utilisées

- Google Sheets API (écrire les réunions dans une feuille de calcul)
- Google Maps Javascript API (afficher les réunions sur une carte)
- Google Maps Geocoding API (transformer une adresse en coordonnées géographique pour afficher les réunions sur la carte)
- Google Maps Directions API (pour afficher l'itinéraire pour se rendre à une réunion)

Prérequis

Il est possible d'utiliser un bundle type **xampp** ou **wamp**, ou installer séparément :

- Un serveur web **Apache**
- Un serveur de base de données **MySQL**

Installation

- Extraire l'archive dans le répertoire du serveur web
- L'application doit être extraite de manière à ce que le lien d'accès vers la page **meetingForm.php** , c'est à dire **http://localhost/DAGPI-Projet---Web-Services/site/meetingForm.php** (nécessaire pour l'authentification par Google)
- Exécuter le script SQL **company_web_service.sql** afin de créer la base de données
- Pour avoir des données de démonstration, exécuter le script **company_web_service-demo_data.sql**

Autoriser les cookies

Google Chrome :

- Aller dans **Paramètres** > **Paramètres avancés** > **(Confidentialité et sécurité) Paramètres de contenu** > **Cookies**
- Désactiver **Bloquer les cookies tiers**

Mozilla Firefox :

- Aller dans **Options** > **Vie privée** > **Règles de conservation : utiliser les paramètres personnalisés pour l'historique**
- Dans **Accepter les cookies**, sélectionner **toujours**

Utilisation

Connexion et utilisation :

- Se rendre sur le site : **http://localhost/DAGPI-Projet---Web-Services/site/**
- Se connecter avec l'identifiant **dagpi** et le mot de passe **dagpi**
- Accéder au tableur en cliquant sur **Tableur** dans le menu
- Créer une nouvelle réunion en cliquant sur **Ajouter un meeting** dans le menu
- Visionner l'emplacement des réunions en retournant sur l'**Index** dans le menu

Création d'un compte :

- Se connecter au compte Google Drive personnel
- Créer un tableur
- Le partager (Bouton "**Partager**" en haut à droite, "**Obtenir le lien de partage**", et "**Modifiable : utilisateurs avec le lien"**)
- Retourner sur l'application Web Service
- Cliquer sur **Inscription** dans le menu
- Remplir identifiant / mot de passe / identifiant du tableur
    - L'identifiant du tableur se trouve dans l'URL d'accès du tableur Google :
    - https://docs.google.com/spreadsheets/d/**1_-r1howh0qoDUZluj2OO-JmDWRgcoztLUcmWVu_daQM**/edit#gid=0 : Dans le tableur utilisé dans le projet, l'identifiant est **1_-r1howh0qoDUZluj2OO-JmDWRgcoztLUcmWVu_daQM**
- Valider le formulaire

Utilisation des APIs

Configuration prérequise du compte Google :

Dans la console Google, un paramétrage spécifique est nécessaire pour activer les APIs sur un nouveau compte Google. Celui fourni dans le projet est déjà paramétré.

Depuis la console Google (https://console.developers.google.com/) :

- Créer un nouveau projet, dans la **liste déroulante**, tout en haut à gauche
- Dans le menu de gauche, **Bibliothèque**, sélectionner et **Activer** les APIs suivantes
    - Google Maps JavaScript API
    - Google Maps Geocoding API
    - Google Sheets API
- Dans le menu de gauche, **Identifiants**, **Créer des identifiants** (**Clé API**)
    - La clé API sera utilisé pour l'API Google Maps
- Dans le menu de gauche, **Identifiants**,
    - Aller dans **Ecran d'autorisation OAuth** en haut
    - Renseigner un **Nom de produit** puis **Enregister**
    - Aller dans **Identifiants** en haut
    - Cliquer sur **Créer des identifiants** (**ID de client OAuth**)

- Sélectionner **Application Web**
- Dans **Origines JavaScript autorisées**, mettre **http://localhost**
- Dans **URI de redirection autorisés**, mettre le lien de la page qui sera ouverte une fois la connexion Google effectuée (pour le projet Web Services, **http://localhost/DAGPI-Projet---Web-Services/site/meetingForm.php**)
- Cliquer sur **Enregistrer**
- L'**ID client** sera utilisé dans l'API Google Sheets

API Google Maps :

- Le fichier HTML où la map doit être affichée doit inclure le fichier JS suivant :

```
<script
src="https://maps.googleapis.com/maps/api/js?key=CLE_API&callback=CALLBACK
_INITIALISATION" async defer></script>
```

La clé de l'API du compte Google utilisée pour le projet est **AIzaSyCE772SsPKfuXNhvj-OPuoS63idOw2yu00.**

Le callback d'initialisation dans l'application est **initMap.**

- La même page HTML doit contenir un div particulier dans le <body> :

```
<div id="map" style="width: 400px; height: 400px"></div>
```

Ce div sera mis à jour par l'API Google et contiendra la carte.

- Une fonction Javascript doit être définie pour charger la map. Son nom correspond au paramètre GET **CALLBACK_INITIALISATION** dans l'URL du fichier JS au dessus. La map est chargée dans le div définit au dessus

```
var map;
function initMap() {
      map = new google.maps.Map(document.getElementById('map'), {
          center: {lat: 44.863, lng: -0.621},
          zoom: 8
      });
}
```

On peut ensuite ajouter un marqueur sur la carte, et une fenêtre pop-up contenant des détails s'ouvrant lorsqu'on clique sur le marqueur :

```
geocoder.geocode( { 'address': 'Gradignan'}, function(results, status) {
      if (status == 'OK') {

      var contentString =
            '<p>Contenu HTML du popup</p>';

      var infowindow = new google.maps.InfoWindow({
            content: contentString
      });


      var marker = new google.maps.Marker({
            map: map,
            position: results[0].geometry.location
      });

      marker.addListener('click', function() {
            infowindow.open(map, marker);
      });
      }
});
```

API Google Sheets :

- L'API Sheets nécessite d'avoir accès au compte Google Drive utilisateur. La documentation de Google contient la page HTML type permettant de se connecter au compte Google à l'aide d'un algorithme Javascript.
- Une fois la connexion effectuée, il faut passer par l'objet javascript **gapi.client.sheets.spreadsheets** pour faire les opérations sur les feuilles de calcul.

Sur le projet de service web, c'est la requête d'insertion de données dans 'une feuille de calcul qui est effectuée :

```
gapi.client.sheets.spreadsheets.values.batchUpdate({
      "spreadsheetId" : sheetId,
      "valueInputOption": "USER_ENTERED",
      "data": [
      {
            "values": [
            [
                  title,
                  date_start,
                  date_end,
                  location,
            ]
            ],
            "range": "A1:D1",
      }
      ]
}).then(function(response) {
      console.dir(response);
}, function(response) {
      console.dir(response.result.error.message);
});
```

La méthode **gapi.client.sheets.spreadsheets.values.batchUpdate** permet de mettre à jour les cellules

Plusieurs paramètres sont utilisés dans la requête :

- **spreadsheetId** : l'identifiant de la feuille de calcul à modifier
- **valueInputOption** : de quelle manière les informations doivent être renseignées "USER_ENTERED" signifie que les informations sont entrées comme s'il s'agis d'un utilisateur qui clique sur la cellule et rentre la valeur. Il n'y aura aucun traitement sur la valeur.
- **data** : objet décrivant les données à insérer
    - **values** : les valeurs à mettre dans le tableur
    - **range** : les coordonnées de la plage à modifier

Deux fonctions sont passées dans la méthode chaînée **then**.

- La première décrit le comportement de la requête lors d'un succès
- La deuxième décrit le comportement de la requête lors d'une erreur